Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра спеціалізованих комп'ютерних систем

ШАКОВ Віталій Юрійович

КОМП'ЮТЕРНО-ІНТЕГРОВАНА СИСТЕМА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНОЇ ОБРОБКИ ВЕЛИКИХ ОБ'ЄМІВ ДАНИХ

спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
магістерська програма – Автоматизація та комп'ютерно-інтегровані технології

Магістерська робота

Виконав студент групи АКІТм-21
В.Ю. Шаков
_____
Науковий керівник:
к.т.н., доцент І.Р. Пітух
_____

Магістерську роботу допущено до захисту:
"____"_____20___ р.

Завідувач кафедри
_____ А.І. Сегін

Тернопіль 2023

Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра спеціалізованих комп'ютерних систем
Освітній ступінь "магістр"
спеціальність:151 – Автоматизація та комп'ютерно-інтегровані технології
освітньо-професійна програма – Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ
Завідувач кафедри СКС
_____ А.І.Сегін
"____"_____20__ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ШАКОВУ Віталію Юрійовичу
(прізвище, ім'я по-батькові)

1. Тема кваліфікаційної роботи
Комп'ютерно-інтегрована система підвищення ефективності розподіленої обробки великих об'ємів даних / Computer-integrated system for improving the efficiency of distributed processing of large volumes of data.
керівник роботи к.т.н., доцент І.Р. Пітух
затверджені наказом по університету від 8 грудня 2022 р. № 491

2. Строк подання студентом закінченої кваліфікаційної роботи
30 листопада 2023р.

3. Вихідні дані до кваліфікаційної роботи:
1. Технології розподіленого опрацювання даних.
2. Розподілене середовище обробки великих даних.
3. Паралельні обчислення.
4. Моделі опрацювання даних у розподілених системах.

4. Основні питання, які потрібно розробити
1. Дослідження технологій опрацювання даних у розподілених системах.
2. Аналіз моделей обробки даних та алгоритмів планування задач в кластерних обчисленнях.
3. Проектування комп'ютерно-інтегрованої системи підвищення ефективності розподіленої обробки великих об'ємів даних.

5. Перелік графічного матеріалу у роботі
1. Блок-схема адаптивного алгоритму планування виконання завдань.
2. Структура комп'ютерно-інтегрованої системи.
3. Схема роботи комп'ютерно-інтегрованої системи.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пітух І.Р. к.т.н., доцент кафедри СКС		
2	Пітух І.Р. к.т.н., доцент кафедри СКС		
3	Пітух І.Р. к.т.н., доцент кафедри СКС		

7. Дата видачі завдання 20 жовтня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження технологій опрацювання даних у розподілених системах	12.2022р. – 02.2023р.	
2	Аналіз моделей обробки даних та алгоритмів планування задач в кластерних обчисленнях	03.2023р. – 06.2023р.	
3	Проектування комп’ютерно-інтегрованої системи підвищення ефективності розподіленої обробки великих об’ємів даних	07.2023р. – 11.2023р.	

Студент _____ В.Ю. Шаков
(підпис)

Керівник роботи _____ к.т.н., доцент І.Р. Пітух
(підпис)

РЕФЕРАТ

Робота виконана на 73 сторінках та містить 29 рисунків, 4 таблиць, 2 додатки, 30 джерел за переліком посилань.

Мета кваліфікаційної роботи. Дослідження та розробка комп’ютерно-інтегрованої системи підвищення ефективності розподіленої обробки великих об’ємів даних.

Результати роботи. Розроблена комп'ютерно-інтегрована система дозволяє вирішити проблему балансування навантаження, враховуючи поточний стан ресурсів. Це дозволяє системі ефективно реагувати на змінні обчислювальні навантаження та уникати перевантажень або недостатнього використання ресурсів. Запропонований алгоритм, який базується на існуючих методах, але додатково адаптований і оптимізований з урахуванням гетерогенності, віддаленості та завантаженості, має інноваційні особливості та дозволяє покращити ефективність обробки завдань.

Рекомендації по використанню результатів роботи. Отримані результати можуть мати позитивний вплив на бізнес-середовище, наукові дослідження та галузі, де використовуються розподілені обчислення, завдяки підвищенню продуктивності, надійності та зниженню витрат.

Ключові слова: КОМП’ЮТЕРНО-ІНТЕГРОВАНА СИСТЕМА, ВЕЛИКІ ОБ’ЄМИ ДАНИХ, РОЗПОДІЛЕНА ОБРОБКА, КЛАСТЕР-СИСТЕМА.

ABSTRACT

Work is executed on 73 pages and including 29 illustrations, 4 tables, 2 additions, 30 sources after the list of references.

The purpose of the qualification work. Research and development of a computer-integrated system for enhancing the efficiency of distributed processing of big data.

Research results. Developed Computer-Integrated System addresses the load balancing challenge, taking into account the current state of resources. This allows the system to effectively respond to changing computational workloads and avoid overloads or underutilization of resources. The developed algorithm, based on existing methods but additionally adapted and optimized with consideration for heterogeneity, distance, and load, possesses innovative features and enhances task processing efficiency.

Recommendations for the use of work results. The obtained results can have a positive impact on the business environment, scientific research, and industries that utilize distributed computing, by increasing productivity, reliability, and reducing costs.

Keywords: COMPUTER-INTEGRATED SYSTEM, BIG DATA, DISTRIBUTED PROCESSING, CLUSTER SYSTEM.

ЗМІСТ

ПРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД - база даних;

ВОД - великі обсяги даних;

КІС - комп’ютерно-інтегрована система;

ОР - обчислювальні ресурси;

ОС – операційна система;

ПЗ – програмне забезпечення

ПВЗ – планування виконання завдань;

ПО - паралельні обчислення;

РО - розподілена обробка;

РРЧ - режим реального часу;

РС - розподілене середовище;

РСОД - розподілені системи обробки даних;

СД – сховища даних;

СРО - система розподілених обчислень;

ТРОД - технології розподіленого опрацювання даних.

ВСТУП

Актуальність теми. На даному етапі світ переживає експоненційний ріст обсягів інформації, який спричиняють Інтернет-речей, соціальні мережі, датчикам і іншим джерела. Збільшення обсягів даних вимагає більш потужніших і ефективніших інструментів для їх опрацювання та аналізу, особливо в тих галузях, де важливо реагувати на події в реальному часі, наприклад, у медицині виробництві та кібербезпеці.

Підприємствами постійно здійснюється пошук шляхів підвищення конкурентоспроможності, аналіз даних для знаходження нових можливостей, вдосконалення процесів та розуміння клієнтів. У багатьох галузях науковцями використовуються дані у великих обсягах для розробки нових методів, моделей і рішень. Дані можуть допомогти вирішувати важливі соціальні проблеми, такі як передбачення епідемій, управління міськими ресурсами та розвиток інфраструктури.

Підприємства та уряди шукають способи вдосконалити свої операції через цифрову трансформацію. Ефективна обробка даних великих обсягів є ключовим аспектом цього процесу, а також представляють виклики для систем, які можуть швидко аналізувати та реагувати на загрози, набувають особливого значення.

Тому розробка комп’ютерно-інтегрованої системи (КІС) [1-3] для підвищення ефективності розподіленої обробки (РО) [4-8] даних великих обсягів (ВОД) [9-12] має на сьогодні ключове значення як для бізнесу так і для науки для вирішення важливих завдач.

Чим раз тим більше коло задач у науці, інженерії та бізнесі стає настільки обчислювально складними і розмірними, що навіть високопродуктивні комп'ютери не здатні їх опрацювати. Це вимагає нових підходів до опрацювання й аналізу даних. Використання кластер-систем для РОВОД надає можливість об'єднувати обчислювальні ресурси (ОР), що можуть відноситись до різних організацій та бути розташованими в різноманітних територіальних регіонах. Це дозволяє розв'язувати більш

складні завдання та забезпечує гнучкість в управлінні ресурсами. Вартість централізованої обробки даних на суперкомп'ютері або кластері значно вища, ніж вартість декількох типових комп'ютерів. Вказаний спосіб буде більш вигідним економічно. У випадку відмови каналу зв'язку у системах розподілених обчислень (СРО) може забезпечуватися локальна обробка, що дозволить підвищити надійність та доступність системи в цілому. Перехід до РОВОД створює нові організаційні виклики, оскільки вимагає змін у способах управління та координації роботи різних обчислювальних ресурсів.

Ці аспекти підкреслюють актуальність розробки КІС для СРО, які б допомагали вирішувати складні завдання в умовах зростаючого обсягу і складності даних і обчислень. Ефективність таких КІС проявляється як у плані економічних витрат, так і щодо надійності й продуктивності, що робить цю тему надзвичайно важливою для подальшого розвитку інформаційних технологій.

Мета і завдання дослідження полягають у розробці КІС, спрямованої на підвищення ефективності РОВОД в різних сферах діяльності, для максимізації економічної вигоди, надійності та підвищення продуктивності, враховуючи організаційні аспекти та вимоги автоматизації.

Для того, щоб досягнути мети потрібно:

– дослідити технології розподіленого опрацювання даних;

– проаналізувати можливості розподілене середовище та паралельні обчислення;

– провести аналіз моделей опрацювання даних у кластер-системах;

– обгрунтувати вибір алгоритму розпаралелення задач;

– розробити комп’ютерно-інтегровану систему та оцінити ефективність запропонованого алгоритму.

Об’єкт дослідження: розподілене опрацювання даних.

Предметом дослідження є КІС підвищення ефективності РОВОД.

Наукова новизна одержаних результатів полягає у тому, що в проектованій КІС запропоновано алгоритм розподілу задач, що враховує

віддаленість та гетерогенність ОР, та забезпечує збалансоване навантаження в умовах великих системах РОВОД.

Практичне значення отриманих результатів. Застосування КІС дозволить підприємствам та організаціям оптимізувати використання своїх ОР, що забезпечить підвищення продуктивності та призведе до зниження витрат на обладнання та електроенергію. Врахування віддаленості ресурсів і поточної завантаженості допоможе уникнути перевантажень та покращити надійність СРО.

Апробація.

1. «Стихальська С.В., Шаков В.Ю. Комп'ютерно-інтегрована система розподіленої обробки даних / Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно-інтегровані технології» (АКІТ - 2023), Тернопіль, 2023. -с. 60-63» [13].

2. «Драпак В.І., Питель Р.О., Романів А.М., Шаков В.Ю. Підвищення ефективності опрацювання даних у розподілених системах / Матеріали науково-практичного симпозіуму «Захист інформації», Тернопіль, 2023. - с. 67-72» [14].

1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ОПРАЦЮВАННЯ ДАНИХ У РОЗПОДІЛЕНИХ СИСТЕМАХ

1.1 Технології розподіленого опрацювання даних

Сучасні технології розроділеного опрацювання даних (ТРОД) включають ряд процесів, які дозволяють обробляти, передавати і зберігати дані в розподіленому середовищі [4-9]. Вони базуються на розподілених обчислювальних платформах, таких як кластери, хмарні обчислення та контейнери, які дозволяють розподілити завдання між різними ОР для паралельної обробки. Для того, щоб здійснювати управління завданнями та збалансовувати навантаження використовуються різні алгоритми та техніки, такі як адаптивне призначення ресурсів, розподілення задач за пріоритетами, та інші. Частина із алгоритмів можуть базуються на машинному навчанні.

Існує кілька переваг ТРОД, наприклад:

1. Масштабованість - розподілені системи обробки даних (РСОД) характеризуються високою масштабованістю вгору/вниз, оскільки існує можливість додавати або видаляти сервери, що дозволяє адаптувати їх під певні потреби, що є особливо важливим в умовах змінного навантаження

2. Гнучкість та адаптивність - ТРОД дозволяють реалізовувати гнучкі і адаптивні системи, що дозволять швидко реагувати на зміни у вимогах та завданнях.

3. Безпека та надійність - більша стійкість до системних збоїв, наприклад, один із серверів дає збій, то решта можуть продовжити обробку даних. За допомогою дублювання та резервування ресурсів можна забезпечити високий ступінь надійності й доступності.

4. Підвищена продуктивність: завдяки ТРОД робоче навантаження розподіляється між кількома серверами, що дозволяє паралельно обробляти завдання на різних обчислювальних ресурсах, що прискорює час обробки. Технології дозволяють працювати з ресурсами, які розміщені в різних точках світу, забезпечуючи глобальний доступ до обчислювальних можливостей та

забезпечують роботу з ВОД.

5. Зниження витрат: ефективне використання ОР дозволяє оптимізувати їх використання і знизити витрати на обладнання, електроенергію та інфраструктуру.

З переліченого випливає, що ТРОД можуть використовуватися в різних галузях, наприклад фінанси, медицина, наукові дослідження, електроннау комерцію та багато інших оскільки дозволяють оптимізувати обробку ВОД та складних даних, що робить її важливим інструментом в епоху цифрової трансформації.

Аналіз існуючих ТРОД допоможе приймати обгрунтовані рішення під час дослідження й розробки проектованої системи для досягнення її максимальної ефективності та продуктивності.

1.1.1 Hadoop

Фреймворк Hadoop [15] має доступний для перегляду та модифікації код і призначений для зберігання та опрацювання ВОД розроблений у 2005 році Дугом Катінгом і Майком Кафареллою. Ядром структури Hadoop є розподілена файлова система (HDFS), яка призначена для зберігання та керування ВОД на кількох вузлах у кластері. HDFS базується на файловій системі Google (GFS) і оптимізована для обробки ВОД. Hadoop також включає модель програмування MapReduce, яка використовується для обробки даних на кількох вузлах у кластері (рисунок 1.1).

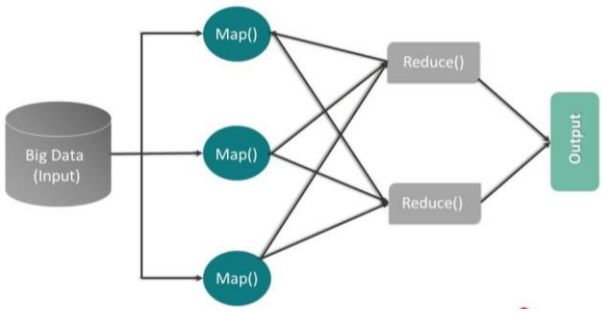


Рисунок 1.1 – Архітектура Hadoop

Hadoop має кілька компонентів, зокрема (рисунок 1.2):

– HDFS - рівень зберігання, який призначений для зберігання ВОД

на кількох вузлах у кластері. Він відмовостійкий, тобто може обробляти збої окремих вузлів без втрати даних.

– MapReduce - рівень обробки, який використовується для обробки даних на кількох вузлах у кластері. MapReduce працює, розбиваючи дані на менші фрагменти, обробляючи ці фрагменти паралельно, а потім об’єднуючи результати.

– YARN - рівень, який використовується для керування ресурсами в кластері Hadoop. Він відповідає за перерозподіл ресурсів для програм, що працюють у кластері, і за моніторинг цих програм.

– Hadoop Common є наборомутиліт та бібліотек, що використовують всі компоненти Hadoop.

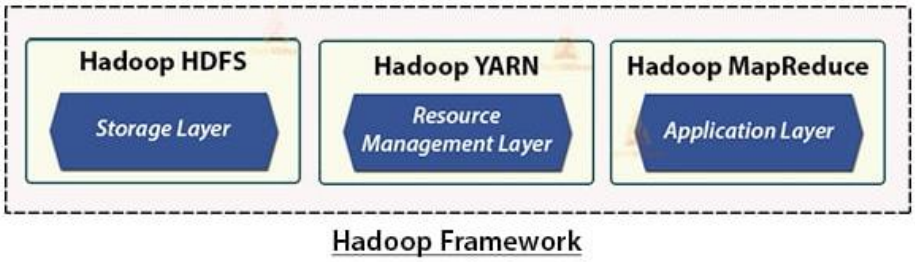


Рисунок 1.2 – Компоненти Hadoop

Hadoop широко використовується завдяки своїй масштабованості, відмовостійкості та економічній ефективності. Hadoop можна запускати на звичайному апаратному забезпеченні, що зумовлює його економічну ефективність, порівняно з традиційними системи опрацювання даних.

1.1.2 Apache Spark

Spark [16] це РСОД з відкритим кодом, призначена для обробки ВОД розроблена у відповідь на обмеження, що має MapReduce, яка використовується фрейморком Hadoop (рисунок 1.3). Хоча MapReduce ефективний для пакетної обробки, він не дуже підходить для обробки в режимі реального часу (РРЧ) чи для ітераційної обробки, а Spark розроблений для роботи з такими типами навантажень.

Ядром Apache Spark є абстракція Resilient Distributed Datasets (RDD),

яка забезпечує зберігання та обробку даних на кількох вузлах у кластері. RDD є незмінними, тобто їх не можна змінити після створення. Це дозволяє Spark виконувати операції над RDD паралельно, не турбуючись про конфлікти між кількома потоками.

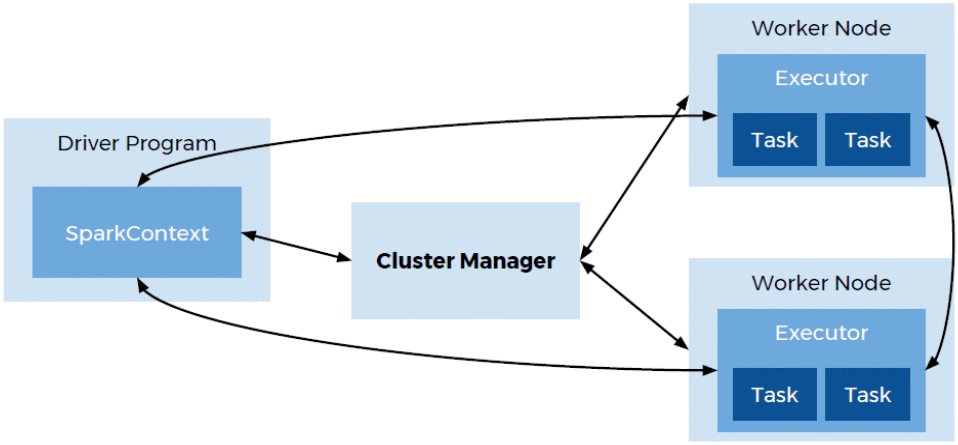


Рисунок 1.3 – Архітекрура Spark

Spark також включає кілька бібліотек для машинного навчання (MLlib), обробки графів (GraphX) і обробки потоків (Spark Streaming). Ці бібліотеки надають API високого рівня для типових завдань опрацювання даних, що спрощує розробникам задачу написання програм з ТРОД (риуснок 1.4).

