

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» _____ 2023 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Оптимізаційна багатофакторна модель оцінки показників
функціонування розподіленої комп'ютерної системи»**

Виконав:

студент IV курсу, групи КА-95
Новохатський Дмитро Євгенович _____

Керівник:

Доцент, к.ф.-м.н., доцент кафедри ММСА Яковлева А. П. _____

Консультант з економічного розділу:

доцент, к.е.н., доцент кафедри ТПЕ Рощина Н.В. _____

Консультант з нормоконтролю:

к.ф.-м.н., доцент кафедри ММСА Статкевич В. М. _____

Рецензент:

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ІН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«__» травня 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Новохатському Дмитру Євгеновичу

1. Тема роботи «Оптимізаційна багатофакторна модель оцінки показників функціонування розподіленої комп'ютерної системи», керівник роботи к.ф.-м.н., доцент Яковлева Алла Петрівна, затверджені наказом по університету від «30» травня 2023 р. № 2065-с
2. Термін подання студентом роботи 12.06.2023.
3. Вихідні дані до роботи: параметри компонентів розподіленої комп'ютерної системи (РКС), запити різного рівня складності.
4. Зміст роботи: дослідження впливу параметрів на оцінки розподіленої комп'ютерної системи (РКС) та їх аналіз.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.
6. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Н. В., доцент кафедри ТПЕ		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	17.04.23-21.04.23	Виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	17.04.23-21.04.23	Виконано
3	Ознайомлення з ДСТУ 3008-95 та стандарти ЄСПД	21.04.23-28.04.23	Виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	28.04.23-05.05.23	Виконано
5	Завершення роботи над першим варіантом частини БДР	05.05.23-12.05.23	Виконано
6	Проведення роботи над експериментальною частиною БДР	05.05.23-12.05.23	Виконано
7	Проведення роботи над програмним продуктом	12.05.23-19.05.23	Виконано
8	Оформлення БДР та аналіз отриманих результатів	12.05.23-21.05.23	Виконано

Студент

Дмитро НОВОХАТСЬКИЙ

Керівник

Алла ЯКОВЛЕВА

РЕФЕРАТ

Дипломна робота містить 120 с., 27 рис., 6 табл., 6 дод., 21 джерело.

Ключові слова: **МОДЕЛЬ ОЦІНКИ, МОДЕЛЬ УПРАВЛІННЯ РЕСУРСАМИ, РОЗПОДІЛЕНІ КОМП'ЮТЕРНІ СИСТЕМИ, ОПТИМІЗАЦІЯ РЕСУРСІВ**

Об'єкт дослідження: розподілена комп'ютерна система (РКС), яка включає в себе розподілені сервери, мережеву інфраструктуру, а також бази даних та їх взаємодію.

Мета роботи: підвищення ефективності в управлінні ресурсами, а також покращення продуктивності розподіленої комп'ютерної системи (РКС).

Предмет дослідження: розробка оптимізаційної багатофакторної моделі, що оцінює показники функціонування розподіленої комп'ютерної системи з використанням баз даних, а також визначення факторів, що впливають на її ефективність.

Отримані результати: було побудовано модель для оцінки показників у розподіленій комп'ютерній системі. Також була створена програма для відтворення змодельованої архітектури, а також збору оцінок показників, прогнозування часу виконання та побудову моделі у вигляді графу.

Для подальшого покращення розробленої моделі можна додати декілька оцінок, розробити більш гнучкий підхід до розробки системи безпеки даних. Ще можна додати блок черги з пріоритетами, який буде розподіляти запити у черги перед подачею на вузли, що зменшить навантаження на систему та покращить її ефективність. А також застосування нейронних мереж для аналізу щодо оптимізації параметрів системи відповідно її характеристикам.

ABSTRACT

The thesis contains 120 p., 27 figures, 6 tables, 6 appendices, 21 sources.

Keywords: EVALUATION MODEL, RESOURCE MANAGEMENT MODEL, DISTRIBUTED COMPUTER SYSTEMS, RESOURCE OPTIMIZATION

Object of research: distributed computer system (DSC), which includes distributed servers, network infrastructure, as well as databases and their interaction.

The purpose of the work: increasing the efficiency in resource management, as well as improving the performance of the distributed computer system (DSC).

The subject of the study: development of an optimization multifactorial model that evaluates the indicators of the functioning of a distributed computer system using databases, as well as the determination of factors affecting its effectiveness.

Results: a model was built for evaluating indicators in a distributed computer system. A program was also created to reproduce the modeled architecture, as well as collect metrics, predict execution times, and build the model in the form of a graph.

To further improve the developed model, one can add several estimates, develop a more flexible approach to the development of a data security system. Also, we can also add a queue block with priorities, which will allocate requests to queues before submitting them to nodes, which will reduce the load on the system and improve its efficiency. As well as the application of neural networks for analysis on the optimization of system parameters according to its components.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛЕЙ ОЦІНКИ ПОКАЗНИКІВ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ	8
1.1 Визначення РКС та її існуючих типів.....	8
1.2 Огляд особливостей РКС та метрик її функціонування	14
1.3 Опис існуючих моделей та засобів оцінки функціонування з використанням баз даних	17
1.4 Висновки до розділу 1	22
РОЗДІЛ 2. АНАЛІЗ МОДЕЛЕЙ ОПТИМІЗАЦІЇ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ	23
2.1 Розгляд основних підходів до оптимізації	23
2.2 Огляд існуючих моделей оптимізації	26
2.3 Порівняння різних методів оптимізації.....	28
2.4 Аналіз механізмів управління ресурсами РКС	32
2.5 Висновки до розділу 2	33
РОЗДІЛ 3. ОПТИМІЗАЦІЙНА БАГАТОФАКТОРНА МОДЕЛЬ ОЦІНКИ ПОКАЗНИКІВ РКС НА ОСНОВІ МЕЖЦЕНТРИЧНОГО ПІДХОДУ	35
3.1 Формалізація загального вигляду РКС	35
3.2 Опис архітектури тестової системи	39
3.3 Програмна реалізація.....	42

3.4 Дослідження впливу параметрів на оцінки розподіленої комп'ютерної системи (РКС).....	45
3.4 Висновки до розділу 3	55
4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	56
4.1 Постановка задачі проектування.....	57
4.2 Обґрунтування функцій програмного продукту	57
4.3 Обґрунтування системи параметрів програмного продукту	61
4.4 Аналіз експертного оцінювання параметрів	63
4.5 Аналіз рівня якості варіантів реалізації функцій	68
4.6 Економічний аналіз варіантів розробки ПП	70
4.7 Вибір кращого варіанту ПП техніко-економічного рівня	76
4.8 Висновки до розділу 4	77
ВИСНОВКИ	78
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	81
ДОДАТОК А.....	83
ДОДАТОК Б.....	93
ДОДАТОК В.....	112

ВСТУП

У сучасному інформаційному суспільстві обробка та зберігання даних мають вирішальне значення для багатьох організацій та підприємств. Баз даних (БД) є ключовими компонентами інфраструктури, що забезпечують ефективну та надійну роботу з інформацією. Але з ростом потреби на обробку великого обсягу інформації, зростає й потреба в використанні великої кількості обчислювальних ресурсів. Тут на допомогу й приходять РКС, зі своїми можливостями до паралельної обробки даних, які можуть складатися з великої кількості комп'ютерів та можуть знаходитись у різних фізичних місцях, з'єднуючись через мережу між собою. Та зростом кількості таких систем, зростає потреба в оптимізованому та ефективному використанні її ресурсів, а також у методах передачі та зберігання даних. Тому до такого функціоналу потрібно, формувати моделі імітації роботи та збору показників системи, з метою подальшого масштабування, та більш ефективного використання ресурсів.

Основна мета такої моделі полягає у визначенні оптимальних значень факторів та їх впливу на показники функціонування системи. Це може бути досягнуто шляхом використання методів аналізу даних, математичного моделювання та оптимізаційних алгоритмів.

У даному дослідженні пропонується розробка оптимізаційної багатофакторної моделі, що оцінює показники функціонування розподіленої комп'ютерної системи з використанням баз даних, а також визначення факторів, що впливають на її ефективність.

Також було розглянуто та побудовано модель для оцінки показників у РКС. Після чого, відповідно створена програма для відтворення змодельованої архітектури та збору оцінок показників, прогнозування часу виконання та побудови моделі у вигляді графу системи відповідно її характеристикам.