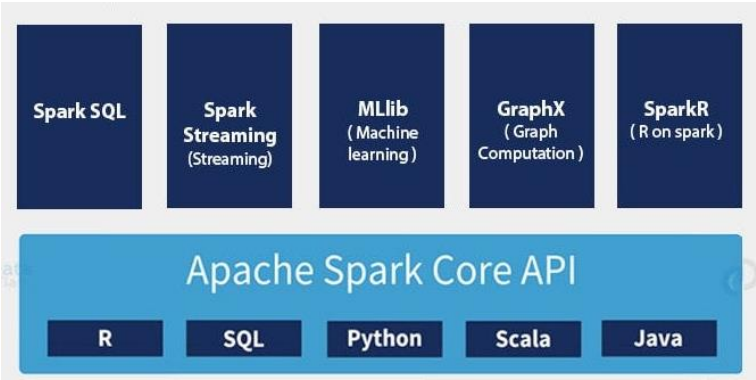


Рисунок 1.4 – Компоненти Spark

Spark має кілька переваг перед Hadoop, зокрема:

– швидкість - Spark може обробляти дані до 100 разів скоріше, за Hadoop, завдяки можливостям обробки в пам’яті;

– опрацювання РРЧ - розроблений для роботи з робочими навантаженнями обробки в РРЧ, що робить Spark придатним для програм, які

вимагають обробки з низькою Затримкою

– ітераційна обробка використовується для робочих навантажень машинного навчання;

– Spark надає API високого рівня для типових завдань опрацювання даних, що полекшує розробникам написання розподілених програм опрацювання даних.

1.1.3 Apache Flink

Flink [17] - це РСОД, що забезпечує обробку потокових даних і пакетну обробку. Спочатку розроблений Технічним університетом Берліна в 2009 році, а відкритий у 2014 році. Flink призначено для роботи з робочими навантаженнями у РРЧ, яким потрібна обробка з низькою затримкою.

Архітекрура Flink наведена на рисунку 1.5.

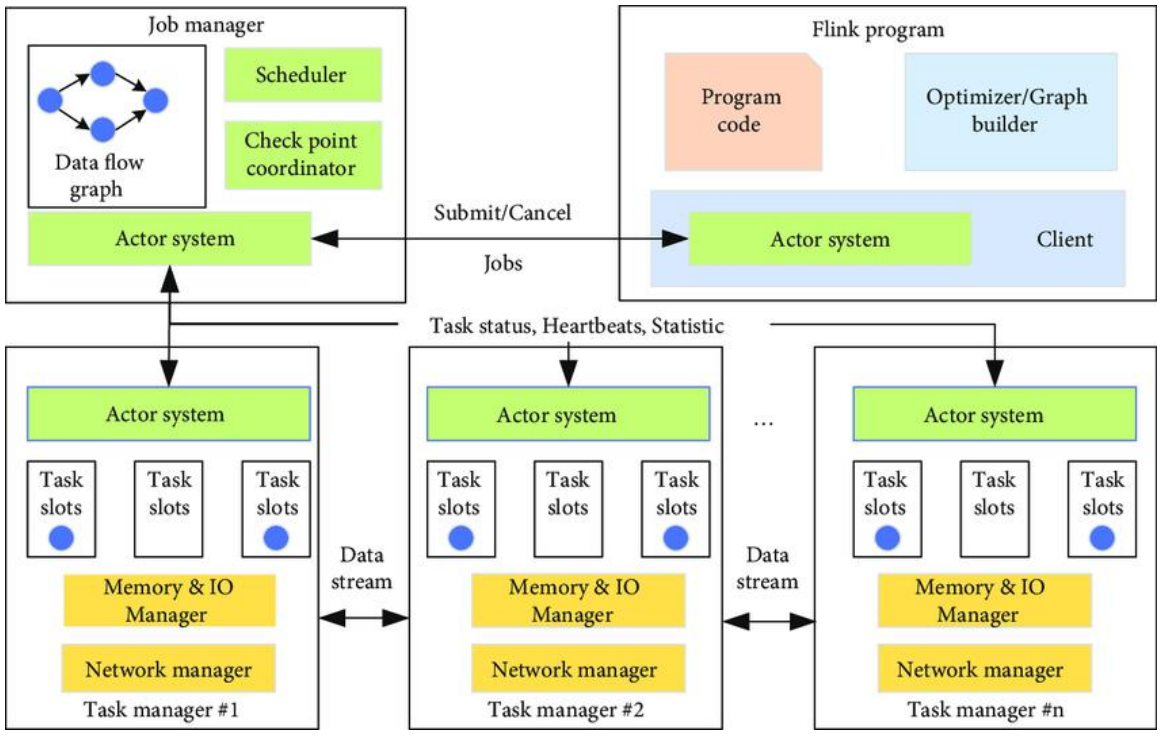


Рисунок 1.5 – Архітекрура Flink

Основою технології є Flink API, який дозволяє розробникам писати код, що використовується як для потокової, так і для пакетної обробки, що базуються на єдиному API:

– API DataStream - для опрацювання потокових даних у РРЧ;

– API DataSet - використовується для опрацювання пакетних даних.

Flink використовує механізм розподіленого потоку даних для паралельного виконання завдань обробки даних на кількох вузлах у кластері. Flink також може автоматично обробляти відмову та відновлення, що забезпечує його переваги для критично важливих програм.

Також Flink містить кілька бібліотек для машинного навчання (FlinkML), обробки графіків (Gelly) і SQL (Flink SQL). Ці бібліотеки надають API високого рівня для звичайних завдань опрацювання даних і полекшують розробникам написання програм РОД (рисунок 1.6).

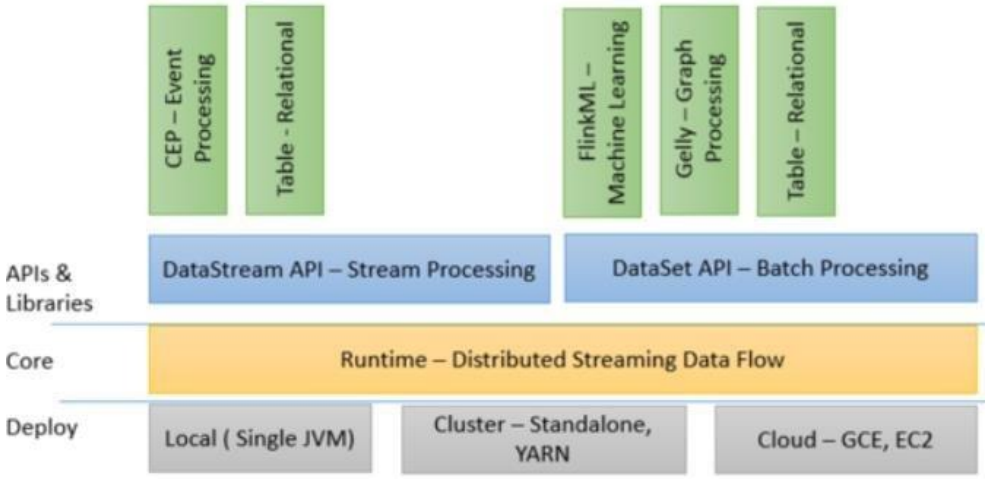


Рисунок 1.6 – Компоненти Flink

Flink має кілька переваг перед іншими системами РОД, зокрема:

– обробка з низькою затримкою – дозволяє опрацьовувати навантаження в РРЧ, які вимагають обробки з низькою затримкою;

– відмовостійкість - дозволяє автоматично обробляти перехід після відмови та відновлення;

– гнучкі API як для пакетної, так і для потокової обробки інформації, що полегшує розробникам написання програм;.

– розширені функції Flink, такі як підтримка складної обробки подій, інкрементної обробки та обробки стану.

В останні роки Flink набув популярності як швидка та гнучка РСОД і зараз широко використовується у виробничих середовищах.

1.1.4 Cassandra

Cassandra [18]є системою керуваня розподіленими базами даних (БД) з вихідним кодом, що є відкритий. Розроблена для обробки ВОД на кількох вузлах у кластері. Cassandra розроблена Facebook у 2008 році, а відкрита у 2010 роціу якості високодоступної, відмовостійкої та масштабованої сиситеми (рисунок 1.7).

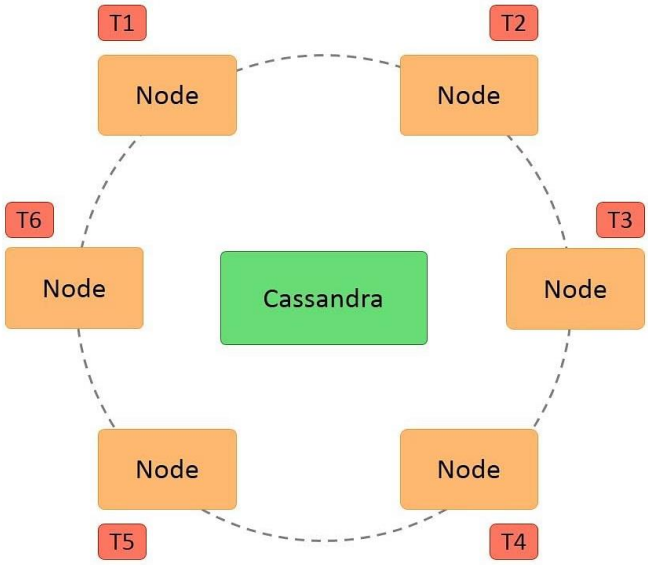


Рисунок 1.7 – Архітекрура Cassandra

Cassandra базується на розподіленій архітектурі, тобто дані зберігаються на кількох вузлах у кластері. Використовується кільцева архітектура, в якій кожен з вузолів у кластері відповідає за зберігання частини даних. Це дозволяє Cassandra обробляти ВОД і забезпечує доступність, навіть у випадку виходу з ладу деяких вузлів кластеру.

Cassandra є БД NoSQL, тобто вона не використовує традиційну модель реляційної БД. Натомість дані зберігаються в моделі сімейства стовпців, де дані організовані в рядки та стовпці. Cassandra також підтримує модель гнучкої схеми, яка дозволяє динамічно оновлювати схему БД без простоїв.

Cassandra надає кілька функцій, які роблять її ідеальною для роботи з ВОД, зокрема:

– висока масштабованість - може обробляти ВОД на кількох вузлах у кластері;

– висока доступність та відмовостійкість - може автоматично обробляти збої вузлів і невідповідності даних;

– гнучка модель даних, що дозволяє динамічно оновлювати схему бази даних без простоїв;

– продуктивність - забезпечується обробкаи ВОД з мінімальною затримкою.

Cassandra використовується рядом організацій для обробки ВОД, зокрема Apple, Netflix і eBay. ЇЇ також використовують кілька інших проектів Apache, включаючи Apache Spark і Apache Hadoop, для опрацюваннята та збереження даних.

Spark, Flink і Cassandra можна запускати в автономному кластері або в кластері, керованому менеджером кластера, таким як Apache Mesos, Hadoop YARN або Kubernetes. Cassandra також можна запускати за допомогою хмарних платформ, наприклад «Amazon Web Services (AWS), Microsoft Azure чи Google Cloud Platform (GCP)» [10].

1.2 Розподілене середовище та паралельні обчислення

Автоматизація процесів управління обчисленнями [1-4] стає дедалі важливішою задачею. Ряд технологій, таких як наприклад, Kubernetes, Docker Swarm [19] та ін. допомагають у керуванні контейнерами та синхронізації ресурсів. Розподілене середовище (РС) [20-22] стикається з викликами досягненя конфіденційності даних та їх захисту. Технології шифрування, контролю доступу та моніторингу безпеки відіграють важливу роль у цьому контексті. ВОД вимагають ефективних інструментів для аналізу та обробки. Розглянуті ТРОД та різні БД для ВОД, допомагають вирішувати завдання аналітики в РС. Використання хмарних платформ для СРО стає все більш популярним. Публічні та приватні хмарні рішення надають доступ до ОР на вимогу.

РС часто будується на мікросервісній архітектурі, де різні компоненти взаємодіють за допомогою API. Це дозволяє створювати гнучкі та

масштабовані СРО (рисунок 1.8). У цілому ТРОД дозволяють створити ефективні інформаційні системи, які опрацьовуватимуть ВОД, забезпечуючи швидкий доступ та високу доступність.

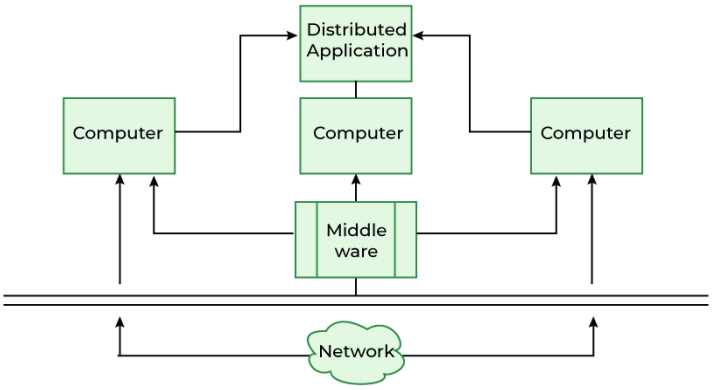


Рисунок 1.8 - Розподілена система

Зазначені технології і підходи дозволяють оптимізувати РС зробити їх продуктивнішими та ефективнішими, що є визначальним у різних галузях, в яких важлива обробка ВОД , зокрема, гетерогенних. ТРОД є складовою обробки ВОД, оскільки дозволяють підприємствам опрацбовувати інформацію швидше та ефективніше. Розглянувши і дослідивши основні ТРОД, визначено, що в кожної з них є свої плюси і негативні сторони, а їх вибор залежатиме від вимог розв’язуваної задачі.

РС є ключовим інструментом технолоігї ТРОД. Вони представляють собою комплексні архітектури, які дозволяють об'єднати різні ОР і обчислювальні вузли у єдину систему для оптимізації опрацювання і управління даними в СРО.

Інакше кажучи РС є мережею ОР, де обмін інформацією реалізований шляхом передавання повідомлень. Це допомагає в спільному використанні ресурсів системи та дозволяє комп’ютерам координувати свою роботу, щоб користувачі сприймали систему як єдиний інтегрований обчислювальний засіб. У РС різні ОР (що можуть фізично розташовуватися в різних точках) спільно працюють як єдина логічна система для вирішення спільних завдань. Ресурси у розподілених системах можуть включати обчислювальні вузли, сховища даних (СД), мережеві ресурси тощо. РС можуть бути локальними,

наприклад, локальна мережа, або глобальними, наприклад, Інтернет.

Паралельні обчислення ПО - це парадигма ТРОД та виконання завдань, в якій обчислювальні завдання розпаралелюються між вузлами мережі для паралельного або послідовного виконання (рисунок 1.9).

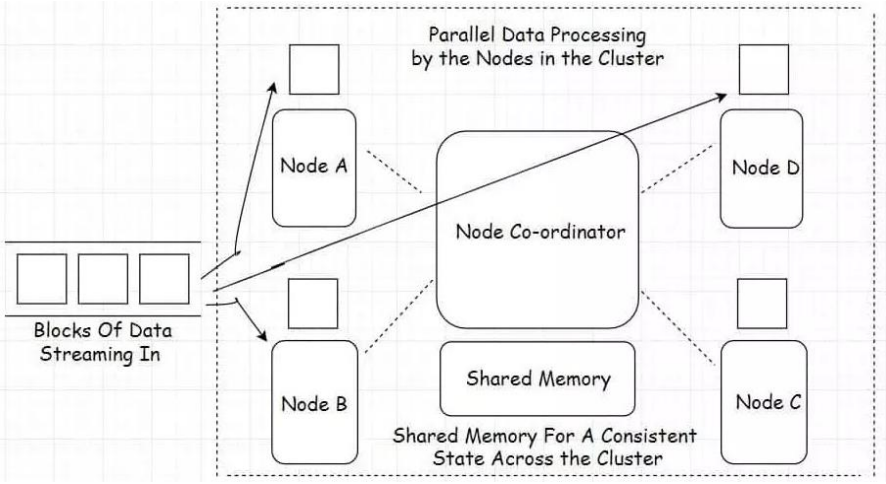


Рисунок 1.9 – Паралельні обчислення

Основною метою РС та ПО є:

– розподіл обчислювальної роботи для підвищення швидкодії та ефективності опрацювання даних;

– розподілення ресурсів та завдань для збільшення продуктивності, надійності та масштабованості.

РС та ПО широко використовуються в різноманітних сферах, включаючи обчислення ВОД, хмарні обчислення, мережеві послуги, інтернету речей (IoT) та можуть розглядатися в більшу вузькому аспекті, зокрема, щодо застосування їх у вирішенні великих ресурсоємістних задач, що відображає більше конкретну їх концепцію та використання. За цією інтерпретацією, РС та ПО використовуються для розподілу складних задач на багато менших обчислювальних блоків і їх обробки паралельно або послідовно на різних ОР, що дозволяє вирішувати задачі, що були б надто великими або часо- та ресурсомісткими для одного сервера чи комп'ютера.

Ефективність даного підхіду для обчислень ВОД, наукових обчислень, моделювання, обробки відео, аналізу біологічних даних та інших сфер, де завдання вимагають значних ОР є беззаперечною ТРОД в цьому контексті

включають використання кластерів обчислювальних вузлів, обчислювання в хмарних середовищах або навіть використання спеціалізованих обчислювальних кластерів - суперкомп'ютерів.

Особливостями організації паралельних та ПО використовуючи РС є розподілення завдань на обчислювальні вузли, які фізично віддалені поміж собою, що створює унікальні виклики та можливості для паралельного обчислення. Основні аспекти включають:

– ефективне керування завданнями - це ключовий аспект, оскільки РС повинна визначити задачі, які можуть виконуватися паралельно та розподілити їх між ОР;

– спільний доступ до ресурсів - обчислювальні вузли в РС можуть ділитися ресурсами, наприклад СД або мережеві з'єднання, тому . важливим завданням є організація доступу до ОР та управління конфліктами;

– передача даних - обмежуючим фактором при ПО може виступати обмін даними, а також швидкість передачі та механізми синхронізації даних;

– синхронізація та координація допомагають уникнути конфліктів та забезпечити коректну обробку даних;

– відмовостійкість та відновлення - у РС, де багато ресурсів, важливо мати механізми відмовостійкості та здатність відновлюватися в разі відмови обчислювальних вузлів;

– забезпечення захисту даних та доступу є безумовним, особливо у СРО з великим числом ресутрів або користувачів;

– можливість підключення нових ОР або вузлів при зростанні завдань.

Загалом, організація ПО у РС вимагає детального планування, архітектурного розгляду та вдосконалених алгоритмів керування ОР для досягнення максимальної ефективності вирішення завдань.

При розгляді СРО з апаратної точки зору важливо враховувати, які СД, мережеві та ОР використовуються і яким чином вони організовані. Основні аспекти СРО з апаратної точки зору включають:

– обчислювальні вузли - комп'ютери або сервери розміщені в різних місцях повинні бути підключені до мережі для виконання завдань;

– мережеве з'єднання – забезпечення швидкого і надійного обміну інформацією між вузлами, наприклад WAN, LAN, Інтернет та ін.;

– сховища даних - ресурси для зберігання, наприклад сервери-сховища, хмарні СД, БД доступ до яких повинен бути швидким і надійним;

– апаратна архітектура - конфігурації ОР, тип процесорів, обсяг пам'яті, обсяги СД та ін.;

– безпека апаратних ресурсів та мережевих з'єднань, включаючи фізичну безпеку приміщень, захист мережевого з'єднання та обладнання;

– відмовостійкість - організація апаратного та ПЗ з можливістю відновлення в разі відмови обчислювальних вузлів або інших апаратних несправностей;

– апаратна масштабованість - можливість додавати нові обчислювальні ресурси при зростанні завдань.

При розгляді РС з апаратної точки зору важливо враховувати вимоги до продуктивності, надійності, безпеки та масштабованості для досягнення успішного функціонування системи.

«СРО це мережа автономних вузлів, які виконують незалежні програми і спілкуються за допомогою повідомлень для обміну даними і координації» [21]. Вузлами є комп'ютери, процесори та процеси, кожен з яких має власний стан, який ізольований від інших процесів (рисунок 1.10).

Швидкості виконання різних операцій у такій системі непередбачуві і можуть відрізнятися, а час доставки повідомлень є невизначеним. Визначення СРО також може застосовуватися і до програмних систем, де взаємодіють процеси, що виконуються на одному обчислювальному пристрої. У такому випадку взаємодія процесів відбувається через колективну пам'ять, а не через мережу. Однак, в більшості випадків, СРО все ж включають декілька процесорів, що спілкуються між собою шляхом за допомогою ріхних каналів комунікації.

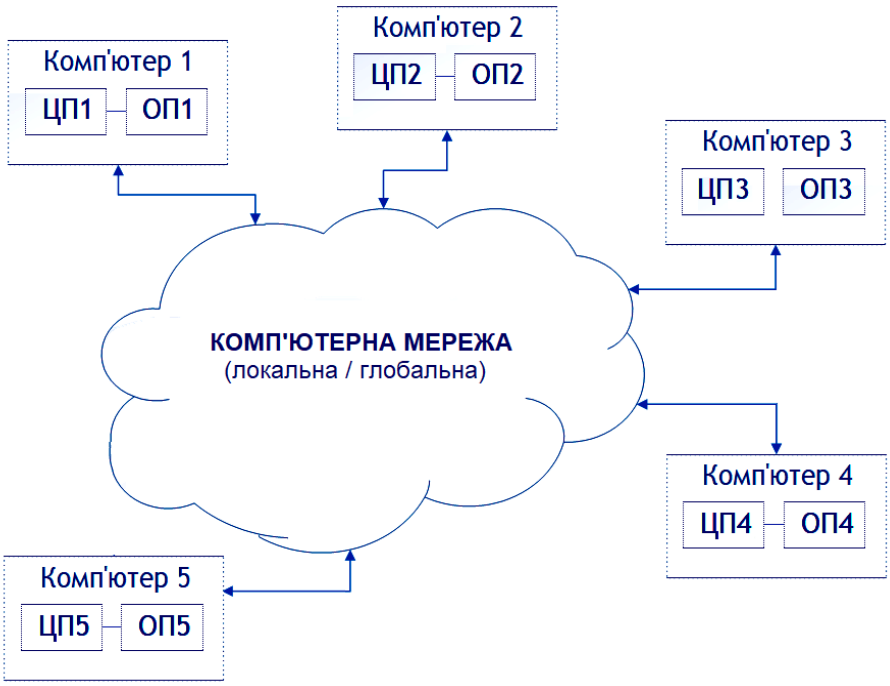


Рисунок 1.10 - Об’єднання незалежних ресурсів у мережу

Основні ознаки, що характеризують сутність СРО, і визначають їх специфіку в порівнянні з централізованими, наступні:

– автономність - кожен з вузлів (комп'ютер, процес або сервер) в розподіленій системі має власну автономію та незалежність у виконанні завдань;

– спільний доступ - можливе спільне використання ОР, наприклад СД або мережеві з'єднання;

– комунікація - зваємозвязок між вузлами реалізується через передачу повідомлень;

– незалежність від апаратного обладнання – передбачається робота РС на різних апаратних платформах та архітектурах;

– відмовостійкість - СРО має бути стійкою до відмов та здатною до відновлення в разі неполадок в окремих вузлах;

– мережеве середовище - взаємодія вузлів відбувається через мережу, наприклад локальну або розширеною до глобальної мережі;

– паралельність та одночасність – в РС багато завдань можуть виконуватися паралельно, і вона здатна керувати одночасністю обчислень;

– синхронізація та координація вузлів для уникнення конфліктів та забезпечення коректної роботи системи;

– масштабованість – можливість включення додаткових вузлів або ресурсів при зростанні завдань;

– доступність і безпека даних.

Основні відмінності, що характерні СРО:

– відсутність єдиного часу - компоненти СРО можуть працювати в різний час без глобальної синхронізації;

– не використовується спільна пам'ять - компоненти не мають загального доступу до неї, оскільки обмін даними реалізується через передачу повідомлень;

– географічне розподілення – «вузли системи можуть знаходитися на великих відстанях один від одного, проте вони все одно вважатимуться частиною СРО, включаючи локальні кластери» [20];

– незалежність та гетерогенність – кожен з вузлів можуть характеризуватися певіним набором параметрів, наприклад щодо обладнання, операційні системи (ОС) і продуктивність.

1.3 Ціль розробки систем опрацювання даних у розподіленому середовищі

Головною метою побудови РСОД є створення ефективного та надійного РС для опрацювання ВОД та виконання завдань, що може забезпечити кращу продуктивність, доступність та відмовостійкість. Це досягається шляхом розподілення ОР та завдань між вузлами РСОД та забезпеченням їх взаємодії у мережі. Ці системи широко використовуються в таких областях, як обчислення в хмарі, мережі, банкінг, наукові дослідження та інші, де важливою є обробка ВОД та надійність системи [5-9, 20-22]. Цілі побудови РСОД включають такі аспекти:

– використання розпаралелення ОР для прискорення обчислень та

оптимізації завдань;

– стійкість до відмов окремих компонентів, що забезпечує надійність роботи в умовах несправностей;

– легкість масштабування в результаті додавання нових ресурсів;

– доступність до ОР та послуг з різних місць і за різних умов;

– спільний доступ і використання ОР, що розміщено в різних географічних регіонах.

РСОД стали все більш важливими в багатьох областях та їх роль зростає, До основних причин цього можна віднести:

1. Зараз обчислювальні середовища мають тенденцію бути розподіленими географічно. Прикладом є банківські мережі, де багато банків обслуговують клієнтів і потребують міжбанківської взаємодії через розподілені системи. Іншим прикладом є Інтернет, який є глобальною розподіленою системою.

2. Зі стрімким наближенням традиційних однопроцесорних систем до меж продуктивності зростає попит на покращення продуктивності обчислень. РСОД надають змогу використовувати різноманітні механізми паралельного обчислення та спільно використовувати ОР для підвищення продуктивності.

3. РСОД дають змогу користувачам та додаткам спільно використовувати віддалені ресурси, наприклад, користувач одного ОР може користуватися дисковим просторот іншого ОР для зберігання файлів.

4. РСОД можуть бути стійкими до часткових збоїв, тобто система продовжуватиме функціонування при непрацездатності або некоректній роботі деяких компонентів. Ця відмовостійкість досягається за рахунок збитковості, де система може знаходити резервні ресурси для заміщення відмови.

2. АНАЛІЗ МОДЕЛЕЙ ОБРОБКИ ДАНИХ ТА АЛГОРИТМІВ ПЛАНУВАННЯ ЗАДАЧ В КЛАСТЕРНИХ ОБЧИСЛЕННЯХ

2.1 Модель опрацювання даних у розподілених системах

Моделлю опрацювання даних у РС є теоретичний або концептуальний підхід до опрацювання даних у РСОД (рисунок 2.1), що описує, яким саме чином дані обмінюються, обробляються та зберігаються [23-26].

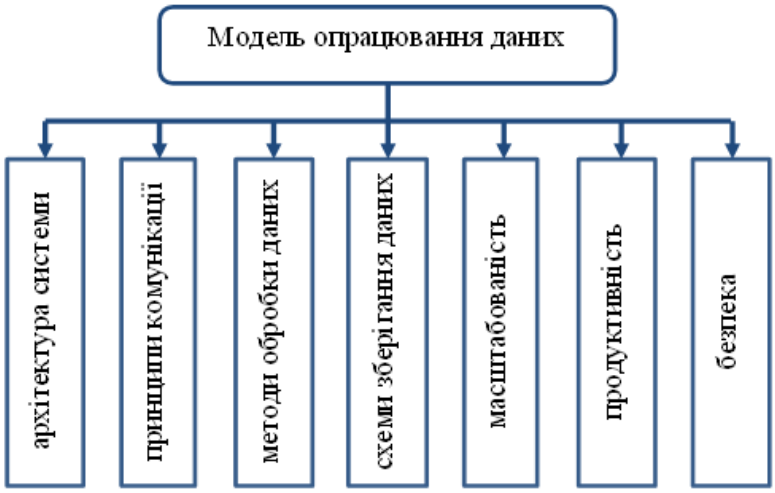


Рисунок 2.1 – Основні аспекти моделі опрацювання ВОД

Модель опрацювання ВОД включає такі ключові аспекти:

– архітектура системи – модель, що описує загальну архітектуру системи, включаючи розтміщення та взаємозв'язки компонентів системи, такі як сервери, клієнти, БД та інші ОР;

– принципи комунікації - визначають, як взаємодіють компоненти системи, наприклад через обмін повідомленнями, запитами та відповідями, включаючи протоколи комунікації та механізми синхронізації;

– схеми зберігання даних - модель визначає, де та яким чином зберігаються в системі дані, що включє розподілені БД, хмарні сховища, локальні СД на серверах;

– методи обробки даних - описують, як дані обробляються в системі, включаючи алгоритми та методи обчислень, які використовуються при аналізі та обробці інформації.

– масштабованість та продуктивність - модель включає стратегії для підтримки масштабованості системи, щоб забезпечити продуктивність у відповідь на збільшення розмірів даних та завдань їх опрацювання;

– забезпечення безпеки - враховує аспекти безпеки та доступу у РС. Дана модель може використовуватися для проектування та аналізу розподілених систем, а також для визначення оптимальних стратегій управління даними. У контексті моделі користувачу надається можливість роботи, що включає мережеві служби та прикладні процеси, які розташовані в різних віддалених абонентських системах. Це дозволяє вирішувати різні види задач:

– віддалений запит - користувач може відправити окремий запит на виконання обробки даних на віддалений сервер або процес.;

– віддалена транзакція - підхід включає в себе відправку групи запитів на виконання до віддаленого прикладного процесу, що можуть об'єднувати декілька дій для виконання на віддалених ресурсах;

– розподілена транзакція - це взаємодія користувача із кількома серверами та процесами, які працюють на різних абонентських системах, для вирішення складних завдань.

Щоб здійснити розпаралелення опрацювання ВОД, прикладні програми розбиваються на частини, що в свою чергу розподіляються на різних системах в мережі. Даний процес сегментації автоматизується за допомогою спеціалізованого ПЗ. В наслідок сегментації кожна частина ПЗ містить алгоритми та інтерфейс для користувача та керування ВОД.

Передача даних для РОД може відбуватися за допомогою методів:

– віддаленого виклику процедур, який передбачає виклик процедур або функцій на серверах що характеризується високою швидкісттю, проте є дорогим у використанні через постійний обмін інформацією між системами;

– з використанням e-mail – при цьому методі дані передаються через електронну пошту, шо є менш витратним, проте більш повільним.

Обираючи метод передачі даних, користувач може балансувати між

швидкістю та вартістю розроділеного опрацюваня ВОД. На практиці розглядаються три основні моделі, які визначатимуть способи опрацювання та керування ВОД:

1. Опрацювання ВОД у однорангових локальних мережах. У цій моделі всі три рівні маніпулювання даними (зберігання, виконання додатків і представлення даних) зазвичай виконуються на одному робочому місці (рисунок 2.2). Основна інформаційна обробка відбувається на одному персональному комп'ютері. Ця модель підходить для невеликих завдань та індивідуальної роботи, але має обмежену масштабованість та можливості спільної роботи.

Рисунок 2.2 – Однорангова локальна мережа

2. Централізована обробка (рисунок 2.3). В цій моделі, зазвичай, централізований сервер або обчислювальний центр відповідає за зберігання та опрацювання даних, а користувачі підключаються до центрального ресурсу через термінали або клієнтські додатки. Цю модель використовують для великих організацій, але вимагає потужних ОР та надійної мережної інфраструктури.

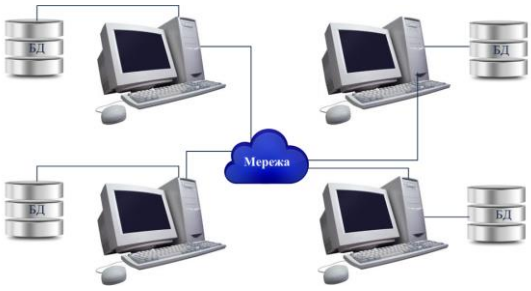


Рисунок 2.3 - Централізована обробка

3. Обробка даних в клієнт/серверній моделі (рисунок 2.4) розділяється ткаим чином: клієнти відправляють запити на сервери для отримання даних

та виконання обробки, після чого результати повертаються клієнтам. Ця модель надає більшу гнучкість та розподілену обробку між вузлами мережі.

Вибір конкретної моделі опрацювання даних залежатиме від потреб, масштабу завдань та доступних ОР проте у всіх цих моделях існують три основні рівні маніпулювання даними:

– зберігання;
– виконання додатків;
– представлення даних.

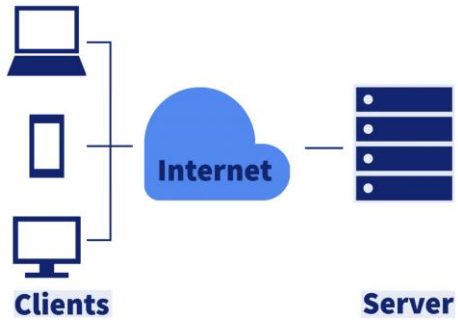


Рисунок 2.4 - Модель клієнт / сервер

Централізована обробка та модель клієнт/сервер часто використовуються в сучасних РСОД, оскільки дозволяють краще розподіляти завдання та забезпечувати ефективну роботу в умовах мережі. Насправді, вони є 2 різними підходами до організації роботи в РС і обидва мають свої плюси та обмеження. Ефективність та надійність централізованої обробка є основною перевагою а також те, що центральний сервер є потужнішим та надійним, оскільки весь обчислювальний та збережувальний обсяг об'єднується на одній платформі. Це може спрощувати управління та забезпечувати однакову якість обслуговування для всіх користувачів. Модель клієнт/сервер надає користувачам більшу гнучкість щодо вибору платформи та ОС для робочих місць, оскільки вони бувають незалежними комп'ютерами. Це дає можливість легше адаптувати робочі місця до потреб конкретних користувачів. Однак дана модель буває більш складною для управління, оскільки є більше компонентів та мережевий обмін а, це може вимагати більших зусиль у аспектах безпечності та надійності.

Звісно, обрана модель в більшості випадків залежатиме від специфічних завдань і можливостей доступних технологій. У сучасних системах часто використовують гібридні підходи, які поєднують переваги обох моделей. Розробники стараються забезпечити ефективність та гнучкість, дозволяючи користувачам вибирати підходи, що найкраще відповідають їх потребам.

2.2 Паралельне опрацювання даних у кластер-системах

Кластерна система - це група позв'язаних між собою ОР (нод), які працюють як єдине обчислювальне РС [22]. У таких системах ПО ВОД відбувається шляхом розподілу завдань між різними ОР. Основні характеристики включають наступне:

– система розподіляє обчислювальні завдання між вузлами, тобто кожним можуть виконуватися окремі частини задачі, що дозволяє скоротити обсяг завдань та підвищити продуктивність;

– вузли в кластері повинні взаємодіяти, щоб обмінюватися даними та координувати обчислення використовуються спеціалізовані протоколи та ПЗ для обробки спільних завдань.

– у деяких кластерах реалізується спільний доступ до обчислювальних та зберігальних ресурсів, що дозволяє збільшити доступну потужність та збереження даних в системі;

– кластери часто побудовані з урахуванням високої доступності, тобто якщо один вузол вийде з ладу, інші можуть продовжити роботу, забезпечуючи високу надійність системи.

– кластерні системи можуть легко розширюватися, додаванням нових ОР, якщо потрібно більше ресурсів для обробки даних;

– можуть використовуватися різноманітні рівні збереження, включаючи локальні диски на вузлах, мережеві СД та об'єднані файлові системи.

За допомогою цих характеристик кластери можуть оптимально опрацбовувати ВОД та завдань, що робить їх популярними для вимогливих завдань до продуктивності, наприклад обчислення у РРЧ та інші сценарії великих розподілених систем [27]. Компоненти інфраструктри кластера об’єднані у єдине інтегроване обчислювальне середовище (рисунок 2.3). Система кластер-обчислень включає в себе розподілену мережу ОР для розв’язання задач, які поділяються на більше елементарних атомарних одиниць, що виконуються на різних ресурсах [28-30].