РОЗДІЛ 1. ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛЕЙ ОЦІНКИ ПОКАЗНИКІВ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

1.1 Визначення РКС та її існуючих типів

Розподілена комп'ютерна система (РКС) - це мережа комп'ютерів, які спільно працюють над виконанням певних завдань, обмінюються даними та ресурсами. Особливістю РКС є те, що вона може складатися з великої кількості комп'ютерів, розташованих на різних вузлах мережі, які можуть бути у різних фізичних місцях і з'єднаних мережевим зв'язком. Така система має загальний вигляд, як на рис. 1.1

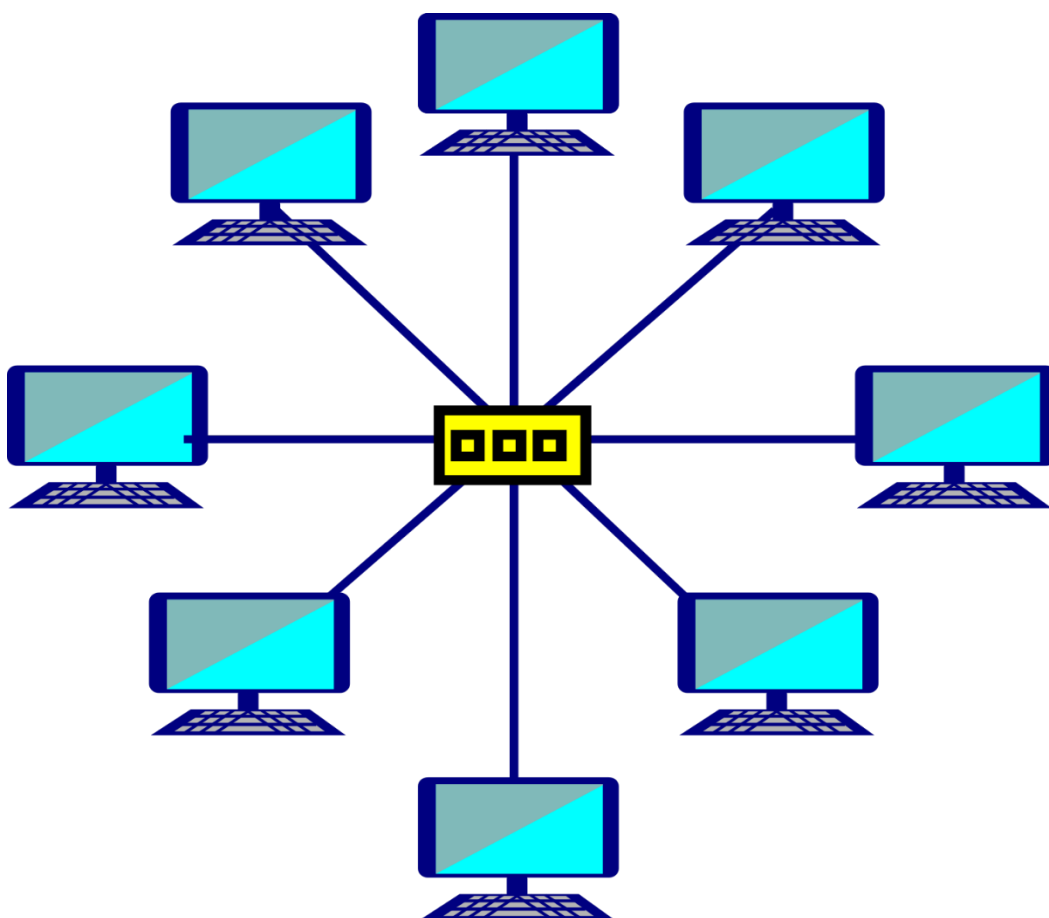


Рис 1.1. Загальний вигляд звичайної РКС

За весь період функціонування різних РКС було створено дуже багато їх типів, кожен з яких відрізняється своєю архітектурою, призначенням, або направленістю задач, які вони виконують. Розглянемо декілька більш поширених та ефективних з них:

1) Кластерна система. Такий тип РКС уявляє собою групу комп'ютерів або вузлів, які працюють разом і формують собою об'єднану систему для обчислень. комп'ютери або вузли в кластері спільно працюють на виконання певних задач та діляться між собою навантаженням. Також такі системи зазвичай використовують кластерні операційні системи або спеціальне кластерне програмне забезпечення. Такий підхід забезпечує високу доступність, масштабованість та надійність шляхом резервування вузлів та розподілу завдань. Серед існуючих вдалих прикладів використання такої системи в сучасному світі можна виділити кластерний суперкомп'ютер Summit, розташований в Oak Ridge National Laboratory в США. Який складається з більш як 4 600 вузлів і використовується для наукових досліджень у різних галузях, включаючи метеорологію, геноміку та матеріалознавство. А також кластерну систему Apache Hadoop, яка використовується для обробки та аналізу великих обсягів даних. Вона базується на розподіленому збереженні даних та обчисленнях на багатьох вузлах, що дозволяє ефективно обробляти дані у масштабі петабайт. Така система має загальний вигляд, як на рис. 1.2.

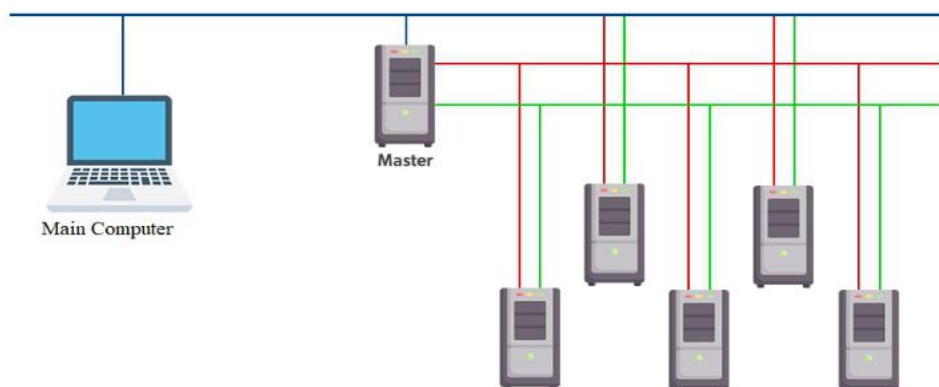


Рис. 1.2. Загальний вигляд кластерної системи

2) Сіткова (Grid) система. При такому підході кожен комп'ютер у сітці може бути незалежним та мати власний набір ресурсів. Сіткові системи сприяють використанню розподілених ресурсів, забезпечують високу продуктивність та масштабованість. Такий тип РКС зазвичай використовують для задач, які потребують багато складних обчислень, таких як наукові дослідження, моделювання та обробка великих обсягів даних, тощо. Серед існуючих вдалих прикладів використання такої системи в сучасному світі можна виділити European Grid Infrastructure (EGI), яка об'єднує розподілені обчислювальні ресурси у різних країнах Європи для наукових досліджень, та дозволяє вченим та дослідникам виконувати складні обчислення із використанням спільних ресурсів. Ще гарним прикладом буде SETI@home - проект, що використовує розподілену сіткову систему для аналізу радіосигналів і пошуку зовнішнього земного життя. Така система має загальний вигляд, як на рисунку 1.3.

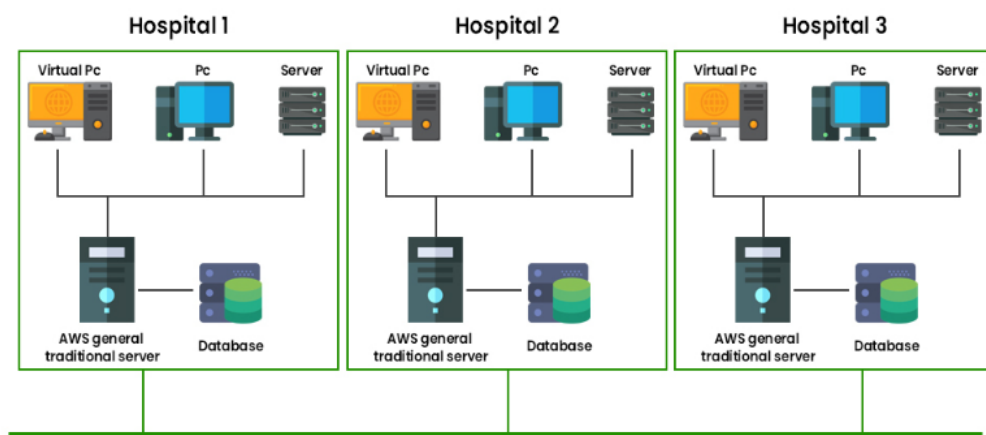


Рис. 1.3. Загальний вигляд сіткової (Grid) системи

3) Хмарна (Cloud) система. Така РКС являє собою систему в якій ресурси, такі як обчислювальна потужність, зберігання даних та програмне забезпечення, надаються через Інтернет. Такий підхід дозволяє користувачам отримувати доступ до ресурсів по запиту та масштабувати їх в залежності від потреб. У хмарних системах розподілені ресурси можуть бути орендовані на певний строк або на основі плати за використання. Гарним представником таких систем є Amazon Web Services (AWS),

який є одним з провідних провайдерів хмарних послуг, який надає широкий спектр обчислювальних ресурсів, зберігання даних та послуги штучного інтелекту. Аналогом можна вважати Microsoft Azure - інший популярний хмарний сервіс, що надає масштабовані обчислювальні та зберігання даних ресурси, а також платформу для розробки та розгортання додатків. Така система має загальний вигляд, як на рисунку 1.4.