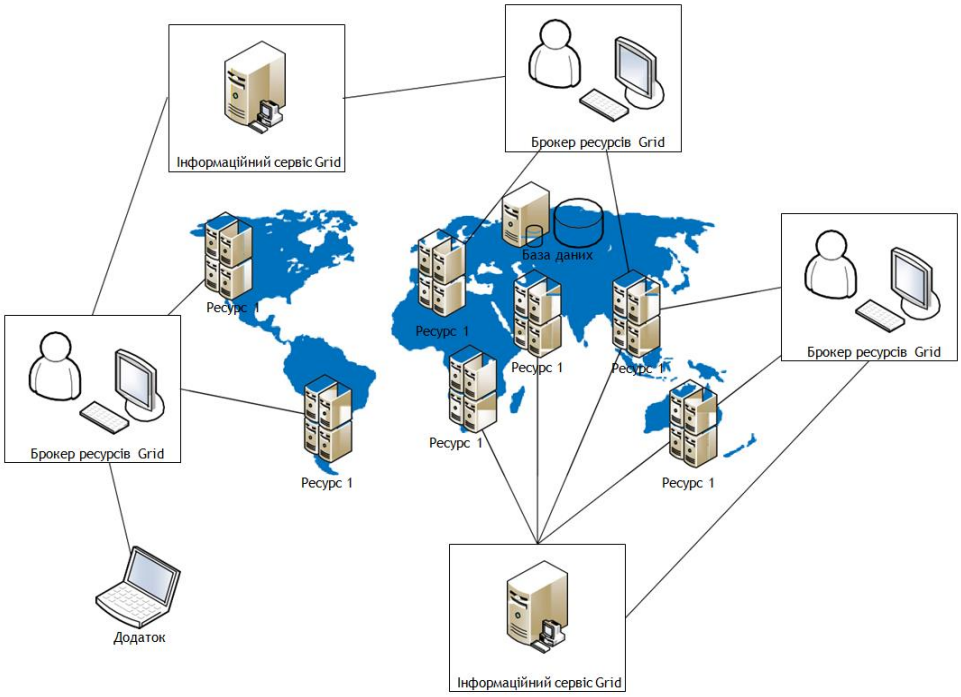


Рисунок 2.3 - Обчислювальне середовище кластер

Кластери відкривають нові можливості для розпаралеленого опрацювання ВОД та вирішення масштабних завдань у різних областях. Важливими аспектами та перевагами кластер-обчислень є:

– кластери легко розширюються шляхом додавання нових вузлів, що дозволяє пристосувати систему до зростаючих потреб і об'ємів даних;

– використання високопродуктивних комп'ютерів та швидких мереж дозволяє обробляти ВОД та виконувати обчислення швидко і ефективно;

– більшість кластерів побудовані з урахуванням високої

доступності, якщо один вузол вийде з ладу, інші можуть продовжити роботу, забезпечуючи надійну роботу системи;

– можуть об'єднувати розподілені ресурси, зокрема суперкомп'ютери, зберігальні системи, спеціалізовані пристрої та обчислювальні потужності для вирішення різних завдань;

– кластер-обчислення використовуються в широкому спектрі галузей, включаючи науку, інженерію, медицину, фінанси, розвиток ПЗ та ін, та є важливим засобом для високопродуктивних обчислень, аналізу даних та розробки технологій;

– дозволяють будувати рішення на основі різних архітектур та платформ, що надає гнучкість для розробки та використання різноманітних додатків.

Загалом, кластер-обчислення відкривають широке коло можливостей для опрацювання та дослідження ВОД, дозволяючи вирішувати складні проблеми в сучасному інформаційному світі.

Основні проблеми цієї системи включають:

– можуть містити різнорідне обладнання та технології, що вимагає стандартизації та уніфікації інтерфейсів і протоколів для забезпечення сумісності та зручності управління різними ресурсами;

– кластери можуть розташовуватися в різних географічних областях та адміністративних доменах, а управління цими розподіленими ресурсами вимагає вирішення питань безпеки, надійності й збалансованого розподілу завдань;

– з ростом кількості ОР і завдань може виникнути деградація продуктивності, для усунення яких потрібно впроваджувати ефективні механізми розподілу;

– вони повинні адаптуватися до змін у навантаженні та доступності ОР, що включає автоматичне розподілення завдань, моніторинг та аварійне відновлення.

– брокер ОР відіграє важливу роль у виявленні, плануванні та

моніторингу ресурсів та завдань, тому ефективна система управління повинна бути здатною керувати цими процесами у реальному часі.

2.2.1 Сервіси Grid

Середовище Grid зазвичай використовується для вирішення великих обчислювальних завдань, наукових досліджень, обробки ВОД та інших завдань, які вимагають певних ОР та доступу до мережевих послуг [23]. Це програмні засоби та компоненти, які надають додаткам у кластер-системі деякі спеціалізовані послуги та можливості для поліпшення продуктивності, надійності та доступності. Вони забезпечують ефективне використання ОР кластеру та обслуговувати потреби додатків, які працюють в РС.

Основні функції сервісів Grid включають:

– розподіл завдання між ОР кластера для забезпечення рівномірного розподілу навантаження та оптимального їх використання;

– забезпечують здійснення моніторингу та керування ресурсами виявляти стан ресурсів та автоматично перерозподіляти завдання або відновлювати роботу на інших вузлах в разі відмови;

– надають механізми для реплікації та резервування даних, що визначає доступність всієї системи та відновлення в разі аварії.

– управління життєвим циклом завдань, включаючи створення, запуск, завершення та відстеження стану завдань;

– забезпечують можливість аутентифікації користувачів та контролю доступу до ресурсів;

– допомагають у керуванні та доступі до розподілених файлів та даних;

– забезпечують збиряння, аналіз й звітність щодо діяльності кластеру та додатків.

Зазвичай Grid сервіси створюються для конкретних систем кластер-обчислень та додатків залежно від вимог конкретної системи. Вони буваютьи відкритими або комерційними, та включають в себе такі компоненти, як

балансувальники навантаження, системи розподіленого зберігання, системи резервних копій та ін.

Grid сервіси можуть надавати різноманітні види послуг для кінцевих користувачів, зокрема сервіси, які надаються в рамках кластер-систем:

– сервіси обчислень - дозволяють виконувати великомасштабні та обчислювально завдання на розподілених ОР, що використовуються для обчислень із значною кількістю задач або доступом до ВОД.

– сервіси опрацювання та передавання - гарантують безпечне збереження, каталогізацію та керування доступом до наборів даних, які розподілені, що особливо це важливо для організацій, які потребують масштабованого та безпечного зберігання. Можуть інтегруватися з обчислювальними сервісами і в такому випадку систему часто називають Grid -система даних.

Ці два типи сервісів представляють собою основні компоненти для багатьох застосувань у сферах обчислень, даних та наукових досліджень, де необхідна велика обчислювальна потужність та ефективне управління даними. Grid системи дозволяють організаціям масштабувати свої обчислення та даними відповідно до власних потреб.

2.2.2 Архітектура системи Grid

На рисунку 2.5 зображено архітектуру Grid системи, що є досить повною та відповідає опису ряду моделей, які зазвичай використовуються для розпаралелення ресурсів та управління ними. Суб'єктами системи Grid, що мають власні стратегії, є [21]:

– постачальники ресурсів - максимізація прибутку та використання своїх ресурсів

– споживачі - вирішення прикладних завдань в залежності від часових обмежень та бюджетних обмежень.

Споживачі можуть вибирати, коли і скільки грошей витратити на вирішення конкретної задачі, тому стратегія зазвичай базується на ефективному використанні ОР у межах виділеного бюджету. Постачальники

намагаються забезпечити найкраще використання ОР, на які було виділено кошти, що включає надання послуг іншим користувачам за плату або забезпечення передачі власних ОР для вирішення завдань споживачів.

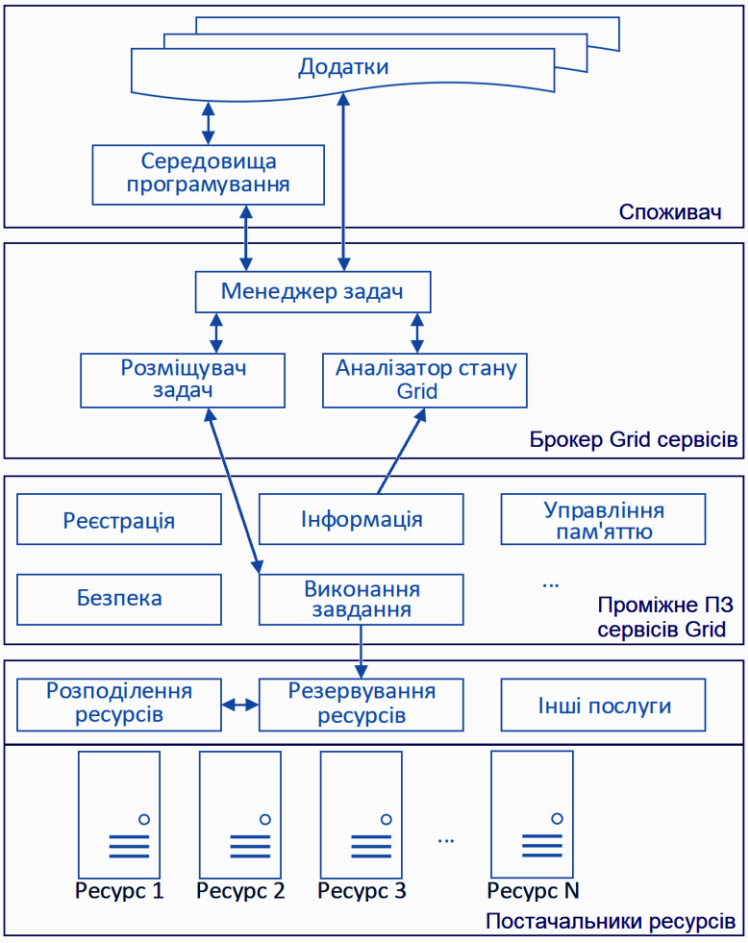


Рисунок 2.5 - Архітектура системи Grid

Системи включають різноманітні компоненти та шари, що спільно працюють для надання послуг користувачам, основні з яких:

– користувачі - комунікують з Grid системою, надсилаючи завдання на обчислення та опрацювання даних, при чому додатки, які вони при йьому використовують різняться від дослідницьких та інженерних до бізнес-орієнтованих.

– інтерфейс рівня користувача - надає послуги та інструменти для взаємодії з Grid системою зручним та інтуїтивним способом;

– програмне оточення - середовище, де виконуються завдання та додатки користувачів, яке надає ресурси та інфраструктуру для виконання обчислень;

– брокери Grid сервісів, що відповідають за виявлення ресурсів у Grid системі, моніторинг та планування виконання завдань (ПВЗ) на розподілених ресурсах, а також реалізують оптимізацію використання ОР.

– інтерфейс ядра - рівень, який відповідає за розподілення ресурсів у кластері та об'єднання їх у єдину систему, забезпечуючи комунікацію різних частин системи та координацію виконання завдань.

– постачальники - надають ресурси: обчислювальні, зберігальні та ін., необхідні для функціонування Grid системи, можуть розташовуватися та належати різним організаціям.

Всі компоненти працюють разом для надання користувачам розподілених обчислювальних та даних послуг у масштабах, що є недосяжними з використаням ізольованих ресурсів.

Користувачам Grid системи передбачено можливість вибирати постачальників ресурсів, які найбільше задовільняють вимоги чи потреби. Постачальники намагаються спершу обслуговувати локальних користувачів, надаючи пріоритет їх завданням. Системи Grid надають механізми їх взаємодії між собою.

Брокер є посередником між користувачами та постачальниками ресурсів кластера, сполучаючи їх за допомогою інтерфейсного програмного забезпечення ядра для надання послуг. Він забезпечує пошук ресурсів, призначення програм та даних до обраних ресурсів, ініціацію обчислень, а також адаптацію до змін. Брокер для користувачів Grid є єдиним ресурсом проте включє такі системи:

– управління завданнями - обробка завдань користувачів, взаємодія з планувальником для створення розкладу, публікація завдань для менеджера ресурсів, підтримка статусу завдань, комунікаувя клієнтів та користувачів, планування та розподіл завдань;

– складання розкладів - реалізує пошук ресурсів з використаням системи дослідження Grid, вибір ОР та розподіл завдань для забезпечення виконання завдань користувачів;

– дослідження кластера - виявляє ОР, взаємодіє з інформаційним сервером;

– встановлення завдань - активує виконання завдань на обраних ресурсах та періодично оновлює статус виконання завдань.

Базовий рівень Grid складається з ОР, наприклад комп'ютер, мережевий пристрий, пристрій для зберігання, інші спеціалізовані інструменти, які розподілені та становлять її основний фізичний склад. Основними його характеристиками є різноманітність архітектур і ОС, їх різні функціональні можливості і гетерогенність. Він є фізичним фундаментом, на якому будується вся інфраструктура Grid системи для того, щоб забезпечити можливість зручності й ефективності використання ОР на вищих рівнях системи використовуються різноманітні сервіси для користувачів та додатків.

2.3 Обгрунтуання вибору алгоритму розпаралелення задач

Аналіз підсистем ПВЗ в РСОД є важливим етапом для забезпечення ефективності та розподілення ресурсів. Підсистема ПВЗ відповідає за призначення завдань на ОР, розподіл робочого навантаженя і оптимізацію використання ОР в системі. Важливі аспекти, які можна розглянути при аналізі цієї підсистеми, включають [20-25]:

– Алгоритм планування включає в себе вибір і розробку алгоритмів, які визначають, які завдання будуть виконуватися на яких ресурсах та в якому порядку. ПВЗ може бути різним для різних видів завдань і залежати від певних аспектів, наприклад пріоритет завдань, їх важкість та час виконання. ПВЗ відповідає за вибір завдань для виконання на конкретних ОР, включати вибір оптимальних ОР для кожного завдання на основі його характеристик і вимог.

– Розподіл ОР - визначає, як їх розподілити між користувачами та завданнями справедливо і оптимально для досягнення високого рівня продуктивності системи.

– Моніторинг і керування виконанням завдань, щоб надавати можливість користувачам та системним адміністраторам доступ до даних стосовно статусу завдань і можливість скасування або перерозподілу завдань у разі потреби.

– Планування відповідно до приоритетів, а також дотриманняя правил обслуговування черг, щоб забезпечити виконання першочергово важливих завдань.

– Врахування оптимізації ресурсів, включаючи розподіл завдань між різними ОР для максимізації продуктивності та зниження завантаження.

– Подолання відмов і адаптація до змін в системі, таких як додавання нових ресурсів чи видалення старих.

– Система ПВЗ повинна бути шкальованою і спроможною працювати з великою кількістю завдань і ресурсів.

Проведення аналізу ПВЗ допомагає визначити, наскільки ефективно система розподіляє завдання і ресурси, і виявити можливість для покращення. Аналіз розподілу задач може виконуватися з різних підходів, в залежності від завдань і обставин, проте загальними принципами є теоретичний аналіз, зокрема оцінка часу виконання, алгоритмів розподілення, визначення їх теоретичної ефективності та складності, витрат ОР та друхих параметрів на основі математичних моделей. Експериментальний аналіз включає проведення практичних експериментів на реальних чи симульованих даних. Шляхом виконання багатьох запусків та вимірювань можна отримати дані для оцінки ефективності алгоритмів. Кількісний аналіз полягає в зборі та аналізі кількісних даних про ефективність алгоритмів. Вимірювання можуть включати в себе час виконання, використання ОР, обсяг передачі даних, завантаження системи та ін. Порівнювати різні алгоритми розподілення задач можна за різними критеріями, такими як час виконання, ресурси, надійність, масштабованість тощо. Цей підхід допомагає визначити, який алгоритм краще відповідає конкретним потребам. Моделювання дозволяє створити віртуальну копію

системи та використовувати різні алгоритми розподілення задач у віртуальному середовищі для аналізу їх ефективності. Симуляція дозволяє вивчити вплив різних факторів на роботу алгоритмів.

Після аналізу важливо провести оцінку їх ефективності та валідацію результатів експериментів, зокрема їх перевірку на відповідність теоретичним моделям та порівняння з очікуваними показниками. Також можливе визначення адаптивності та реакції на зміни та проведення аналізу при підвищеному навантаженні, що допомагає визначити, як система працює в умовах великої кількості задач та інтенсивного використання ресурсів. Проведення аналізу алгоритмів допомагає визначити, які алгоритми найбільше відповідають конкретним потребам та обставинам в РСОД.

До основних типів алгоритмів розподілення навантаження можна віднести:

– статичні - вирішують питання розподілення навантаження до початку і не змінюють рішень протягом роботи системи;

– динамічними використовується інформація щодо стану системи для розподілення навантаження у РРЧ, вони включають політику балансування, що вимагає значних затрат порівняно зі статичними;

– адаптивні поєднують в собі риси динамічних але можуть адаптуватися до змінних умов та вимог та обирати аолітику балансування, наприклад витісняючу або невитісняючу.

Типи динамічних алгоритмів в залежності від ступеня централізації:

– централізовані - можуть бути менш надійними через можливість збоїв центрального компонента;

– ієрархічні - комбінують в собі централізовані та розподілені елементи;

– повністю розподілені - зазвичай найбільш надійні та менше піддаються збоям центрального компонента;

– комбіновані.

Стабільність алгоритму балансування може бути системною або

алгоритмічною та вимагає недопущення перевантаження окремих вузлів та виконання непотрібних дій.

Елементами динамічних алгоритмів, як правило, є

- політика балансування забезпечує визначення того, чи являється вузол як об’єкт балансування;

– вибір партнера визначає, з ким можна взаємодіяти для балансування навантаження;

– вибір задачі визначає, яку саме із задач передавати та як це робити.

– збиряння інформації про стан РСОД допомагає алгоритму приймати рішення, включає різі методи, як збір даних по необхідності, з певною періодичністю та вибірка даних про стан РСОД загалом.

2.3.1 Розподіл ресурсів

Розподіл ресурсів - це процес надання доступу та використання різних ресурсів, таких як обчислювальні потужності, пам'ять, ресурси мережі, СД тощо, для задач і процесів в інформаційній системі або мережі. Розподіл ОР може бути необхідним для керування навантаженням системи, забезпечення справедливого доступу до ресурсів різним користувачам та задачам, а також для оптимізації використання інфраструктури.

Основні аспекти розподілу ресурсів включають:

– Планування ресурсів передбачає вибір, який ресурс буде наданий якій задачі та на який термін. ПВЗ визначає, які задачі отримають доступ до ресурсів і в якому порядку.

– Для забезпечення безпеки ресурсів використовуються механізми контролю доступу, такі як правила доступу, ролі та автентифікація користувачів.

– Іноді ресурси потрібно розділяти на фізичному або логічному рівнях між різними сутностями, наприкалд як віртуальними машинамии чи контейнерами.

– Для ефективного розподілу ОР відбувається моніторинг їх використання, а також реакції на зміни у навантаженні або стані системи.

– Великими системами використовується автоматизоване управління ОР, щоб оптимізувати їхнє використання без значного втручання вручну.