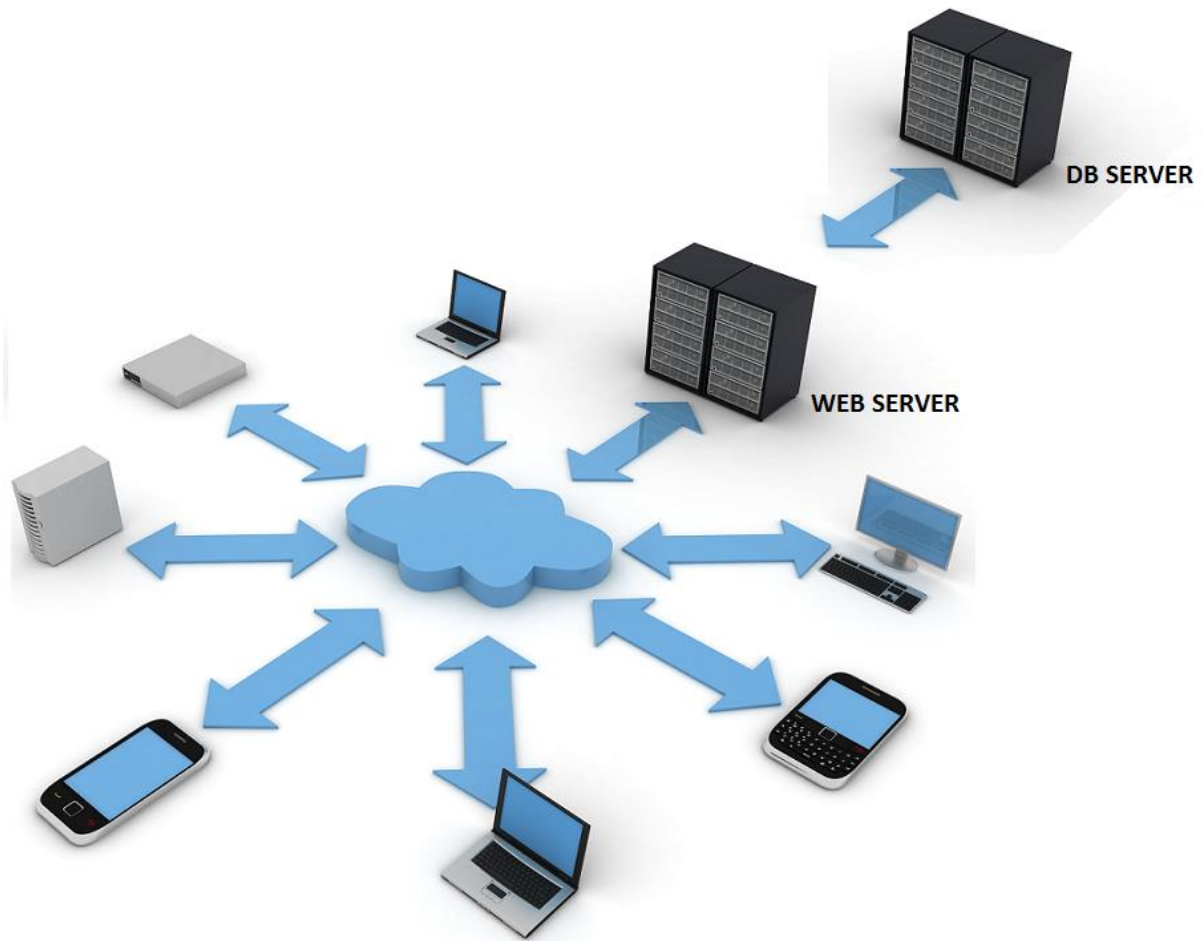


Рис. 1.4. Загальний вигляд Хмарної (Cloud) системи

4) Модель розподілених обчислень. Це модель, в якій обчислювальні завдання розподіляються між різними комп'ютерами або вузлами у мережі. Кожен вузол виконує свою частину обчислень, і результати об'єднуються для отримання кінцевого результату. Як і Сіткова (Grid) система, модель розподілених обчислень зазвичай ефективно використовувати у середі розв'язання складних задач, які вимагають великої обчислювальної потужності. В сучасному світі вдалими прикладами можна

вважати проект Folding@home, який використовує розподілені обчислення для дослідження складності біологічних процесів та пошуку ліків для лікування хвороб, таких як рак і COVID-19. Також усім відомий майнінг Bitcoin – це теж мережа розподілених комп'ютерів, які виконують обчислення для забезпечення безпеки та обробки транзакцій у криптовалютній мережі Bitcoin. Така система має загальний вигляд, як на рисунку 1.5.

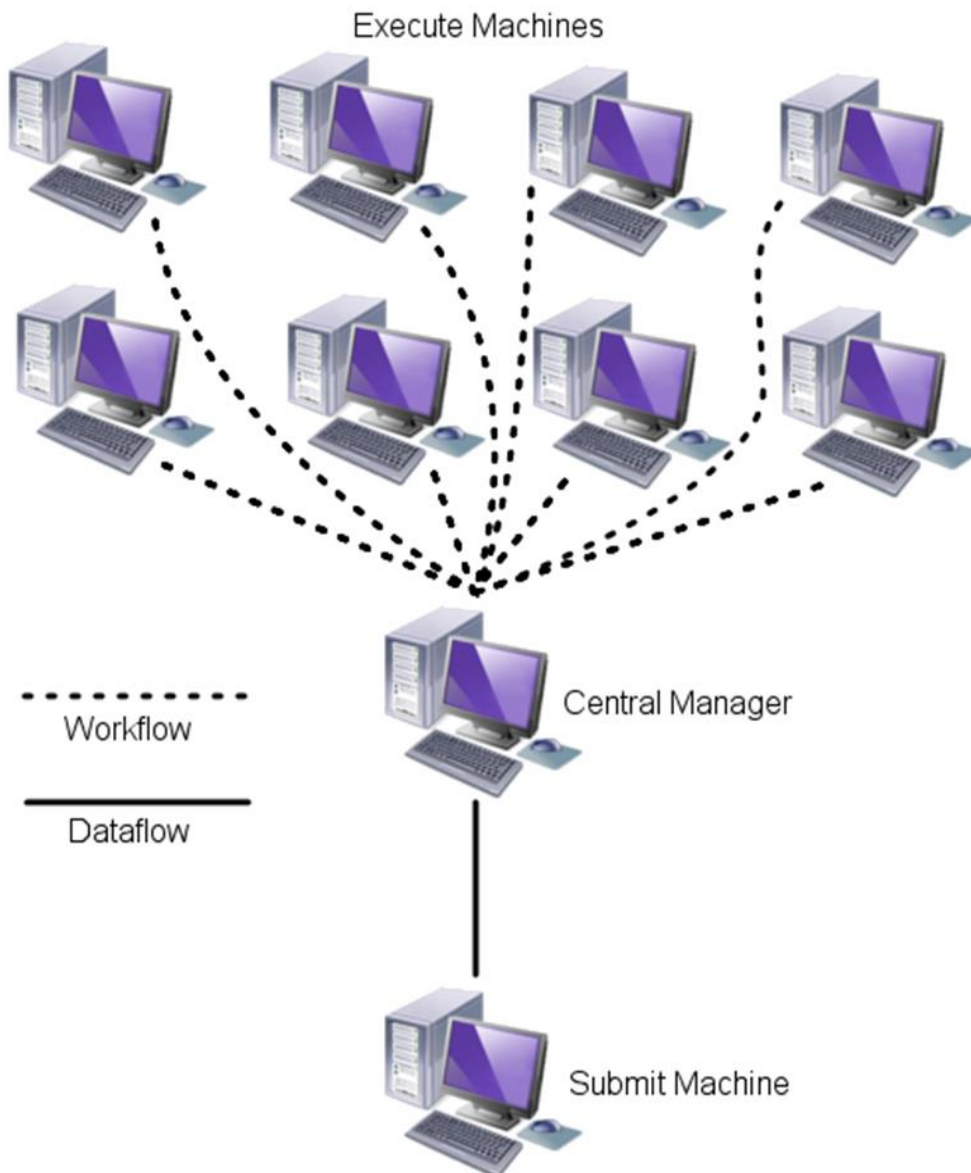


Рис. 1.5. Загальний вигляд моделі розподілених обчислень

5) Розподілені бази даних. Це система, в якій дані розподілені між різними вузлами або серверами баз даних. Кожен вузол має свою власну частину даних, а також може взаємодіяти з іншими вузлами для отримання доступу до даних. Тут

прикладом є Google Spanner, який являє собою розподілену базу даних, що була розроблена Google, та забезпечує горизонтальне масштабування, високу доступність та консистентність даних. А також Cassandra, яка теж уявляє собою розподілену базу даних, але на відміну від попереднього, вже використовується в багатьох веб-проектах, таких як Facebook та Twitter, для забезпечення швидкого доступу до великого обсягу даних у розподіленому середовищі. Така система має загальний вигляд, як на рисунку 1.6.