– Розподіл ОР включає аналіз продуктивності та виявлення проблем для покращення використання ресурсів.

2.3.2 Процес планування задач

У процесі ПВЗ [5] чи процесів зазвичай виділяється наступні три етапи:

– пошук та фільтрація ресурсів;

– відповідно до конкретних цілей відбір ресурсів і планування;

– запуск процесу.

На рисунку 2.6 показано модель ПВЗ, функціональні блоки якої поєднано різними типами потоків даних, зокрема інформації про ОР - штрих та задачі - суцільна лінія.

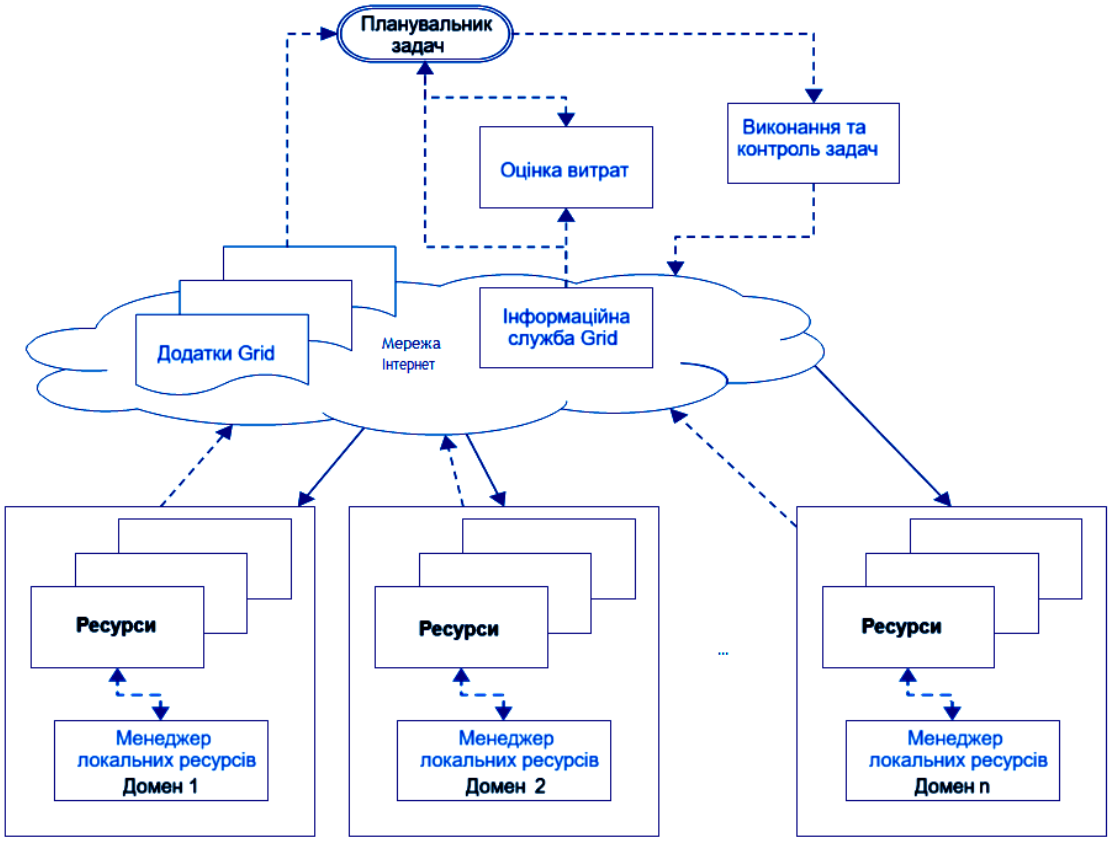


Рисунок 2.6 - Схема ПВЗ у РСОД

Процес ПВЗ у РСОД є критично важливим для ефективного використання ОР та оптимізації обчислень. Він складається з етапів:

– збір вимог користувача та визначення характеристик задачі - включає обсяг обчислень, час виконання, пріоритет задачі, а також можливість розпаралелювання задачі на підзадачі;

– визначення які ОР, сервери, вузли кластера тощо, доступні для виконання цих задач;

– розподіл задач між доступними ОР, враховуючи вимоги, а також можливості і доступність наявних ресурсів. Процес включатиме вибір оптимальних ресурсів для задачі, встановлення пріоритетів та забезпечення оптимального використання ресурсів.

– розробка плану виконання, що включає послідовність задач і час їх виконання, з використанням алгоритмів планування для оптимізації.

– моніторинг виконання задач, збір даних щодо їх стану та стан ОР, можливість втручання для управління процесом виконання.

– оптимізація процесу, зокрема, зміна пріоритетів, розподіл ресурсів або перерозподіл завдань для підвищення продуктивності або врахування негайних потреб користувача.

– завершення і оцінка результатів щодо їх відповідності вимогам, і забезпечення очищення ОР;

– на етапі звітності генеруються звіти про виконані задачі, витрати ОР, часові затрати та інших параметрів для аналізу та подальшого вдосконалення ПВЗ.

Процес ПВЗ є динамічним і вимагає постійного моніторингу та оптимізації з метою забезпечення ефективності використання ОР та задоволення вимог користувачів. Він враховує важливість інформаційної підтримки і забезпечення найбільшого рівня ефективності.

2.3.3 Алгоритм найменшого часу виконання

«Алгоритм найменшого часу виконання (Earliest Finish Time або EFT)

один з алгоритмів в області ПВЗ дач в РСОД, що має на меті визначити, коли кожна задача може виконуватися, так, щоб загальний час закінчення виконання був мінімізований (рисунок 2.7)» [7]. Використання EFT дозволяє оптимізувати тривалість виконання задач у контексті РСОД або інших розподілених обчислювальних середовищ.

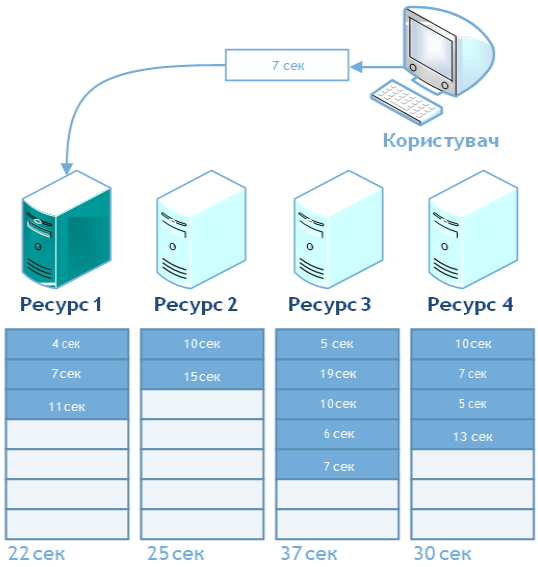


Рисунок 2.7 - Загальна схема алгоритму

Основні кроки алгоритму EFT:

1. Визначення завдань та їх характеристик. Задачі повинні бути описані разом зі своєю обчислювальною складністю (наприклад, часом виконання, обсягом пам'яті та ін.). Кожна задача пов'язується зі списком ОР, на яких її можна виконувати.

2. Визначення ресурсів, які доступні для виконання задач. ОР описуються власною інформацією, такою як процесор, пам'ятіь та ін обчислювальні можливості.

3. Розрахунок часу на виконання кожної із задач на кожному доступному ОР, обчислення часу запуску та завершення.

4. Вибір найкращого варіанту (ОР та тривалість виконання), що мінімізує час опрацювання всіх задач. Для цього EFT вибирає задачу, що може виконуватися якнайшвидше, та ОР, на якому вона буде виконана.

5. Оновлення часових параметрів після вибору варіанту для інших

задач та ОР на основі нового стану.

6. Кроки 4 і 5 повторюватимуться, поки всі задачі не будуть розподілені.

7. Завершення процесу після розподілу всіх задач і виборі ОР та тривалості виконання. Задачі можуть бути відправлені на виконання на визначені ОР.

– Алгоритм (рисунок 2.7) орієнтований на мінімізацію часу опрацювання всіх задач, тому він використовується в ситуаціях, де важливо досягнути якнайшвидшого завершення. Однак важливо враховувати, що він не завжди гарантує оптимальний розподіл ОР, і в певних випадках є менш ефективним порівняно з іншими. Цей алгоритм спирається на прогнозування тривалості виконання конкретної задачі на кожному доступному ОР і обирає той, який має найкоротший очікуваний час завершення для задачі. Тому він потребує великої кількості доступної і актуальної інформації щодо стану ОР та завдань. Недоліком є значні витрати ресурсів для обчислення приблизної тривалості виконання; велика обчислювальна складність, особливо в РСОД із значною кількістю задач та ресурсів, погіршення загальної ефективності через велику кількість обчислень та комунікаційних операцій, складність управління параметрами, тобто визначення точних параметрів для обчислення приблизного часу на виконання і навчання моделей для прогнозування.

Переваги: мінімізація часу завершення сіх задач, використання актуальних даних про стан ОР і завдань, що може допомогти зробити більш об'єктивний вибір.

2.3.4 Алгоритм найменшої черги

«Алгоритм мінімальної черги (Shortest Queue, SQ) - є одним з класичних у багатозадачних ОС та системах в РРЧ (рисунок 2.8)» [8]. Основна ідея, щоб усі задачі обслуговуються у такому ж порядку, в якому надійшли, і кожна задача виконується протягом фіксованого кванту часу, після чого управління передається наступній задачі в черзі на вільний ОР.

Якщо задача ще не завершена після витрати свого кванту часу, вона повертається до кінця черги для подальшого виконання.

Основні характеристики даного алгоритму

– усі задачі отримують рівні шанси на виконання, і жодна задача не може захопити процесор назавжди;

– відносно легкий у реалізації, оскільки вимагає лише базової логіки для керування чергою задач;

– може використовуватися в РСОД в РРЯ, оскільки гарантує обслуговування всіх задач за обмежений час.

правильного вибору кванту часу, цей алгоритм може мінімізувати простій процесора та максимізувати його завантаженість.

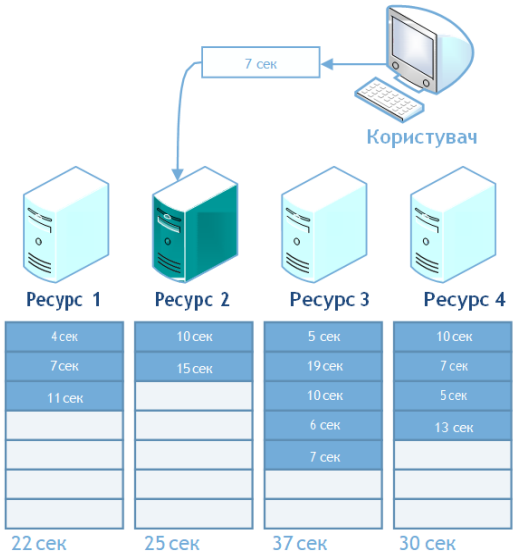


Рисунок 2.8 – Схема алгоритму найменшої черги

До недоліків відноситься:

– неефективність для задач зі змінними обчислювальними потребами, якщо деякі задачі вимагають більше часу для завершення, а інші менше;

– визначення оптимального кванту часу буває досить складним, вибір занадто великого кванту призводить до затримок в реакції на системні події;

– великих системах із багатьма задачами можуть виникнути

ситуації, коли не забезпечується найкращий час завершення для деяких задач.

На практиці, алгоритм зазвичай використовується спільно з іншими щоб збалансувати його недоліки. Алгоритм SQ є одним з алгоритмів ПВЗ у системах з багатьма чергами (Multi-Queue Systems) де нові задачі направляються до тої черги, що має найменшу кількість очікуючих задач.

Основні кроки алгоритму найменшої черги:

1. Кожна черга в системі веде облік кількості задач, які очікують у ній;

2. Коли нова задача надходить в систему, вона спочатку розглядає черги, щоб знайти ту, у якій менше всього очікуючих задач;

3. Нова задача направляється до цієї черги для виконання;

Цей алгоритм дозволяє балансувати навантаження між чергами, оскільки завдання будуть спрямовані до черги з найменшою кількістю задач. Однак, він не враховує тривалість виконання конкретної задачі, пріоритети або специфічні вимоги завдань, що може привести до того, що задачі у чергах з більшою кількістю задач можуть чекати довше, що може вплинути на загальну продуктивність системи.

2.3.5 Алгоритм кластеризації

Алгоритм кластеризації методом К-середніх (K-means) - це популярний алгоритм кластеризації даних, який використовується для групування схожих об'єктів в одній категорії або кластері (рисунок 2.9) [5]. Цей алгоритм відомий своєю простотою та ефективністю, і він широко використовується в машинному навчанні, аналізі даних і різних додатках, де необхідно автоматично розділити дані на групи.

Основними кроками роботи алгоритму є:

1. Вибирається кількість кластерів (K), яку ви хочете створити у ваших даних.

2. Випадковим чином ініціалізується K центрів кластерів. Кожен центр представляє потенційний центр кластера.

3. Кожен об'єкт даних призначається до найближчого за евклідовою

відстанню центра кластера.

4. Розраховується нове положення центрів кластерів, як середнє арифметичне об'єктів, які були призначені до кожного кластера.

5. Крок 3 і 4 повторюються ітеративно до збіжності, коли об'єкти не змінюють свого призначення до кластерів.

6. Завершивши ітерації, ви отримуєте K кластерів, і кожен об'єкт належить до одного з них.Основні переваги алгоритму це простота реалізації і розуміння, висока швидкість для багатьох даних. Він працює для багатьох типів даних, якщо кількість кластерів правильно вибрана.

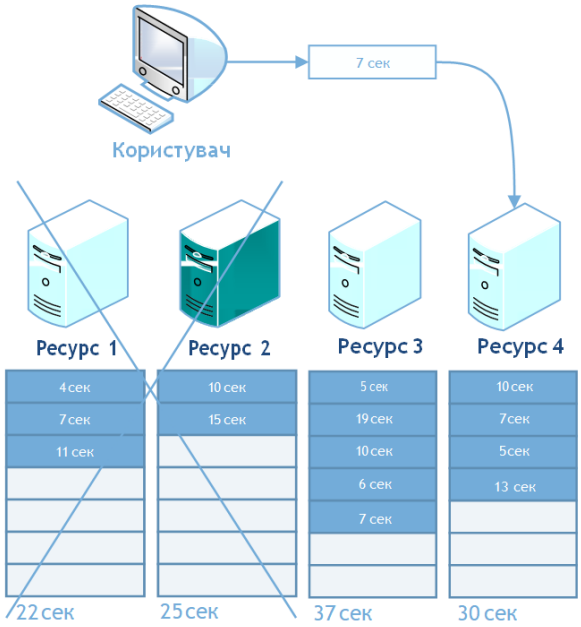


Рисунок 2.9 - Алгоритм на основі методу K-середніх

Недоліки:

– не завжди знаходить глобальний оптимум, оскільки залежить від випадкового вибору початкових центрів кластерів;

– кількість кластерів потрібно задати заздалегідь, і неправильний вибір може призвести до некоректного розділення даних;

– наявність викидів в даних може вплинути на результати.

У разі, якщо ви плануєте використовувати алгоритм K-means у ваших дослідженнях або застосуваннях, важливо правильно підготувати дані та врахувати особливості алгоритму для досягнення кращих результатів.

Алгоритм К-середніх (K-means) в контексті системи ПВЗ виявляється у виборі підмножини ОР для обчислення очікуваної тривалості виконання завдання, а не розрахунку на всіх ОР, що допомогає зменшити витрати на обчислення, що особливо важливо у великих кластер-системах. Проте, визначення кількості ОР, які слід використовувати щоб оцінити тривалість виконання задачі, є важливим аспектом методу. Вибір кількості ОР (позначеного як K) може вплинути на продуктивність системи. Якщо K вибрано надто малим, це може призвести до недостатньої обробки завдань та низької використаності ресурсів. Якщо K велике, це може призвести до зайвого навантаження системи та зайвих обчислень. Тому важливо підійти до цього вибору обережно і можливо застосовувати попередні стохастичні експерименти, які допоможуть підібрати оптимальне значення K для конкретної системи. Шляхів підбору K кілька:

– Метод "ліктя" (Elbow Method): В цьому методі ви обчислюєте внутрішню дисперсію для різних значень K та спостерігаєте, де внутрішня дисперсія зазнає різкого зменшення. Точка, в якій спостерігається "ліктьова" крива, вказує на оптимальне значення K.

– Силуетний метод (Silhouette Method): Цей метод використовує міру силуету для оцінки якості кластеризації для різних значень K. Оптимальне K відповідає найвищому середньому значенню силуету.

– Стохастичні методи: Ви можете провести стохастичні експерименти, де декілька разів рандомізовано вибираєте значення K та оцінюєте вплив на продуктивність системи.

2.3.6 Визначення алгоритму планування задач

Перевірка та оцінка роботи алгоритмів для ПВЗ в РСОД - важливий етап у розробці та вдосконаленні кластерних рішень. Існує кілька підходів та методів, які використовуються для цього:

– Симуляція роботи алгоритмів на модельних даних. При якому створюється модель РСОД та вводитяться тестові завдання. Спостереження за тим, як алгоритми планування реагують на ці завдання, може надати

розуміння про їх продуктивність та ефективність.

– Емпіричне тестування передбачає виконання на реальних або імітованих завданнях та порівняння їх результатів з показниками продуктивності. Виконується аналіз метрик, наприклад час виконання завдань, використання ресурсів, середні часи очікування та інші важливі характеристики.

– Порівняльні експерименти кількох різних алгоритмів, запустивши їх на однакових наборах завдань дозволяють порівнювати їх продуктивність з різних поглядів, включаючи час виконання, використання ОР, рівномірність завдань тощо.

– Стрес-тестування - використовється навантаження, що перевищує максимальну спроможність системи, щоб перевірити, як алгоритми ведуть себе в умовах перевантаження допомагає виявити їх слабкі сторони та прогнозувати працездатність у РРЧ.

– Аналіз та моделювання використовують математичні моделі та аналіз, наприклад, аналіз складності або математичне моделювання завдань та ОР допомогають передбачити роботу алгоритмів у різних умовах.

– Використання реальних даних про роботу кластер-системи для тестування алгоритмів планування.

На прктиці корисним є використання комбінації цих методів. Після виконання тестування можена зробити висновки щодо ефективності алгоритмів та вибрати найбільш відповідний умовам.

Для порівняння продуктивності алгоритмів ПВЗ у розробленому емуляторі РСОД встановлено параметри, що наведені в таблиці 2.1.

Емуляція роботи РСОД була виконана на робочій станції з 64-розрядною ОС Windows 7, процесор - Intel Core i7-4770, тактова частота - 3.4 ГГц, оперативна пам'ять - 16 ГБ. Для об'єктивності результатів проведені 10 експериментів, що незалежали один від одного, і середнє арифметичне було використано для обчислення результатів (таблиця 2.2 і 2.3) роботи алгоритмів ПВЗ з різним балансуванням навантаження (рисунок 2.10 і 2.11).

Таблиця 2.1 – Параметри для порівняння продуктивності алгоритмів

Кількість ОР	10
Коефіцієнт продуктивності кожного ОР	від 0.8 до 1.2
Кількість задач	500
Середній час виконання однієї задачі	1 с ±200 мс
Час додавання задач у чергу	Кожні 100 мс
Час доступу до ОР	5мс з ±5мс
Час доступу до сховища даних	5мс з ±5мс

Спершу розглянуто, наскільки алгоритми проявляють свою ефективність в ідеальних умовах, коли відсутні затримки між ПВЗ і ОР, та між ОР і сховищем ВОД. Результати емуляції сценарію з миттєвим доступом, в одиницях вимірювання – мс, наведені в таблиці 2.2.

Таблиця 2.2 - Результати емуляції сценарію

Алгоритм	Загальний час	Середній час очікування	Час очікування (max)	Середній час простою	Час простою (max)
K-means	52260,3	645,6	1849,8	1580,3	2692,2
Min-час	52182,2	476	1575,5	1423,1	2508,2
Min-черга	52124,5	526,9	1774,2	1492,6	3773,2

З результатів (таблиця 2.2) видно, що в цьому ідеальному варіанті динамічні алгоритми є досить швидкими, оскільки вони миттєво визначають найоптимальніший ОР для виконання кожної із задач, і в цьому контексті навантаження на ОР збалансоване. Однак варто зауважити, що ці результати базуються на ідеальних умовах, і у реальних умовах затримки та інші фактори можуть впливати на продуктивність алгоритмів.

Оскільки алгоритми найменших черги та часу виконання вимагають опитування кожного доступного ОР при кожному запиті щоб визначити

найкращий варіант, це може спричинити зниження їх продуктивності у випадку великої розподіленості системи. Тому для отримання результатів (таблиця 2.3), більш адекватних реальним умовам, досліджено вплив затримок між ПВЗ і ОР, а також випадковою затримкою до 10 мс між ОР та СД, де знаходяться необхідні дані для виконання завдань.

Таблиця 2.3 - Результати тестування

Алгоритм	Загальний час	Середній час очікування	Час очікування (max)	Середній час простою	Час простою (max)
K-means	74890,50	77,20	952,30	21725,00	27460,40
Min-час	383187,33	645,00	1345,11	68043,89	264086,44
Min-черга	384269,00	647,33	1206,89	68446,56	266066,78

Як видно, балансування є менш продуктивним в умовах високої розподіленості системи через час, необхідний для обробки запитів до ОР та доступ до СД. Навіть при тому, що ці затримки становили в середньому лише 5 мс, вони призвели до суттєвого збільшення сумарної тривалості виконання 1 тисячі завдань паралельно на 10 ресурсах.

Іншим ключовим аспектом є тривалість очікування в черзі завдання для виконання, який рівний різниці часу, коли завдання було поставлено в чергу, і часом початку його виконання.

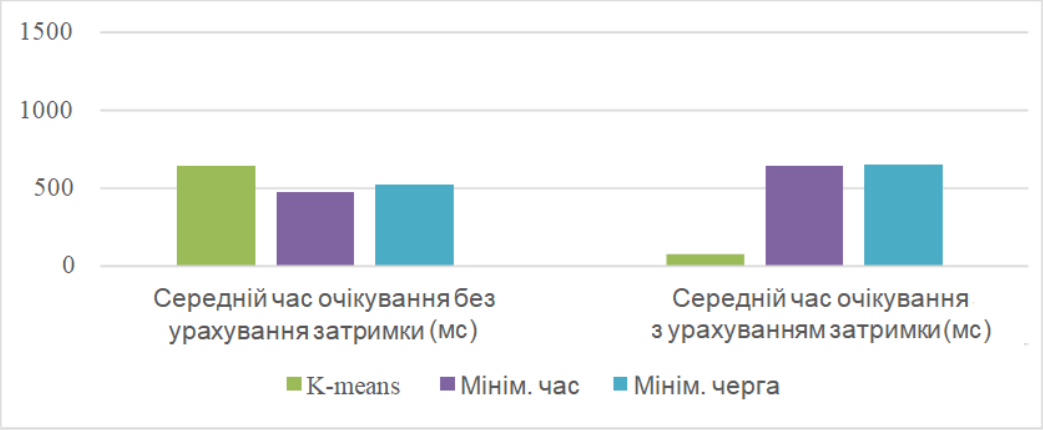


Рисунок 2.10 - Результати середньої тривалості очікування задач

Згідно з діаграмою на рисунку 2.10, видно, що всі алгоритми майже не зазнають змін у цьому показнику без врахування додаткового часу (затримок), необхідного для комунікації між ПВЗ ОР.

Показник середньої тривалості простою ОР (рисунок 2.11) відображає час, протягом якого ресурс був неактивний, тобто простоював, тим часом як інші ресурси виконували завдання. З рисунка 2.11 видно, що затримки суттєво збільшують цей час для динамічних алгоритмів. Це зумовлене тим, що затримка виникає двічі: при зверненні від ПВЗ до кожного з ОР для визначення їхнього статусу, а потім при завантаженні ВОД зі СД.

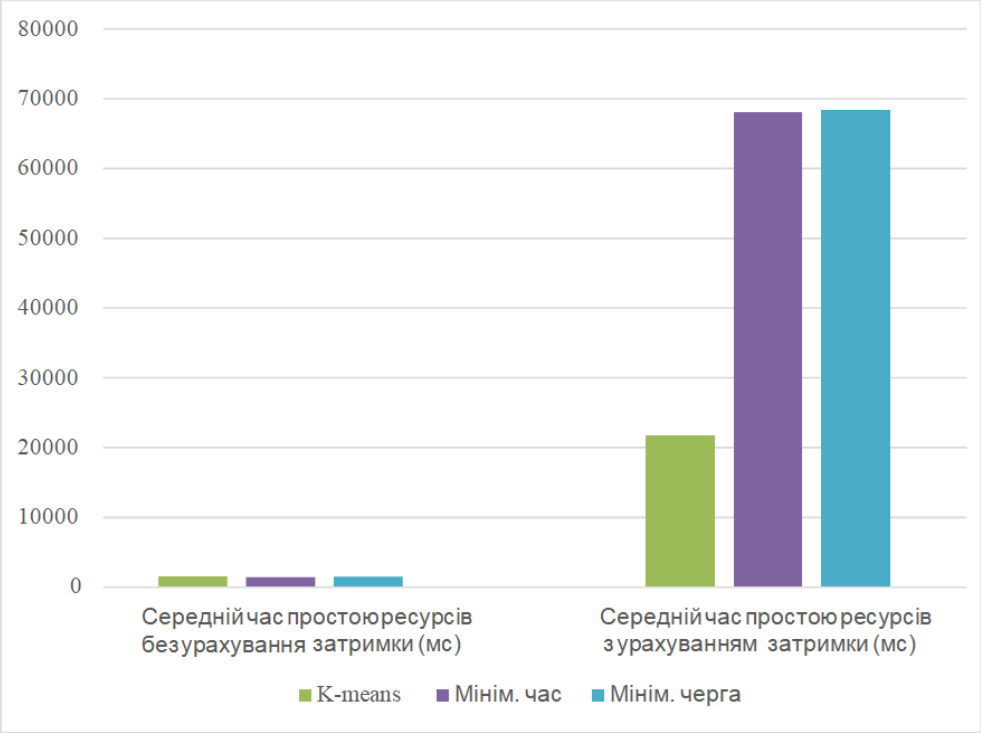


Рисунок 2.11 - Результати середньої тривалості простою ОР

Проведені дослідження показують, що в умовах обмеженої розподіленості ОР та сховищ ВОД, динамічні алгоритми, які враховують поточний стан кожного з ОР та допомагають максимізувати загальну тривалість виконання і середню тривалість очікування завдань у черзі, є ефективними на практиці. Тому, важливо адаптувати вибір алгоритму ПВЗ до конкретних умов та специфіки РСОД для досягнення оптимальності її характеристик.

2.4 Вимоги до проектованої системи

Комп'ютерно-інтегрована система (КІС) для підвищення ефективності розподіленої обробки ВОД повиння мати такі функціомальні можливості:

– управлінння ОР, включаючи сервери, кластери, СД, мережееве обладнання та ін., автоматично масштабувати та балансувати навантаження для забезпечення оптимального використання ОР;

– засоби для створення графіків та планів обробки ВОД, що допомагатиме оптимізувати часові рамки та ОР, необхідні для виконання конкретних завдань;

– можливість відстежування продуктивность ОР, аналізу ВОД для виявлення вузьких місць та надавати інструменти для усунення проблем та оптимізації продуктивності;

– засоби управління ВОД, включаючи реплікацію, бекапи, стиснення даних і розподілене зберігання, що допомагатиме забезпечити надійність та доступність ВОД;

– забезпечення безпеку ВОД та ОР шляхом управління правами доступу, шифруванням даних та моніторингом загроз безпеці;

– дозволяє автоматизувати повторювані завдання, такі як резервне копіювання даних, архівування та масштабування ОР, щознижує навантаження на адміністраторів та підвищує ефективність системи;

– інтеграція з іншими додатками та системами, що забезпечить сумісність та обмін ВОД між різними частинами інфраструктури;

– формування звітів та аналітики про продуктивність, використання ОР та інші аспекти функціонування системи, що допомагатиме у прийнятті рішень та покращенні роботи системи;

– засоби для оптимізації енергоспоживання, включаючи управління плануванням ресурсів і ефективне використання обладнання.

КІС для обробки ВОД допомагатиме покращити продуктивність, ефективність та безпеку своєї інфраструктури опрацювання даних.

3. ПРОЕКТУВАННЯ КОМП’ЮТЕРНО-ІНТЕГРОВАНОЇ СИСТЕМИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛЕНОЇ ОБРОБКИ ВЕЛИКИХ ОБ’ЄМІВ ДАНИХ

3.1 Розробка оптимізованого алгоритму планування завдань

Розглянуті методи ПВЗ характеризуються певними перевагами й обмеженнями, тому розробка оптимізованого алгоритму, який був би спроможний врахувати їх є актуальною задачею.

Запропонований алгоритм має бути здатним ефективно управляти гетерогенними та розподіленими ОР і СД. Це допоможе знизити загальну тривалість обробки ВОД і мінімізувати тривалість очікування завдань у черзі, що важливо для підвищення продуктивності та зменшення затрат. Оптимізація критерію тривалості простою ОР дозволить РСОД працювати максимально ефективно і знизити надмірні витрати. Даний алгоритм, комплексно враховуватиме усі аспекти та обмеження ПВЗ у РСОД для підвищення її ефективності та оптимізації використання ОР.

Оскільки реальні РСОД завжди є розподіленими, тому врахування затримок сигналів, якими обмінюються ОР, ПВЗ та СД є обовязковим. Статичними алгоритмами, такими як стохастичний та Round-Robin, не враховується поточний стан ОР, що забезпечує. максимальну швидкість додавання задач у чергу, але може призвести до неоптимального розподілення задач по ОР.

Один із способів виправити цей недолік - комбінувати статичну та динамічну складові алгоритмів. Замість того, щоб опитувати кожен ОР кожного разу під час розв’язання задачі балансування, алгоритм може кешувати стани ОР у підсистему ПВЗ. Ця інформація зберігатиметься в пам'ять й оновлюватиметься фоновим потоком з адаптивним періодом, залежно від навантаженості системи. Тобто, після запиту до ПВЗ, алгоритм буде враховувати стан кожного ОР з кешу, що дозволить покращити розподіл задач та зменшити тривалість очікування.

Поєднуючи статичні та динамічні компоненти, запропонований алгоритм буде спроможний оптимально розподіляти завдання, враховуючи поточний стан ОР. Алгоритм K-means продемонстрував гарні результати в тривалості виконання завдань завдяки обробці лише обмеженої кількості ОР, що спрощує час обробки загального масиву даних.

Отже, при розробці оптимізованого алгоритму передбачено додаткове налаштування пріоритету використання кожного ОР. Це можна зробити, сортуючи список ОР за їхнім поточним навантаженням. Під час планування, алгоритм буде працювати з першою частиною цього списку, який має вищий пріоритет. Це дозволить швидше вирішувати задачі. Вибір найкращого ОР для виконання поточного завдання проводитиметься шляхом порівняння загальної тривалості виконання задачі на різних ОР. При цьому слід враховувати відстань між ОР та СД. Таким чином, при однаковому часі виконання на двох ОР, варто віддати перевагу тому, який розташований ближче до СД, де зберігаються необхідні дані.

Гетерогенність ресурсів РСОД означає, що кожен ОР має відмінності в оперативній пам'яті, процесорній потужності та завантаженні файлової системи. При оновленні інформації про ресурс у кеші, потрібно розраховувати його поточну продуктивність та періодично оновлювати це значення. Цей коефіцієнт буде використовуватися для калібрування часу виконання завдань на ресурсах. У разі подібного часу виконання, ПВЗ повинне віддати перевагу потужнішому ресурсу, який впорається з задачею за менший час. Ці вказівки можуть взаємно уточнювати одна одну.

При розробці оптимізованого алгоритму ПВЗ враховано розглянуті умови та правила. Його відмінністю від існуючих підходів є те, що він аналізує кеш стану ОР, що забезбечує суттєве покращення загальної тривалості виконання задач. Блок-схема роботи алгоритму наведена на рисунку 3.1:

– На вхід поступають параметри: задача, перелік ОР, періодичність оновлення кеш.

– Оновлення переліку стану ОР відбувається тільки в разі

перевищення інтервалу часу з моменту останнього оновлення.

– Паралельно виконується ПВЗ за наступною схемою:

– зчитування з пам'яті кешу стану ОР;

– вибір ОР з найменшою передбачуваною тривалістю виконання задачі;

– кеш стану вибраного ресурсу оновлюється відповідно до тривалості виконання цієї задачі.

– Обраний ОР для виконання даної задачі передається ПВЗ для подальшого запуску задачі.

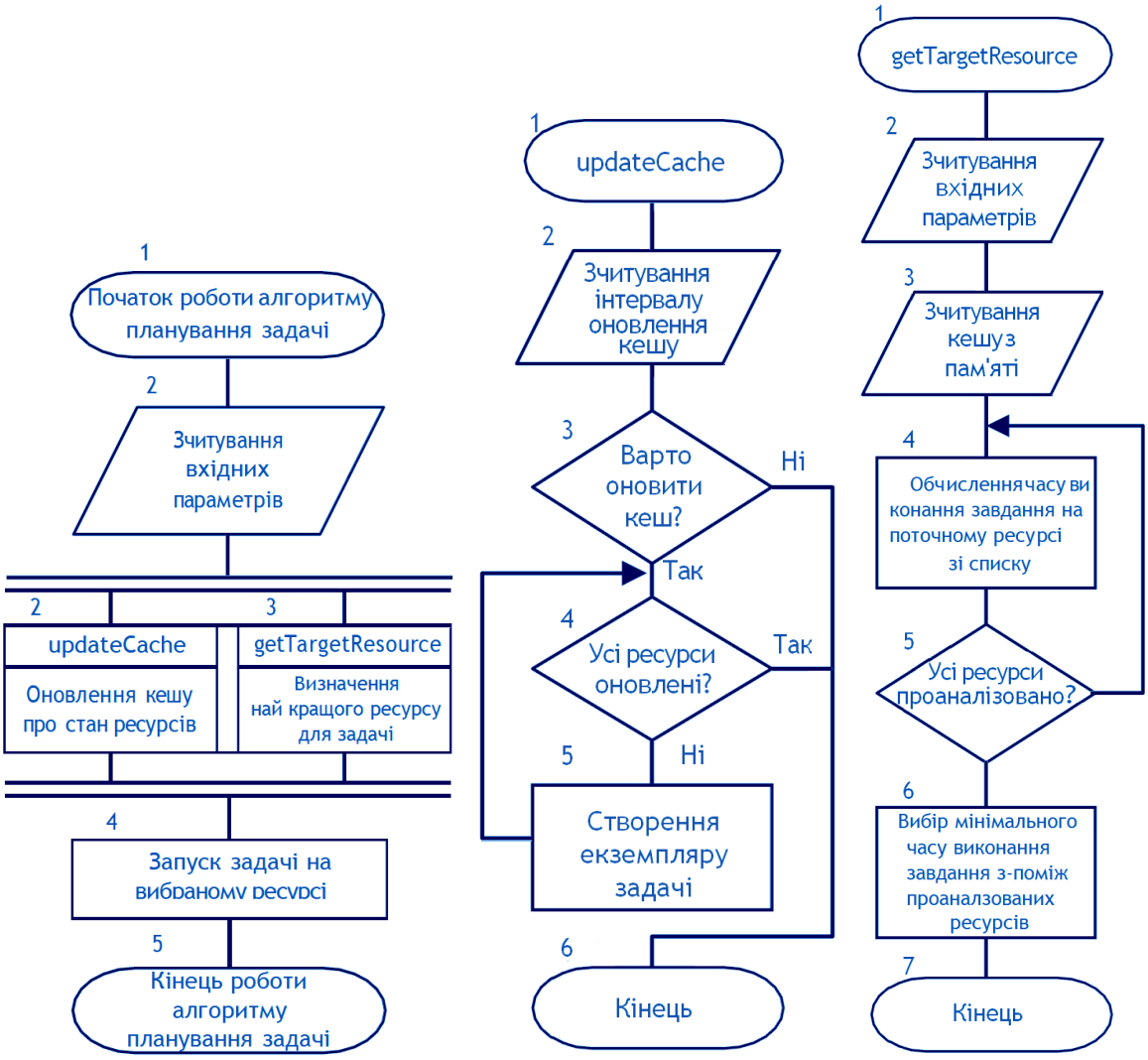


Рисунок 3.1 - Блок-схема оптимізованого алгоритму ПВЗ

Алгоритм враховує поточний стан ОР та їх властивості, а також віддаленість ОР один від одного, що робить його ефективним та підвищує продуктивність ПВЗ.