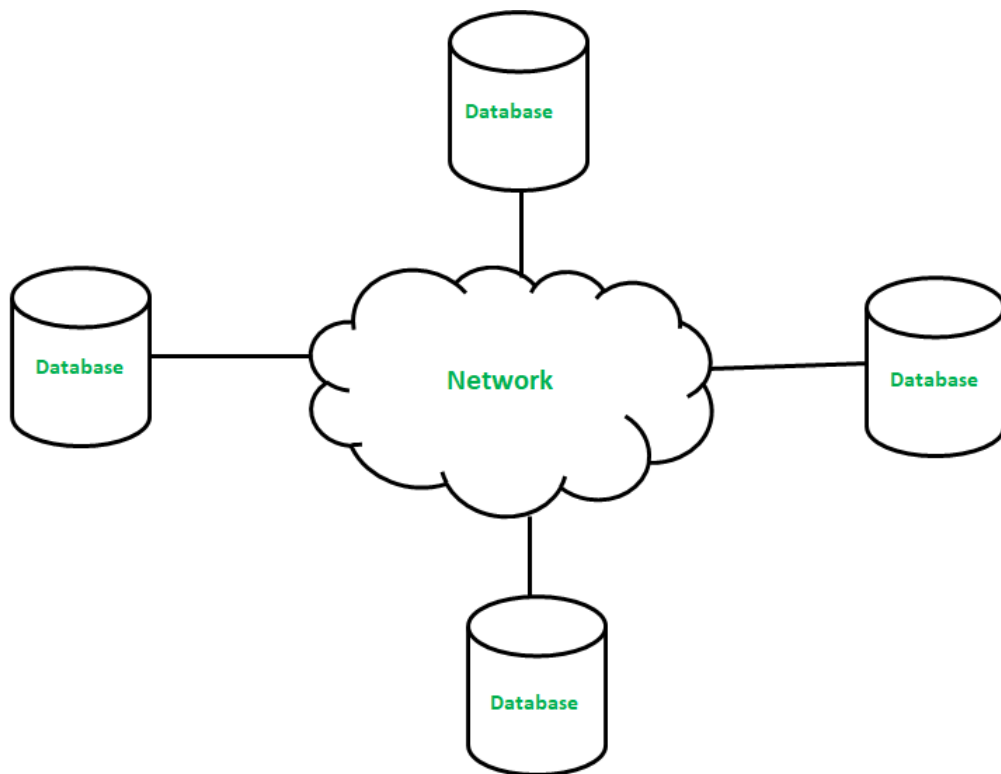


Рис. 1.6. Загальний вигляд розподіленої бази даних

Взагалі розподілені комп'ютерні системи (РКС) використовується ще в багатьох сферах, таких як банківська справа, виробництво, наукові дослідження, веб-додатки, соціальні мережі тощо. Окрім цього, з огляду на швидкий розвиток мережових технологій та зростання обсягу даних, РКС стає все більш актуальною темою для дослідження та розробки нових технологій.

1.2 Огляд особливостей РКС та метрик її функціонування

Однією з головних особливостей РКС є те, що вона складається з декількох автономних комп'ютерів, які співпрацюють для виконання певних задач. Що в свою чергу дозволяє працювати в умовах розподіленої та ненадійної мережі, а також забезпечувати паралельну обробку даних. Як ми вже розглянули, РКС можуть мати різні архітектури та топології, а також використовувати різні протоколи мережевого зв'язку та виконувати різні функції. Також серед важливих особливостей, слід виділити такі речі як:

- 1) Розподілення ресурсів. Якщо ми розглядаємо РКС з боку обчислювальних ресурсів, таких як процесори, пам'ять, сховища даних та периферійні пристрої, то вони розподілені між різними комп'ютерами системи. Це дозволяє значно ефективніше використовувати доступні ресурси, а також забезпечує високу масштабованість системи.
- 2) Комунікація та обмін даними між пристроями системи. Обмін даними так комунікація між комп'ютерами в РКС здійснюється через мережу. Такий підхід включає в себе передачу повідомлень, синхронізацію даних і координацію роботи між цими комп'ютерами. Це дуже важливо для забезпечення взаємодії та спільної роботи різних компонентів системи, не потребуючи фізичного знаходження окремих ресурсів в одному місці.
- 3) Рівномірне розділення різних задач між пристроями системи. Ми можемо виконувати розподілення різних задач між різними комп'ютерами або вузлами в РКС. Рахуємо що такі пристрої працюють паралельно, та у такому ж режимі отримують задачі для швидкого та ефективного виконання. Це дозволяє розподілити більш цілеспрямовано або ж рівномірно навантаження на систему та таким чином покращити продуктивність системи.
- 4) Висока доступність РКС та механізми, що забезпечують її. Також РКС зазвичай мають механізми для забезпечення високої доступності. Тобто, якщо в нас один

комп'ютер або вузол відмовляє, інші комп'ютери цієї системи можуть продовжувати працювати без перебоїв, що забезпечує неперервну роботу системи. А також нам потрібно приділяти велику увагу забезпеченню такої високої доступності. Досягти ми цього можемо шляхом використання наступних механізмів та стратегій резервування та відновлення:

- Розміщення резервних екземплярів компонентів та пристроїв. Для забезпечення більш високої та безперебійної роботи РКС слід розташовувати резервні екземпляри компонентів та пристроїв, для таких важливих компонентів системи, як сервери, мережеві маршрутизатори або сховища даних, та автоматично переключатися на них у разі відмови основного компоненту системи.
- Створення реплікації даних. Ми можемо створювати реплікації даних на кількох вузлах системи, що у подальшому дозволяє нам продовжувати роботу, навіть якщо будемо мати справу з відмовою одного з вузлів, дані якого були своєчасно репліковано. Також ми можемо здійснювати цій процес на різних рівнях, від окремих записів до цілих баз даних.
- Балансування навантаження. Якщо ми маємо справу з великою РКС, до доцільно буде використовувати механізми балансування навантаження. Завдяки праці яких ми можемо розподіляти завдання між різними компонентами системи. Такий підхід дозволяє уникнути перевантаження окремих компонентів системи у майбутньому.

5) Масштабованість. Розподілені системи можуть бути легко масштабовані шляхом додавання нових комп'ютерів або вузлів до системи. Що дозволяє збільшити потужність системи і обробляти більшу кількість задач та більший об'єм даних. Масштабованість у свою чергу поділяють на:

- Горизонтальна масштабованість (scale-out). При горизонтальній масштабованості наша РКС може бути розширена шляхом додавання нових вузлів або компонентів. Це можуть бути як нові сервери, так й вузли обробки або сховищ даних. При такому підході РКС зазвичай мають більшу гнучкість, оскільки можна легко додавати нові ресурси до системи.

- Вертикальна масштабованість (scale-up). При вертикальній масштабованості наша РКС розширюється шляхом збільшення потужності компонентів або ж їх ресурсів. Це може бути як покращення процесора, так й додавання пам'яті або збільшення потужності сховища даних. Хоча такий підхід зазвичай досягає фізичних обмежень компонентів та його складніше для реалізовувати, але він може забезпечувати велику потужність та продуктивність на окремих вузлах.

Що стосується метрик, які слід враховувати при подальшому створенні нашої моделі, то нам варто враховувати такі речі як:

- 1) Продуктивність. Параметр вимірює швидкість та ефективність обробки операцій або задач системою. Метрики продуктивності можуть включати час відповіді, пропускну здатність, швидкість обробки транзакцій або запитів, завантаження ресурсів і час виконання операцій.
- 2) Доступність. При розгляданні параметра, ми дивимось на здатність системи залишатися функціонуючою і доступною для користувачів протягом певного періоду часу. Метрики доступності можуть включати час безвідмовної роботи, час відновлення після відмови, кількість та тривалість перебоїв.
- 3) Надійність. Параметр вказує на стійкість системи до відмов і здатність зберігати цілісність даних. Метрики надійності можуть включати кількість відмов, середній час між відмовами (MTBF), середній час відновлення після відмови (MTTR), а також рівень резервування даних.
- 4) Масштабованість. При розгляданні параметра, ми дивимось на здатність системи ефективно масштабуватися при зростанні обсягу даних, навантаження або користувачів. Метрики масштабованості можуть включати кількість вузлів, час відповіді при збільшенні навантаження, гнучкість додавання нових ресурсів.
- 5) Витривалість. Ця метрика вказує на здатність системи зберігати і відновлювати дані в разі відмови або аварійного відключення. Метрики витривалості можуть включати час відновлення системи після відмови, час відновлення даних, цілісність даних після відмови.

1.3 Опис існуючих моделей та засобів оцінки функціонування з використанням баз даних

Використання баз даних при оцінюванні РКС взагалі полягає у зборі та зберіганні даних про роботу різних компонентів системи та їх взаємодії. А також, використовуючи бази даних, ми можемо збирати та аналізувати дані про роботу системи в реальному часі, що дозволяє вчасно виявляти та усувати можливі проблеми, що допоможе нам збільшити ефективність системи, а також забезпечити надійність та стабільність її роботи. Наприклад, ми можемо використовувати бази даних для збору інформації про завантаженість різних компонентів системи, таких як:

- швидкість передачі даних;
- час відповіді на запити;
- пам'ять;
- мережевий трафік;
- використання ресурсів процесору;
- тощо.