Оптимальні значення параметрів, які використовує запропонований алгоритм, визначаються в залежності від характеристик РСОД. Ефективність розподілу завдань залежить від темпу поступлення нових задач і ОР та показника оновлення кешу, який зберігає інформацію про ОР. Для його визначення, спираючись на основні параметри системи, проведно аналіз за умов, що наведені в таблиці 3.1.

Таблиця 3.1 – Параметри РСОД

Кількість обчислювальних ресурсів	10
Кількість задач	200
Коефіцієнт продуктивності кожного ресурсу	від 0.9 до 1.1
Середній час виконання однієї задачі	1 секунда ±200 мс
Період додавання задачі в чергу ресурсу	100 мс
Час доступу до обчислювальних ресурсів	5мс з ±5мс
Час доступу до сховища даних дорівнює	5мс з ±5мс

Отримані графіки демонструють графіки залежність тривалості оновлення кешу від параметру загальної тривалості виконання всіх задач (рисунок 3.2), середної тривалості очікування завдання у черзі (рисунок 3.3) та середньої тривалості простою ОР (рисунок 3.4).

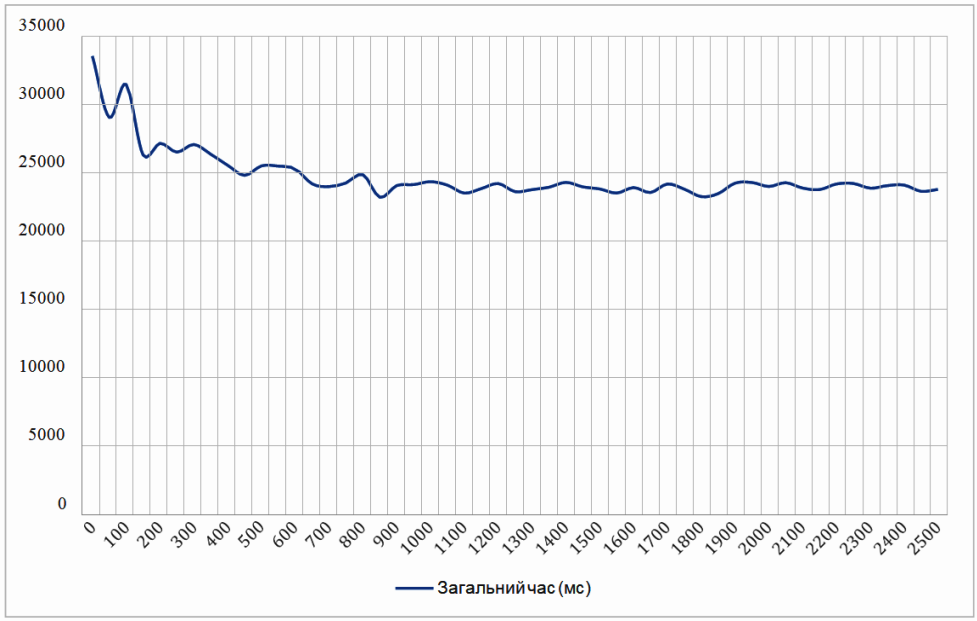


Рисунок 3.2 – Вплив загальної тривалості виконання задач

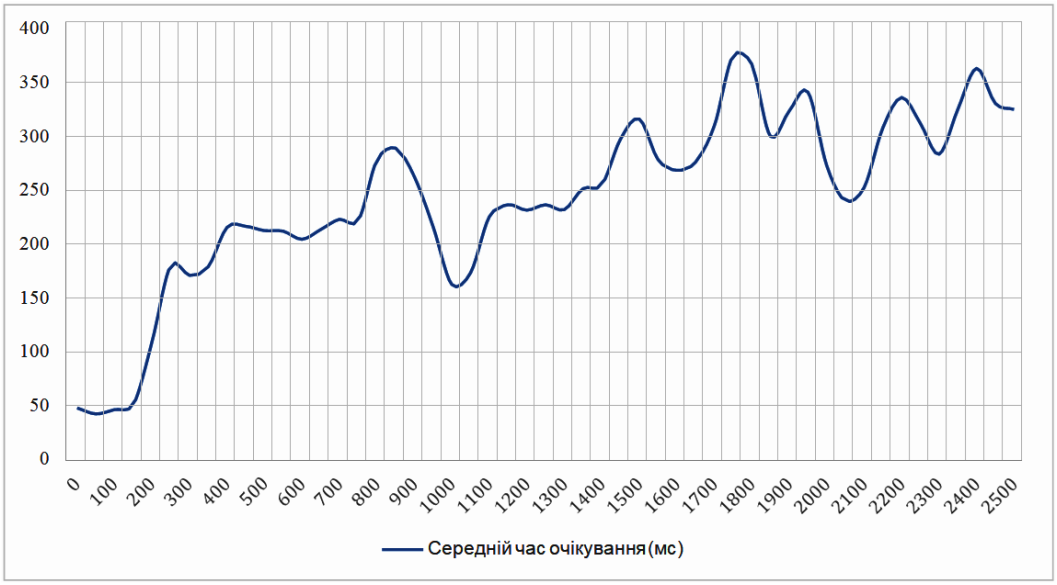


Рисунок 3.3 - Вплив середньої тривалості очікування задачі у черзі

Рисунок 3.4 - Вплив середньої тривалості простою ОР

Після визначення оптимальної моделі балансування навантаження в РСОД для порівняння результатів роботи запропонованого алгоритму з існуючими використано параметри РСОД, що наведені в таблиці 3.1.

Для об'єктивної оцінки ефективності КІС експерименти проводилися в умовах недовантаженості (рисунок 3.5) та перевантаженості (рисунок 3.6). Параметрами, які змінюються під час тестування є інтенсивність поступлення заавдань та періодичність кешування ПВЗ, відповідно з розрахованими оптимальними значеннями.

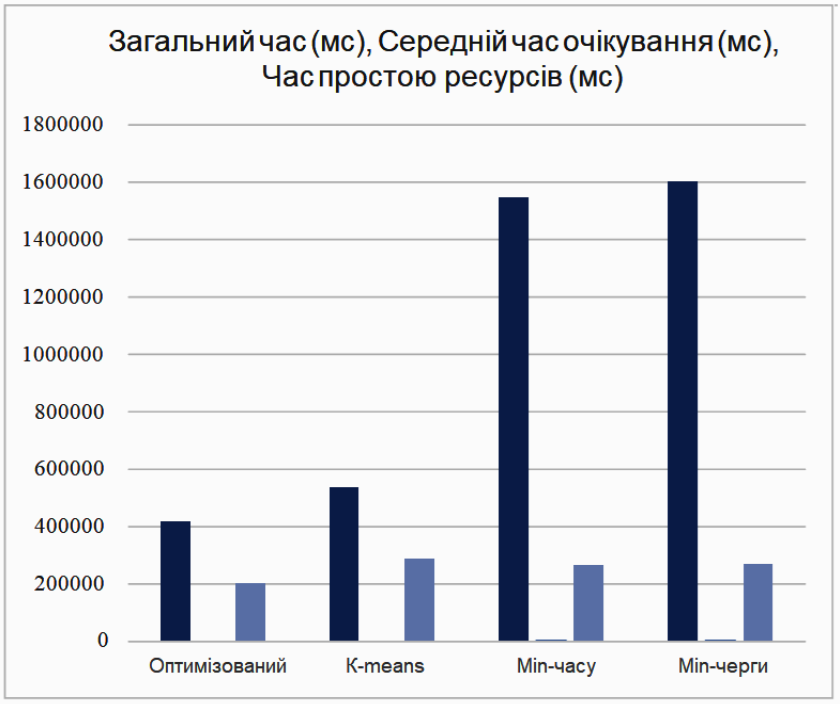


Рисунок 3.5 – Ефективність КІС в режимі недовантаженості

З графіка (рисунок 3.5) видно що в алгоритмі балансування навантаження важливою є розподіленість ОР, оскільки навіть незначна тривалість затримки повідомлень між ОР, ПВЗ та СД може здійснювати значний вплив на загальну тривалість виконання задач та середню тривалість очікування.

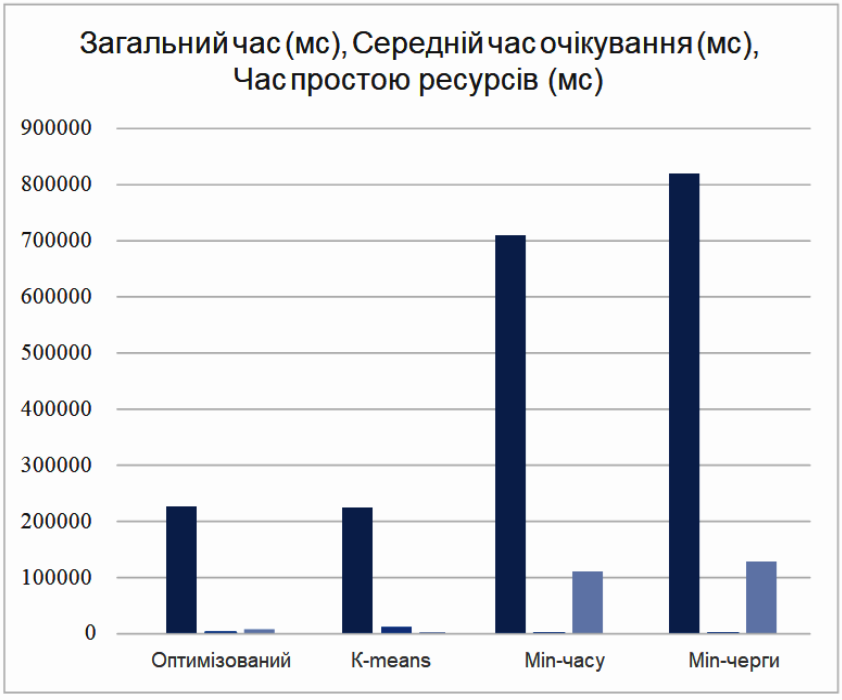


Рисунок 3.6 - Ефективність КІС в режимі перевантаженості

З графіків видно, що при недовантаженості КІС, тривалість очікування мінімальна, оскільки завдання оброблялися швидше, ніж надходили у загальну чергу. Таким чином, ОР витрачали менше часу на очікування задач, що призводило до значного збільшення тривалості простою ОР. У такому випадку оптимізованим алгоритмом використовувався завеликий інтервал кешування, що забезпечує підвищення ефективності порівняно з іншими алгоритмами завдяки не частому зверненню ПВЗ до ОР.

У випадку перевантаженості РСОД, коли задачі надходили в чергу швидше, ніж вони могли бути оброблені, інтервал кешування був скорочений, що дозволило оновлювати інформацію щодо поточної завантаженості ОР частіше, що в свою чергу покращило точність балансування навантаження

Результати експериментів підкреслюють практичні застосування та переваги розробленого алгоритму балансування навантаження для РСОД, оскільки він. дозволяє оптимізувати використання ОР, що призводить до підвищення загальної продуктивності системи. Використання ресурсів стає більш ефективним завдяки врахуванню поточних параметрів та стану ресурсів.

Алгоритм допомагає мінімізувати час очікування задач в черзі завдяки розумному розподілу навантаження. Задачі розподіляються на ОР з урахуванням їхнього стану та віддаленості від СД, що зменшує середній час очікування. Він також сприяє покращенню якості обслуговування завдяки раціональному вибору ОР для виконання задач. Врахування параметрів ОР дозволяє вибирати найбільш потужні ресурси для максимізації продуктивності.

Переваги розробленого алгоритму:

– адаптивність до змінних умов - він здатен враховувати різноманітні параметри системи, такі як інтенсивність завантаження та продуктивність ресурсів, і реагувати на них;

– зменшення надмірного простою ОР в системі дозволяє максимізувати їхню використовуваність;

– дозволяє РСОД ефективно реагувати на різкі зміни в завантаженості та ОР, забезпечуючи оптимальне розподілення завдань;

– ефективно зменшує навантаженя на ОР, оновлюючи інформацію про їхній стан тільки в разі потреби, що допомагає економити ОР.

3.2 Розробка структури комп'ютерно-інтегрованої системи

Схема КІС, що використовуватиме розроблений оптимізований алгоритм балансування навантаження для підвищення ефективності РО ВОД, буде містити ряд компонентів та функціональних блоків:

– користувачі - джерело завдань для обробки.

– інтерфейс користувача - дозволяє користувачам надсилати свої завдання та отримувати результати обробки, також може надавати інформацію про стан системи та розподіл завдань;

– ОР - виконують завдання, які надходять від ПВЗ, кожен з яких має свої характеристики, такі як продуктивність, віддаленість від СД тощо;

– кеш стану ОР - компонент зберігає інформацію про поточну продуктивність, віддаленість від СД та інші параметри кожного ОР, що оновлюється з певною регулярністю, щоб враховувати зміни в системі;

– ПВЗ - отримує задачі від користукачів і використовує розроблений алгоритм для ефективного їх розподілу між ОР використовуючи кеш їх стану для прийняття рішень про оптимальний розподіл;

– СД - компонент, де зберігаються дані, які можуть бути необхідні для виконання задач, доступ до них враховується алгоритмом;

– блок моніторингу та звітності - відслідковує ефективність роботи системи та аналізує результати РО задач, генерує звіти та надає інформацію про продуктивність та стан ОР.

Запропонований оптимізований алгоритм в ПВЗ та кеш. КІС для підвищення ефективності РОВОД - це складна інформаційна система, яка включає в себе різноманітні компоненти та підсистеми для оптимізації

обробки та розподілу завдань між ОР:

– клієнтські додатки або веб-інтерфейси – забезпечують взаємодіють користувачів із системою через спеціальні. Вони надсилають завдання на обробку і отримують результати. Клієнтські додатки можуть бути доступні на різних платформах, включаючи комп'ютери, мобільні пристрої тощо.

– мережа забезпечує комунікацію між усіма компонентами системи, передає завдання від клієнтських додатків до ПВЗ та ОР, а також дозволяє обмін даними між компонентами.

У КІС для підвищення ефективності РО ВОД, контролер може розміщуватися в різних місцях залежно від конкретної архітектури та потреб системи, заокрема:

– Централізований контролер розміщений на окремому обчислювальному вузлі або сервері. Він відповідає за прийняття завдань від клієнтів, вибір ресурсів для виконання завдань та їхній розподіл. Дозволяє забезпечувати більший контроль над розподілом завдань і надавати можливість використання централізованої стратегії балансування навантаження.

– Децентралізований контролер - не є окремим вузлом, а є розподіленим між ОР. Кожен ОР може мати свій власний локальний контролер, який відповідає за розподіл завдань. Локальні контролери можуть співпрацювати та обмінюватися інформацією для координації балансування навантаження в системі.

– Розподілений контролер - може бути розміщений на кожному ОР, а також на центральному сервері. Він може діяти як локальний контролер для власного ресурсу та як глобальний контролер для координації балансування навантаження між ОР.

– Розподілений контролер з хмарним підходом - у великих РСОД вони можуть бути інтегровані в хмарні послуги. Вони можуть діяти як частина хмарної інфраструктури та автоматично керувати балансуванням

навантаження між різними ресурсами.

Структура КІС для підвищення ефективності РО ВОД може виглядати наступним чином (рисунок 3.7):

– клієнти можуть бути програмами або користувачами, які взаємодіють з системою через інтерфейси;

– сервер є центральним вузлом системи та може виконувати різні функції;

– ОР - фізичні або віртуальні сервери, які виконують обчислення;

– локальні контролери - відповідають за обробку завдань на своєму ОР, а також за збору та надсиланню інформації про стан ОР до сервера;

– мережа забезпечує зв'язок між клієнтами, сервером, контролерами і ОР, може бути дротовою або безпровідною, а також може використовувати різні протоколи комунікації;

– СД, необхідних для виконання завдань;

– кеш контролера - контролера використовується для збереження інформації про стан ОР, швидкості обробки та інших параметрів та допомагає приймати рішення про розподіл завдань.

– кеш СД може використовуватися для збереження частини даних, щоб прискорити доступ до них, може використовуватися ОР для зменшення часу доступу до ВОД.

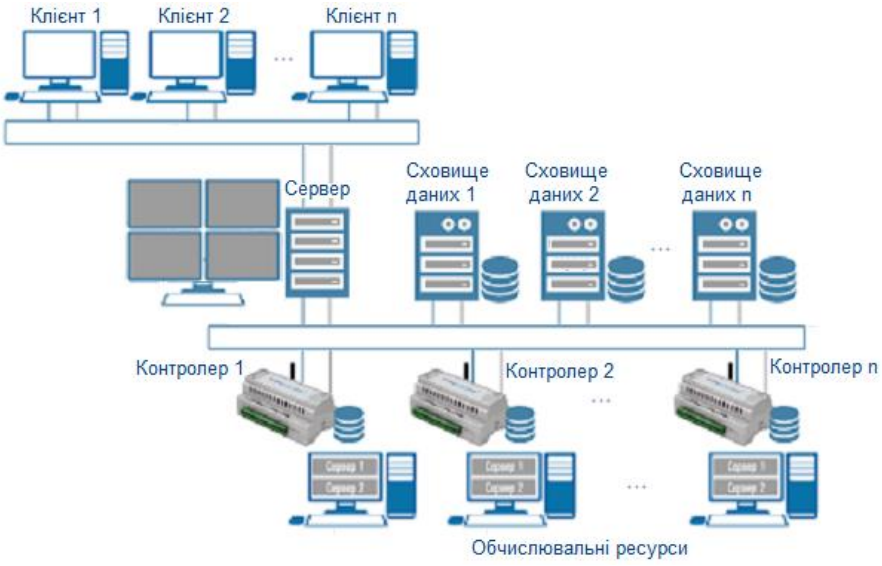


Рисунок 3.7 – Структура КІС

Система працює наступним чином: клієнти надсилають свої завдання на центральний сервер, який визначає, які ресурси відповідатимуть за обробку завдань. Центральний сервер може взаємодіяти з системним СД, де зберігаються дані, необхідні для обробки завдань. Завдання, які подані клієнтами, можуть вимагати різного роду обчислень та доступу до різних даних.

Основна роль центрального сервера полягає в ефективному розподіленні завдань між обчислювальними ресурсами. Він аналізує властивості завдань, вимоги до ресурсів та їх доступність, а також враховує інші параметри, які можуть впливати на виконання завдань.

Локальні контролери, які розміщені на обчислювальних ресурсах, отримують завдання від центрального сервера. Вони відповідають за управління обчисленнями на своєму ресурсі та взаємодію зі СД, де можуть зчитувати необхідні дані для обробки завдань.

Центральний сервер і локальні контролери взаємодіють за допомогою мережі для обміну даними та станами виконання завдань. Крім того, система може мати моніторинг та управління для нагляду за ресурсами та задачами, а також для налаштування параметрів роботи системи.

Запропонована КІС дозволяє ефективно розподіляти завдання між ОР, враховуючи їхню різноманітність та розподіленість. Такий підхід допомагає оптимізувати використання ОР та підвищувати ефективність РО ВОДх.

Центральний сервер може взаємодіяти з локальни ми контролерами, які розміщені на ОР, для передачі завдань та отримання статусу виконання. КІС використовує комунікаційний протокол для обміну даними між центральним сервером і контролерами. Ця взаємодія дозволяє серверу ефективно розподіляти завдання та координувати обчислення на різних ОР, враховуючи їхні можливості та навантаження.

Локальні контролери відповідають за прийняття завдань від центрального сервера, виконання їх на ОР та повідомлення про статус виконання. Вони можуть взаємодіяти із системним СД для зчитування необхідних інформаційних ресурсів для виконання завдань.

Важливою частиною системи є мережа, яка забезпечує зв'язок між центральним сервером і локальними контролерами. Ця мережа дозволяє передавати дані та інструкції між центральним сервером і контролерами, що дозволяє ефективно керувати обробкою завдань на різних ОР.

КІС також може включати в себе додатки для моніторингу та управління, що дозволяють операторам наглядати за роботою ресурсів та задачами, а також налаштовувати параметри системи для оптимальної продуктивності.

3.3 Розробка програмного забезпечення проектованої системи

В додатку А представлено лістинг коду на мові Java розробленого в середовищі IntelliJ IDEA, який включає такі класи як ClusterEmulator, LoadBalancer, Resource, Task і Utils, основні концепції та функції яких:

– ClusterEmulator - це головний клас, який містить метод main, де виконуються експерименти з різними алгоритмами балансування навантаження. Його функції включають генерацію ресурсів та завдань, виконання експериментів з різними алгоритмами та запис результатів у вихідний файл.

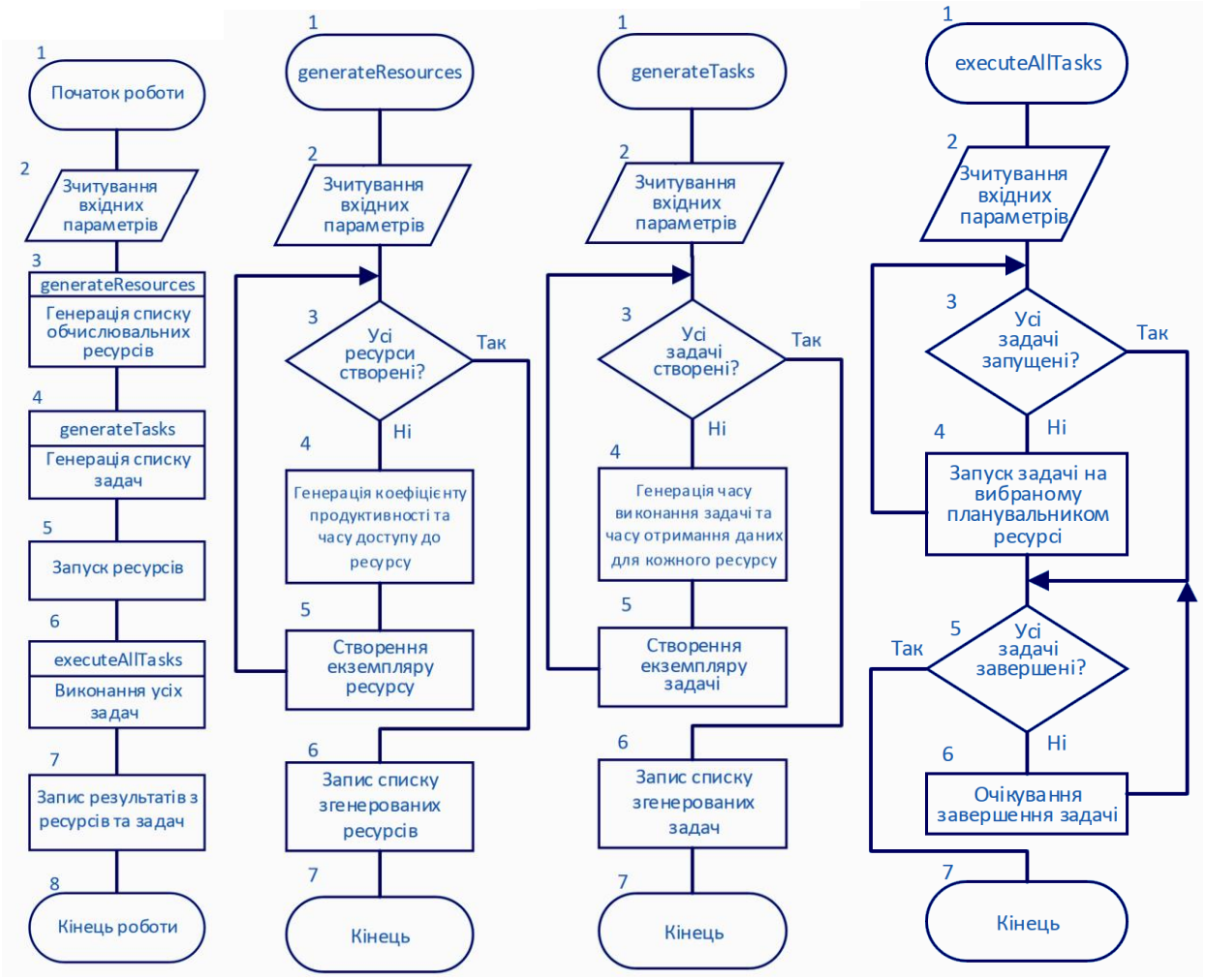
– LoadBalancer – клас, що відповідає за балансування навантаження на ресурси за допомогою різних алгоритмів. Він містить методи для вибору ресурсу для задачі відповідно до обраного алгоритму, наприклад, випадковий вибір, циклічний вибір, вибір ресурсу з найменшим часом виконання та інші. Крім того, в цьому класі використовуються кеш та інші параметри для оптимізації вибору ресурсу.

– Resource - клас ОР, кожен з яких має свою швидкість обробки, час відповіді (ping time), чергу завдань та статистику часу простою. Клас також включає методи для виконання завдань та оновлення статистики.

– Task - клас завдань, які розподіляються між ОР. Завдання мають час виконання, час очікування та методи для виконання. Він також містить дані про час доступу до ресурсів.

– Utils - клас, який містить допоміжні методи для генерації ресурсів та завдань.

На рисунку 3.8а наведено схему роботи КІС. На першому етапі проводиться читання вхідних параметрів КІС, зокрема, кількості задач та ОР. Далі, важливими параметрами є «середній час виконання задачі та допустиме відхилення цього часу» [21], які емулятор буде генерувати у заданих межах. Також обов'язковими є визначення максимального часу доступу до ОР від ПВЗ, який генерується окремо для кожного ОР, та доступу від кожного ОР до сховища БД. Останнім параметром є інтервал поступлення задач до загальної черги.



а) б) в) г)

Рисунок 3.8 – Схема роботи проектованої КІС

На наступному етапі роботи КІСефективного опрацювання ВОД, генерується список ОР (рисунок 3.8б) і завдань (рисунок 3.8в). У випадку

генерації списку ОР, вхідні параметри це кількість, рівень ефективності та максимальний час доступу до ОР. Щодо завдань, функція враховує інформацію щодо кількості потрібних завдань, мінімального та максимального часу на виконання кожної, список ОР (необхідний для створення переліку значень часу доступу до ВОД від кожного ОР) і максимального часу доступу до ВОД.

Далі відбувається запуск кожного з ОР у окремому потоці та очікування відповіді від ОР про їх готовність щодо приймання завдань. Потім відбувається перебір завдань за переліком, і кожне з них поступає ПВЗ, який отримує також правила розподілу завданнь та список доступних ОР (рисунок 3.8г).

Завершення даного етапу відбувається після виконання всіх завдань та вивільнення всіх ОР. Важливим є те, кожен з ОР відстежується менеджером, яки йотримує дані щодо їх стану, зокрема завантаження та простій на певний момент часу. Після того, як ОР завершують роботу всі потоки завершуються автоматично і КІС також припиняє роботу.

На завершальному етапі відбувається зчитування та збереження даних, зокрема загалоного часу виконання завдант, середнього та максимального часу очікування в черзі завдань, середнього та максимального часу простою ОР, з метою їх подальшого аналізу.

ПЗ є емулятором для дослідження алгоритму балансування навантаження в РСОД. Дозволяє створити обчислювальний кластер, розподілити завдання і вимірювати різні показники ефективності для різних алгоритмів балансування навантаження, що може бути корисним для експериментів та досліджень у галузі РОД. Передбаченоє можливість одночасного запуску декількох режимів щодо ПВЗ, що є корисним для проведення порівняльного аналізу вихідних даних.

Симуляція КІС - це процес моделювання та аналізу функціонування кластера, який складається з декількох вузлів або комп'ютерів, які спільно працюють для обробки завдань і обчислень. Симуляція дозволяє вивчити та оцінити різні аспекти роботи КІС без фактичної побудови або експлуатації

реального обладнання.

Основна мета та переваги симуляції РСОД включають:

– оцінка продуктивності при різних умовах і навантаженнях, що допомагає визначити, як ефективно використовувати ресурси та оптимізувати конфігурацію кластера;

– аналіз завантаженості - дозволяє відтворювати сценарії різної завантаженості РСОД, що допомагає визначити, як система поводиться під навантаженням та як розподіляються ресурси між задачами;

– валідація алгоритмів і стратегій ПВЗ, систем управління та інших аспектів керування РСОД;

– тестування на практичність - дозволяє перевірити, як рсод реагує на різні варіанти випадків та непередбачені ситуації, не ризикуючи реальним обладнанням;

– оптимізація ресурсів - За допомогою симуляції можна визначити, як оптимізувати розміщення та виділення ОР для досягнення найкращої продуктивності та розподілу завдань.

Враховуючи обмежену доступність реальних ОР для великомасштабних тестів, симуляція є незамінним інструментом для ефективного тестування та вдосконалення алгоритмів. Для симуляції КІС використовувалося спеціалізоване ПЗ та інструменти, які дозволили моделювати різні аспекти роботи кластера, що включає в себе імітацію ресурсів, мережі, задач і робочих навантажень.

Пакет Globus Toolkit є одним із популярних програмних засобів для створення та керування кластер-інфраструктурою. Він дозволяє розробникам створювати рішення для обчислювання в розподіленому середовищі та використовувати стандарти безпеки та обміну даними.

Симуляція дійсно є потужним інструментом для аналізу та оптимізації алгоритмів, і вона дозволяє вам вирішувати складні завдання без великих витрат на реальні ОР.

ВИСНОВКИ

В роботі запропонована КІС яка забезпечує підвищення ефективності розподіленої обробки ВОД за рахунок використання вдосконаленого алгоритму планування виконання завдань.

В роботі проведені дослідження технологій опрацювання даних у розподілених системах. Встановлено, що РСОД є потужним інструментом для зберігання, обробки та аналізу ВОД. Вони дозволяють використовувати різноманітні ОР, навіть якщо вони розташовані на різних серверах, належать різним організаціям чи використовуються для вирішення різноманітних завдань.

Проведений аналіз моделей обробки даних та алгоритмів планування задач в кластерних обчисленнях. Визначено, що вбір алгоритму ПВЗ дач для РСОД залежить від конкретних вимог і особливостей системи. Не існує універсального алгоритму, який підходить для всіх сценаріїв. Це вимагає створення оптимізованого алгоритму, який дозволить підлаштовуватися під змінні умови.

Також були проаналізовано обмеження існуючих алгоритмів балансування навантаження, і запропоновано підходи до їх усунення. В результаті визначено параметри, які є важливими для вибору найкращого ОР, зокрема гетерогенність ОР, віддаленість, продуктивність і можливість кешування. Гетерогенність і віддаленість ОР мають великий вплив на тривалість виконання задач і очікування завдання в черзі. Врахування цих параметрів дозволяє покращити ефективність розподілу завдань і зменшити навантаження на ОР. Проведений аналіз дозволив визначити вимоги до проектованої КІС.

В результаті запропоновано оптимізований алгоритм планування завдань, який комплексно враховує всі аспекти та обмеження ПВЗ у РСОД для підвищення ефективності та оптимізації використання ОР. Використано кешування станів ОР у підсистему ПВЗ, що дозволяє покращити розподіл задач та зменшити тривалість очікування. Розроблено блок-схему роботи

оптимізованого алгоритму ПВЗ та доведено його ефективність експериментальним шляхом. Розроблено алгоритм роботи КІС на основі адаптивних алгоритмів ПВЗ в кластер-системах може покращити їхню ефективність та реагування на змінні умови, що робить їх ефективним інструментом для обробки великих обсягів даних у різних сценаріях.

Розроблено структуру КІС що використовуватиме розроблений алгоритм балансування навантаження, визначено її компоненти та їх функціональні можливості. Розроблено ПЗ проектованої КІС, що досзоляє створити обчислювальний кластер, розподілити завдання і вимірювати різні показники ефективності для різних алгоритмів балансування навантаження, що може бути корисним для експериментів та досліджень у галузі РОД. Розроблено блок-схему роботи проектованої системи.

Використання КІС, базованої на алгоритмах ПВЗ в РСОД, може значно покращити ефективність та здатність цих систем реагувати на змінні умови. Такий підхід робить їх потужним інструментом для обробки великих обсягів даних в різних сценаріях, де можуть бути різноманітні обчислювальні завдання та умови використання ресурсів.

Проектована система дозволить оптимізувати розподіл задач між ОР, враховуючи їх гетерогенність, віддаленість та інші параметри, що впливають на продуктивність. Внаслідок цього, система може більш ефективно виконувати завдання і зменшувати час їх обробки. Все перелічене забезпечити КІС підвищення ефективності обробки великих обсягів даних дуже цінним рішенням для сучасних організацій та дослідницьких проєктів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гаврилюк В.Комп'ютерно-інтегровані системи управління".- НТУУ "КПІ".- 2017.- 300с.
2. Трегуб В.Г. Проектування систем автоматизації.- Ліра-К, 2019.- 334с.
3. Любицький С.В. Основи побудови комп’ютерно-інтегрованих систем / С.В. Любицький, П.В. Новіков.- КПІ ім. Ігоря Сікорського.-Київ, 2020. - 77 с.
4. Терещенко Т.О. Розподілені мікропроцесорні системи.– Київ : КПІ ім. Ігоря Сікорського, 2018. - 192 с.
5. Ghosh Ratan K., Ghosh Hiranmay. Distributed Systems: Theory and Applications .-Wiley-IEEE Press, 2023. - 563 p.
6. Ван Стін М., Таненбаум А.С. Розподілені системи. 3-е видання. Версія 01. - CreateSpace Independent Publishing, 2017. - 596 с.
7. Mo H., Sansavini G., Xie M. Cyber-Physical Distributed Systems: Modeling, Reliability Analysis and Applications .- Wiley, 2021. - 224 p.
8. Кузьмякова А. Паралельні, паралельні та розподілені обчислення.- Arcler Press, 2023. - 260 с.
9. Castellani B.C., Rajaram R. Big Data Mining and Complexity.- SAGE Publications, 2022. - 232 p.
10. Das S., Rao R.S., Das I., Jain V., Singh N. (eds.) Cloud Computing Enabled Big-Data Analytics in Wireless Ad-hoc Networks.- CRC Press, 2022. - 291 p.
11. Luengo J. et al. Big Data Preprocessing Enabling Smart Data. Springer, 2020. - 272p.
12. Kumar S., Mapp G., Cengiz K. Intelligent Network Design Driven by Big Data Analytics, IoT, AI and Cloud Computing.- The Institution of Engineering and Technology, 2022. - 427 p.
13. Стихальська С.В. Комп’ютерно-інтегрована система розподіленої обробки даних / С.В. Стихальська, В.Ю. Шаков.- Збірник матеріалів

проблемно-наукової міжгалузевої конференції «Автоматизація та комп’ютерно-інтегровані технології», Тернопіль, 2023. -с. 60-63.

14. Драпак В.І. Підвищення ефективності опрацювання даних у розподілених системах / В.І. Драпак, Р.О. Питель, А.М. Романів, В.Ю. Шаков.- Матеріали науково-практичного симпозіуму «Захист інформації», Тернопіль, 2023. – 67-72 с.

15. Hadoop.- [Електронний ресурс].- Режим доступу: https://hadoop.apache.org/

16. Apache Spark.- [Електронний ресурс].- Режим доступу: https://spark.apache.org/

17. Apache Flink.- [Електронний ресурс].- Режим доступу: https://flink.apache.org/

18. Apache Cassandra.- [Електронний ресурс].- Режим доступу: https://cassandra.apache.org/

19. Kubernetes VS Docker Swarm – What is the Difference? .- [Електронний ресурс].- Режим доступу: https://www.freecodecamp.org/news/kubernetes-vs-docker-swarm-what-is-the-difference/

20. Atif M., Groote J.F. Understanding Behaviour of Distributed Systems Using mCRL2.- Springer, 2023. - 241 p.

21. Burns B. Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services.- O’Reilly Media, 2018. - 166 p.

22. Пасічник В.В. Паралельні та розподілені обчислення / В.В. Пасічник, С.А. Лупенко, А.М. Луців.- Магнолія 2006, 2021.- 648с.

23. Gordon-Ross A., Munir A., Ranka S. Modeling and Optimization of Parallel and Distributed Embedded Systems.- New York: John Wiley & Sons Inc, 2016. - 510 p

24. Erciyes K. Distributed Real-Time Systems: Theory and Practice.- Springer, 2019. - 347 p.

25. Parsian Mahmoud. Data Algorithms: Recipes for Scaling Up with Hadoop and Spark.- O’Reilly, 2015. - 858 p.

26. Корочкін О.В. Паралельні та розподілені обчислення.Вибрані розділи: Навч. посібник. / О.В.Корочкін, Русанова О.В. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 123 с.

27. Kopetz H., Steiner W. Real-Time Systems: Design Principles for Distributed Embedded Applications.- Springer, 2022. - 411 p.

28. Czaja L. Introduction to Distributed Computer Systems. Principles and Features.- New York: Springer, 2018. - 263 p.

29. Joshi Unmesh. Patterns of Distributed Systems.- Pearson Education/Addison-Wesley, 2023. - 492 p.

30. Perry Michael L. The Art of Immutable Architecture: Theory and Practice of Data Management in Distributed Systems.- Apress, 2020. - 435 p.