Завдяки цієї інформації ми можемо виявляти проблемні місця в системі та у подальшому покращити її функціонування. Також можна використовувати бази даних для зберігання налаштувань системи, що дозволяє виконувати її автоматичне керування.

Усі ці дані можуть бути використані для створення багатофакторної моделі, яка дозволяє оцінити вплив різних факторів на показники функціонування системи. Окрім цього, у подальшому така модель може допомогти визначити, які компоненти системи будуть мати найбільший вплив на продуктивність системи, та які частини функціонування можуть бути покращені.

Крім того, бази даних можна використовувати для розробки алгоритмів машинного навчання, які дозволяють передбачити майбутні показники функціонування системи на основі зібраних даних. Це може допомогти виявити можливі проблеми та уникнути їх заздалегідь, а також вдосконалити стратегію управління системою.

Отже, якщо підвести проміжний підсумок, то загальне використання баз даних дозволяє нам покращити ефективність та надійність розподіленої комп'ютерної системи шляхом збору та аналізу даних про її роботу та використання цих даних для побудови моделей та алгоритмів оптимізації цієї системи. При створенні яких, можна використовувати методи математичного моделювання та статистичного аналізу, та зберігати проміжні дані цих алгоритмів та їх особливостей, а також результати їх роботи для подальшого аналізу. Що насамперед дозволяє визначити оптимальні параметри системи та підвищити її ефективність.

Існують багато моделей та засобів оцінки функціонування РКС. Але у межах дослідження ми будемо розглядати лише ті, які можуть більш ефективно використовувати бази даних у процесі своєї реалізації. Та будемо розглядати такі моделі та засоби оцінки функціонування РКС:

1) Модель мереж Петрі (Petri Nets).

У загальному вигляді модель мереж Петрі (Petri Nets) - це формальна математична модель, яка використовується для опису систем, що складаються з процесів, які взаємодіють один з одним. Ця модель складається з двох типів елементів: позицій та переходів. Позиції представляють стани системи, а переходи вказують на події, які можуть бути виконані для зміни стану системи. Де стани системи уявляють собою конфігурацію її компонентів та поточний стан виконання процесів, що забезпечують роботу цієї системи.

Застосування моделі мереж Петрі в РКС полягає в тому, що вона дозволяє моделювати процеси, які відбуваються в системі, а також визначати можливі стани системи в залежності від змінних параметрів. Для цього ми як

раз можемо використовувати бази даних, у яких будемо зберігати інформацію про стани системи, її параметри та результати обчислень. Також такий підхід забезпечує більш ефективну і точну оцінку функціонування системи, що дозволяє проводити аналіз та оптимізацію роботи системи, а також виявляти й усувати можливі неполадки.

2) Модель аналізу мереж NMA (Network Analysis Model).

Мережева модель NMA - це математична модель, яка дозволяє оцінити продуктивність і завантаженість РКС. Ця модель дозволяє описувати розподілені системи з декількома серверами, де заявки надходять в систему через вхідну точку та збираються у чергу для обробки. Ці заявки обслуговуються на різних серверах, які можуть працювати паралельно. Важливими параметрами, які ми враховуємо у моделі NMA, є:

- пропускна здатність;
- шляхи передачі даних;
- завантаженість мережевих компонентів;
- час обробки заявок на кожному сервері;
- час очікування в черзі;
- кількість заявок, які приходять в систему за одиницю часу.

Використовувати бази даних у моделі NMA ми можемо для того, щоб мати інформацію про завантаженість серверів та обсяги вхідних та вихідних потоків даних. Цю інформацію ми можемо використовувати для опису системи та обрахування метрик продуктивності, що дозволяє провести аналіз таких речей як:

- оцінка максимальної пропускної здатності системи;
- визначення середнього часу обробки заявок та часу очікування в черзі;
- аналіз завантаженості серверів та визначення затримок або обмежень, які уповільнюють весь процес роботи системи.

Таким чином, використання моделі NMA з базою даних дозволяє більш точно оцінити продуктивність та завантаженість РКС.

3) Модель Марківських процесів

Модель Марківських процесів - це математична модель, яка використовується для опису систем, що розвиваються в часі. Вона передбачає, що майбутній стан системи залежить лише від її поточного стану.

Для зберігання інформації про стани системи в різні моменти часу можемо використовувати бази даних. Також ми можемо збирати дані про часи безвідмовної роботи системи та часи відмов системи. Що у подальшому буде корисним для оцінки надійності системи. Використання баз даних у цьому випадку буде нам дуже корисним для знаходження ймовірності можливості переходу системи з одного стану в інші, а також для оцінки стабільності роботи системи у майбутньому.

Також ми можемо використовувати цю модель для аналізу динаміки взаємодії компонентів РКС. Ось, наприклад, ми маємо систему з багатою кількістю вузлів дані про стан яких, ми як раз будемо зберігати у базі даних, та у цьому випадку, ми можемо використовувати таку модель процесів для опису взаємодії між цими вузлами та прогнозування впливу зміни одного вузла на роботу системи в цілому.

Загалом, модель Марківських процесів та використання баз даних можуть забезпечити точніші та надійніші прогнози щодо роботи РКС, що буде зручно для подальшого її аналізу при створенні оптимізаційної моделі.

4) Модель аналізу даних реєстрації подій (Event Log Analysis Model).

Модель аналізу даних реєстрації подій - це методологія збору та аналізу даних, що виникають під час роботи РКС. Дані про усі процеси, що відбуваються в системі, реєструються та зберігаються в базі даних. За допомогою спеціальних інструментів, які можуть аналізувати дані з баз даних, ми зможемо проводити моніторинг та аналіз стану РКС, для того, щоб виявляти проблеми та шукати способи їх вирішення.

Основна ідея застосування цієї моделі полягає у тому, що кожна діяльність в системі може бути записана у вигляді параметрів, таких як:

- час події;
- тип події;
- відповідальний користувач;
- об'єм оброблюємих користувачем даних;
- тощо.

Насамперед, окрім перерахованих, є ще багато параметрів даних, що будуть корисними нам у межах дослідження даних. Збираючи які, ми можемо створити історію записів для збереження великих обсягів даних про події. Та у подальшому, це буде корисним для нас при здійсненні аналізу з метою підвищення ефективності та швидкості роботи системи. Ще ми можемо проводити аналіз вже саме подій в системі, що дозволяє використовувати її ресурси значно ефективніше. Взагалі таку модель було б ефективно використовувати, як додаткову, або у синергії з другими системами або засобами.

5) Модель аналізу черг з відмовами (Queuing Model).

Модель аналізу черг з відмовами (Queuing Model) це математична модель, яка використовується для моделювання та аналізу чергових систем у розподіленій комп'ютерній системі. Завдяки цієї моделі ми можемо оцінити продуктивність системи, враховуючи такі параметри як: часи обслуговування та завантаженість ресурсів.

Бази даних у цьому випадку, ми використовуємо для збору, зберігання та у подальшому для аналізу даних про черги та параметри обробки. Наприклад, ми можемо зберігати інформацію про кількість запитів, що надходять до черги, час очікування та час обслуговування. Або, наприклад, ми можемо збирати дані про завантаженість компонентів та системи з її ресурсами в цілому при виконанні чергових запитів. Та отримані дані зможемо використовувати для оптимізації нашої системи та виявлення можливих проблем.

1.4 Висновки до розділу 1

У розділі 1 було розглянуто основні поняття з приводу того, що взагалі таке розподілена комп'ютерна система (РКС). Також визначали найбільш важливі її особливості та розглянули існуючі типи таких систем, взявши декілька реальних прикладів для аналізу. Також виявили більш значні метрики, такі як: продуктивність, доступність, надійність, масштабованість та витривалість та як вони впливають на РКС. Знаючи усі ці особливості, ми можемо ефективніше використовувати ресурси, забезпечувати взаємодію компонентів, неперервну роботу та масштабувати систему для зростаючих потреб. Також завдяки існуючим моделям та засобам оцінки РКС з використанням баз даних, ми упевнились, що використання баз даних у процесі оцінювання (РКС) дозволяє збирати та зберігати дані про роботу компонентів системи та їх взаємодію, таких як завантаженість компонентів, швидкість передачі даних, час відповіді на запити, використання ресурсів процесору та пам'яті, мережевий трафік та інші важливі метри. Окрім цього, ми можемо проводити аналіз цих даних в реальному часі, що дозволяє виявляти потенційні проблеми та усувати їх, сприяючи підвищенню ефективності та надійності системи.

РОЗДІЛ 2. АНАЛІЗ МОДЕЛЕЙ ОПТИМІЗАЦІЇ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Розгляд основних підходів до оптимізації

Оптимізація показників функціонування розподіленої комп'ютерної системи є важливим аспектом дослідження в галузі комп'ютерних наук. У межах цього дослідження розглядаються різні підходи та методи для покращення продуктивності, ефективності та надійності розподіленої системи. Основні підходи включають в себе:

- розподілене планування ресурсів;
- оптимізацію мережевої комунікації;
- масштабування;
- оптимізацію алгоритмів та обробки даних;
- використання кешування;
- реплікація даних.

Усі ці підходи допомагають підвищити продуктивність та ефективність розподіленої комп'ютерної системи, що має велике значення для багатьох сучасних додатків та сервісів, які базуються на таких системах. Можна виділити таку низку основних підходів, щодо оптимізації показників нашої РКС:

1) Розподілене планування ресурсів.

Цей підхід включає розподіл завдань та ресурсів між вузлами розподіленої системи таким чином, щоб досягти оптимального використання ресурсів і покращити продуктивність. Для цього можуть застосовуватися алгоритми планування, які враховують особливості системи та вимоги до завдань. Серед яскравих прикладів сучасного застосування можна виділити

Google Kubernetes Engine (GKE). Це рішення від Google Cloud для керування контейнерами. Воно забезпечує автоматичне розподілення ресурсів та планування завдань між вузлами кластера. Також гарним прикладом є Apache Mesos. Що уявляє собою відкрите програмне забезпечення для керування ресурсами в розподілених системах. Воно дозволяє ефективно планувати та розподіляти ресурси між різними задачами.

2) Оптимізація мережевої комунікації.

Для забезпечення ефективної комунікації між вузлами розподіленої системи важливо визначити оптимальні шляхи передачі даних та мінімізувати затримки. Це може включати вибір оптимального протоколу передачі, оптимізацію маршрутизації та використання методів компресії та кешування для зменшення навантаження на мережу. Тут представником є Content Delivery Network (CDN): CDN, наприклад, Cloudflare або Akamai, що використовується для оптимізації доставки контенту. Вони мають розподілені сервери по всьому світу, що дозволяє зменшити затримки та покращити швидкість доставки контенту. Також Software-Defined Networking (SDN) є теж яскравим представником оптимізації мережевої комунікації. SDN технологія, як Cisco ACI або VMware NSX, надає централізоване керування та оптимізацію мережевого трафіку в розподіленій системі, забезпечуючи ефективну комунікацію.

3) Масштабування.

Один з ключових аспектів розподіленої системи - її здатність масштабуватися залежно від зростаючих вимог. Це може включати горизонтальне масштабування (додавання нових вузлів у систему) та вертикальне масштабування (покращення апаратних характеристик вузлів). Оптимізація масштабування полягає у забезпеченні ефективного розподілу завдань та ресурсів при збільшенні обсягу роботи. Серед прикладів, більш яскравими є Amazon Web Services (AWS) Elastic Compute Cloud (EC2). Який дозволяє масштабувати обчислювальні ресурси у хмарному середовищі. Та

Apache Hadoop, який є фреймворком для обробки великих обсягів даних у розподіленому середовищі.

4) Оптимізація алгоритмів та обробки даних.

Ефективні алгоритми та методи обробки даних є ключовими складовими розподіленої системи. Покращення алгоритмів та оптимізація обробки даних можуть включати використання паралельних обчислень, розподілених алгоритмів, кешування та інших методів для підвищення швидкодії та зниження навантаження на систему. Приклади: Apache Spark, який є фреймворком, що використовує паралельні обчислення для розподіленої обробки даних в реальному часі. А також модель програмування MapReduce.

5) Використання кешування та реплікації даних.

Кешування та реплікація даних можуть бути використані для поліпшення доступу до даних в розподіленій системі. Кешування дозволяє зберігати часто використовувані дані на швидкодіючих пристроях, що знижує час доступу. Реплікація даних дозволяє мати копії даних на різних вузлах системи, що підвищує доступність та надійність. Гарним прикладом є система кешування даних Memcached, яка широко використовується для покращення продуктивності веб-додатків. Вона зберігає часто використовувані дані у швидкодіючому кеші, що зменшує час доступу до даних. А також MongoDB, яка уявляє собою розподілену базу даних, що підтримує реплікацію даних між різними серверами.

6) Керування завантаженістю ресурсів.

Це така техніка розподілу навантаження між різними вузлами або серверами у розподіленій системі. Вона дозволяє рівномірно розподіляти запити та задачі, забезпечуючи оптимальне використання ресурсів та покращення продуктивності. Гарні приклади що використовують цей підхід у своїй праці:

- Веб-сервер Nginx, який використовується для балансування навантаження між різними серверами, розподіляючи запити в залежності від завантаженості та доступності ресурсів.
- Apache HTTP Server, який може бути налаштований з модулем `mod_proxy_balancer` для балансування навантаження між вузлами у розподіленій системі.

2.2 Огляд існуючих моделей оптимізації

На попередньому кроці ми розглянули загальні підходи, щодо оптимізації показників розподіленої комп'ютерної системи. Зараз же проводимо огляд вже існуючих моделей з використанням цих підходів, та приклади їх застосування в сучасному світі:

1) Модель оптимізації завантаження (Load Balancing).

Ця модель ставить за мету розподілити навантаження між вузлами або серверами баз даних таким чином, щоб досягти максимальної продуктивності системи. Вона використовує алгоритми для прийняття рішень щодо маршрутизації запитів та балансування навантаження між вузлами. Приклади: Google MapReduce, а також відкритий фреймворк Apache Hadoop, який також використовує модель MapReduce.

2) Модель оптимізації індексів (Index Optimization).

Ця модель фокусується на покращенні продуктивності запитів до баз даних шляхом ефективного використання індексів. Вона враховує структуру бази даних, типи запитів та характеристики даних для вибору найоптимальніших індексів, що забезпечують швидкий доступ до даних. Серед прикладів, можна виділити Oracle Database. Ця система керування базами даних надає можливості оптимізації індексів для покращення продуктивності запитів. Вона використовує різні алгоритми і структури даних

для ефективного доступу до інформації, що дозволяє прискорити виконання запитів. А також розподілена система Elasticsearch для пошуку та аналізу даних з використанням власної моделі оптимізації індексів.

3) Модель оптимізації запитів (Query Optimization).

Ця модель спрямована на покращення продуктивності запитів до баз даних шляхом вибору оптимального плану виконання запиту. Вона враховує багато параметрів, таких як:

- структура бази даних;
- наявність індексів;
- статистика даних;
- оптимізаційні алгоритми.

Існуючими ж прикладами використання цієї моделі є MySQL Query Optimizer. Це компонент системи управління базами даних MySQL, що використовує різні методи та алгоритми для оптимізації запитів. Він аналізує структуру запитів та статистичні дані про дані для побудови оптимального запиту. Аналогом в Microsoft SQL Server є Query Optimizer.

4) Модель оптимізації розподіленого зберігання даних (Distributed Data Storage Optimization).

Ця модель ставить за мету оптимізувати розподілене зберігання даних у РКС. Вона враховує розташування даних, реплікацію даних, доступність та надійність системи. Таку модель використовує оптимізована для швидкого запису та читання великих обсягів даних Apache Cassandra (NoSQL), яка розподіляє дані по вузлах кластера, забезпечуючи високу доступність та надійність. А також хмарна система Amazon S3 (Simple Storage Service), який розподіляє дані по різних регіонах та зонах доступності, що дозволяє оптимально зберігати та доступатися до даних.

5) Модель оптимізації масштабування (Scalability Optimization).

Для забезпечення ефективного розширення нашої системи, ця модель враховує такі фактори, як:

- кількість вузлів;
- навантаження;
- розподіл даних;
- комунікацію між вузлами.

При чому така модель може застосовувати одразу кілька методів розширення. Це може бути горизонтальне масштабування, що передбачає збільшення обсягу ресурсів шляхом додавання нових вузлів або серверів. Або ж використання автоматичне масштабування, принцип якого полягає в автоматичному реагуванні системи на зміни в обсягах даних або навантаженні. Яскравим прикладом є Google File System (GFS). Також Twitter використовує цю модель для зберігання та обробки даних, що дозволяє масштабувати свою інфраструктуру для задоволення потреб користувачів у реальному часі. Слід ще відмітити систему Facebook's Haystack для зберігання зображень мільярдів зображень з високою швидкістю доступу.

2.3 Порівняння різних методів оптимізації

Зробимо невеликий проміжний підсумок порівнюючи методи та підходи щодо оптимізації РКС на основі використаної інформації у попередніх підрозділах. Будемо порівнювати ці моделі та методи окремо за такими параметрами:

- умови завантаженості;
- обсяг даних;
- складність задачі.

1) Модель оптимізації завантаження (Load Balancing).

- a) Умови завантаженості: Ця модель ефективно працює при високій завантаженості системи, коли потрібно розподілити завдання між різними вузлами. Вона допомагає підтримувати рівномірний розподіл навантаження та забезпечує високу доступність системи.
- b) Обсяг даних: Не має прямого впливу на модель оптимізації завантаження, тому що вона фокусується на розподілі завдань та балансуванні навантаження між вузлами.
- c) Складність задачі: Ця модель є простою і ефективною для задач розподілу завдань без складних операцій аналізу запитів або індексування.

Переваги:

- розподіл завдань та навантаження між вузлами;
- висока доступність системи;
- простота в реалізації.

Мінуси:

- не вирішує проблеми оптимізації запитів або індексів;
- може бути неефективною у випадку низького навантаження системи.

2) Модель оптимізації індексів (Index Optimization).

- a) Умови завантаженості: Ця модель особливо корисна, коли навантаження системи високе або коли виникають запити, які часто використовуються, завдяки ефективному використанню індексів.
- b) Обсяг даних: Чим більший обсяг даних, тим важче стає зберігати та підтримувати індекси, тому оптимізація індексів стає більш складною.
- c) Складність задачі: Якщо запити вимагають складних операцій та фільтрації даних, то така модель оптимізації індексів може бути корисною.

Переваги:

- покращення продуктивності запитів за рахунок використання індексів;
- скорочення часу виконання запитів;
- ефективне фільтрування та сортування даних.

Мінуси:

- вимагає підтримки та оновлення індексів;
- може займати багато місця у випадку великого обсягу даних.

3) Модель оптимізації запитів (Query Optimization).

- a) Умови завантаженості: Модель буде ефективною при складних запитах або запитах, які вимагають аналізу та оптимізації. Вона допомагає знизити час виконання запитів та оптимізувати ресурси системи.
- b) Обсяг даних: При збільшенні обсягу даних, можуть потребуватись додаткові операції та розрахунки для ефективного виконання запитів.
- c) Складність задачі: Якщо запити вимагають складних операцій аналізу, з'єднань таблиць, групування або агрегування даних, тоді у такому випадку ця модель може бути однією з найкращих.

Переваги:

- зниження часу виконання запитів та оптимізація ресурсів;
- покращення продуктивності складних запитів;
- вибір оптимального плану виконання запиту.

Мінуси:

- вимагає аналізу та оптимізації запитів;
- може бути складною для реалізації у великих системах.

4) Модель оптимізації розподіленого зберігання даних (Distributed Data Storage Optimization).

- a) Умови завантаженості: модель корисна при використанні з розподіленою базою даних або обчисленнями.
- b) Обсяг даних: збільшення обсягу даних впливає на розподіл даних та потребу в масштабованості, та може вимагати розподілення на багато вузлів та оптимізацію передачі даних.

- с) Складність задачі: Допомагає підтримувати ефективне масштабування та реплікацію даних в умовах використання розподілених обчислень та зберігання даних.

Переваги:

- ефективне розподілення даних у розподілених системах;
- оптимальне використання ресурсів системи;
- можливість масштабування та реплікації даних.

Мінуси:

- вимагає додаткових налаштувань та управління розподіленими даними;
- складність реалізації та підтримки.

5) Модель оптимізації масштабування (Scalability Optimization).

- а) Умови завантаженості: Модель ефективна в ситуаціях, коли потрібно масштабувати систему, щоб впоратися з ростом навантаження.
- б) Обсяг даних: при збільшенні обсягу даних, збільшується потреба в масштабуванні системи, та може вимагати більшої потужності обчислення та зберігання для ефективної роботи системи.
- с) Складність задачі: Модель оптимізації масштабування допомагає забезпечити гнучкість системи для росту навантаження та розширення, включаючи до себе горизонтальне та вертикальне масштабування, а також оптимізацію розподілу ресурсів.

Переваги:

- забезпечення гнучкості та масштабованості системи;
- можливість адаптуватися до зростаючого навантаження;
- оптимальне використання ресурсів.

Мінуси:

- вимагає налагодження та налаштування для масштабування;
- додаткові витрати на розширення ресурсів.

2.4 Аналіз механізмів управління ресурсами РКС

Управління ресурсами у розподіленій обчислювальній системі (РКС) є ключовим аспектом, який вимагає ретельного планування та оптимізації для досягнення максимальної продуктивності та ефективності. Існують різні механізми управління ресурсами, які дозволяють ефективно використовувати обчислювальні ресурси, мережеві з'єднання та інші компоненти РКС. Розглянемо та проаналізуємо п'ять найбільш відомих та ефективних:

1) Механізми моніторингу ресурсів.

Завдяки таким механізмам, ми можемо відстежувати використання ресурсів у РКС, такі як обсяги пам'яті, потужність обчислювальних вузлів, пропускна здатність мережі тощо. Прикладом таких механізмів є системи моніторингу, наприклад, Prometheus або Nagios.

2) Механізми планування ресурсів.

Використовуються для ефективного розподілу ресурсів між різними завданнями або вузлами у РКС. Вони можуть включати алгоритми планування, що забезпечують рівномірне розподілення ресурсів та забезпечують виконання вимог продуктивності. Прикладом таких механізмів є Kubernetes, який забезпечує автоматичне масштабування та планування ресурсів у контейнерних середовищах.

3) Механізми оптимізації завантаження.

Основною метою тут є розподілення навантаження між різними вузлами або серверами баз даних з метою досягнення максимальної продуктивності системи. Вони можуть використовувати алгоритми балансування навантаження та маршрутизації запитів. Прикладами таких механізмів є Apache Hadoop та Google MapReduce.

4) Механізми оптимізації масштабування.

Ці механізми дозволяють системі ефективно розширюватися залежно від збільшення обсягу ресурсів або потреб користувачів. Вони можуть включати горизонтальне масштабування, додавання нових вузлів або серверів,

а також автоматичне розподілення навантаження. Прикладами таких механізмів є Apache Spark та Amazon EC2 Auto Scaling.

5) Механізми оптимізації енергоспоживання.

Такий підхід спрямований на зниження споживання енергії в РКС шляхом ефективного використання ресурсів та управління енергозалежними компонентами, такими як процесори, пам'ять, диски тощо. Вони можуть включати розумне керування енергією, призначене для оптимізації режимів споживання енергії та резервування ресурсів. Прикладом такого механізму є PowerNap для Linux.

2.5 Висновки до розділу 2

У розділі 2 ми спочатку розглянули існуючі загальні підходи щодо оптимізації. Були розглянуті такі основні підходи, як розподілене планування ресурсів, оптимізація мережевої комунікації, масштабування, оптимізація алгоритмів та обробки даних, використання кешування та реплікація даних, а також керування завантаженістю ресурсів, допомагають підвищити продуктивність та ефективність розподіленої системи. Після чого ми розглянули вже існуючі моделі, усі з яких напряду або ж опосередковано використовують в основі розглянуті раніше підходи щодо оптимізації. Окрім цього, загальний огляд таких моделей оптимізації РКС показує, що існують різні підходи та приклади їх застосування в сучасному світі. Моделі, такі як оптимізація завантаження, оптимізація індексів, оптимізація запитів, оптимізація розподіленого зберігання даних та оптимізація масштабування, допомагають досягти максимальної продуктивності та ефективності розподіленої системи. Ці моделі використовуються в відомих нам різних сучасних системах, таких як Google MapReduce, Oracle Database, Apache

Cassandra, Amazon S3, Google File System, Twitter і Facebook's Haystack, для оптимізації різних аспектів системи та забезпечення надійності та швидкодії.

Після чого ми порівняли усі описані нами методи оптимізації, за такими параметрами: умови завантаженості, обсяг даних, складність задачі, а також виділили більш вагомні переваги та недоліки цих моделей. Окрім цього, за результатом цього міні дослідження можемо зробити такі висновки: модель оптимізації завантаження (Load Balancing) ефективна при високій завантаженості системи та розподілі завдань між вузлами, забезпечуючи високу доступність. Модель оптимізації запитів (Query Optimization) покращує продуктивність складних запитів та оптимізує ресурси, але вимагає аналізу та оптимізації запитів. Модель оптимізації масштабування (Scalability Optimization) забезпечує гнучкість системи для росту навантаження та оптимізацію розподілу ресурсів, але вимагає налагодження та налаштування для масштабування.

Також ми ще визначили що управління ресурсами у РКС включає в себе механізми: моніторингу ресурсів, планування ресурсів, оптимізації завантаження, масштабування та енергоспоживання. Ці механізми в свою чергу дозволяють ефективно використовувати ресурси, розподіляти завдання, балансувати навантаження, масштабувати систему та знижувати споживання енергії. Наприклад, такі відомі системи, як Kubernetes, Apache Hadoop, Apache Spark та PowerNap є демонстрацією таких механізмів, які допомагають досягти максимальної продуктивності та ефективності у РКС.

РОЗДІЛ 3. ОПТИМІЗАЦІЙНА БАГАТОФАКТОРНА МОДЕЛЬ ОЦІНКИ ПОКАЗНИКІВ РКС НА ОСНОВІ МЕЖЦЕНТРИЧНОГО ПІДХОДУ

3.1 Формалізація загального вигляду РКС

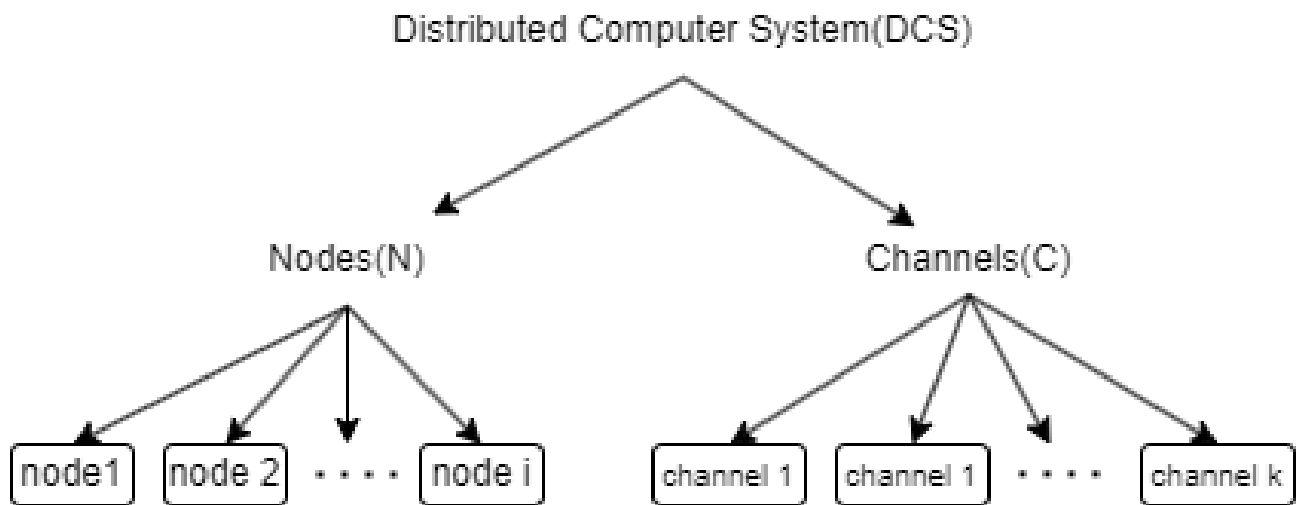


Рис. 3.1. Загальний вигляд параметрів РКС.

Будемо використовувати звичайну РКС у вигляді як на рисунку 3.1. Звідси маємо, що наша розподілена комп'ютерна системи (РКС), складається в двох параметрів, та може буде уявлена у вигляді наступної залежності:

$$Distributed\ Computer\ System = DCS(N, C) \quad (3.1)$$

де N – це параметр, який відповідає за вузли системи, та може бути уявлений у вигляді наступної залежності:

$$Nodes = N(node_1, node_2, node_3, \dots, node_i) \quad (3.2)$$

В свою чергу кожен з вузлів складається з параметрів його обладнання:

- частота процесору (Core Frequency);
- кількість ядер (Core Quantity);
- потужність (Power);
- обсяг пам'яті (Memory).

А також має таку залежність, відповідно до його параметрів та структури, як на рисунку 3.2:

$$node = n(CF, CQ, P, M) \quad (3.3)$$

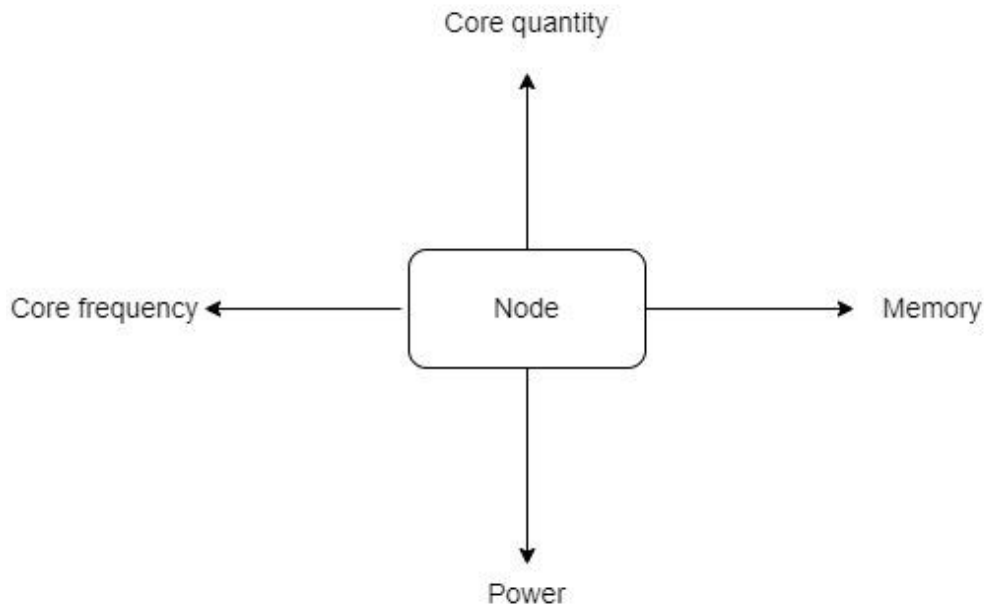


Рис. 3.2. Вигляд вузла та його параметрів.

де C – це параметр, який відповідає за канали системи, та може бути уявлений у вигляді наступної залежності:

$$Channels = C(channel_1, channel_2, channel_3, \dots, channel_k) \quad (3.4)$$

В свою чергу кожен з каналів складається з параметрів його обладнання:

- затримка (Delay);
- пропускна здатність (Bandwidth);
- шум (Noise).

А також таку залежність, відповідно до його параметрів та структури, як на рисунку 3.3:

$$channel = ch(F, BW) \quad (3.5)$$

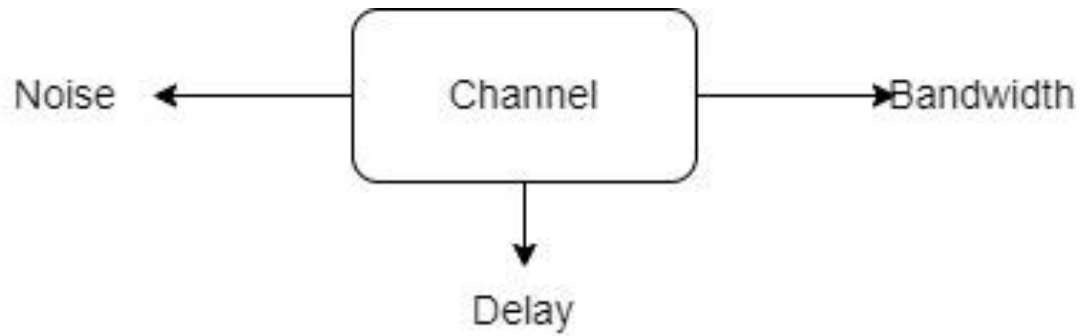


Рис. 3.3. Вигляд каналу та його параметрів.

Оскільки канали системи відносяться то гетерогенних, то нам необхідно враховувати затримку передачі даних, яка з'являється через час необхідний для підготовки та перетворення даних, перед їх отриманням безпосередньо вузлами системи. Формула кількості каналів в залежності від кількості обчислювальних вузлів виглядає так:

$$c = \frac{n(n-1)}{4} \quad (3.6)$$

Продуктивність системи будемо розглядати та обчислювати за формулою:

$$DCS_{speed} = \frac{t_s}{t_{DCS}} \quad (3.7)$$

$$\text{Де } t_s = \frac{diff}{\omega_{avg}}, \quad (3.8)$$

$$\omega_{avg} = \frac{\sum_{i=1}^n \omega_i}{n} \quad (3.9)$$

$$\omega_i = \frac{CQ_i * CF_i + M_i}{P_i} \quad (3.10)$$

$$t_{DCS} = \frac{diff}{n\omega_{avg}} \cdot \lambda^2 + 0,5 * c * delay \quad (3.11)$$

де CQ_i, CF_i, M_i, P_i , відомі параметри для розрахунку потужності ω_i для n_i ;

n – вузли системи;

c – кількості каналів в залежності від кількості обчислювальних вузлів;

$delay$ – деякий час для пересилання даних;

ω_{avg} – середня потужність вузлів системи;

$diff$ – складність обсягу даних для обробки при подані на вхід;

λ – коефіцієнт передачі даних;

MSP - максимальна пропускна здатність, яку обраховуємо далі.

Також нам необхідно визначити нормовану швидкість передачі даних. Так як при збільшені обсягу даних потрібно буде збільшувати максимальну пропускну здатність каналу зв'язку. Окрім цього, ми можемо розрахувати швидкість передачі даних, використовуючи теорему теорема дискретизації, де максимальна пропускна здатність (MSP) каналу зв'язку в бітах на секунду (bps) обмежується шириною смуги (BW) каналу і рівнем шуму (N) на каналі за допомогою наступної формули:

$$MSP = BW * \log_2(1 + S/N) \quad (3.12)$$

де B - ширина смуги каналу в герцах (Hz),

S - сигнал-шум (signal-to-noise ratio) на каналі,

N - рівень шуму на каналі.

У нашій системі однією з головних оцінок будемо враховувати рівень захищеності. Будемо вважати що кожні n секунд у нашу систему будуть намуруватись втрутитись з метою викрадення або пошкодження даних. А також визначимо захищеність системи у вигляді:

$$\text{Security} = S (Am, Pr, n) \quad (3.13)$$

де Am – обсяг наявних даних поданих та оброблюємих у системі;

Pr – процент даних, що були вдало опрацьовані без втрати пакетів;

n – період циклу у секундах.

Також будемо враховувати рівень надійності системи, який насамперед буде складовою завантаженості на її компоненти відповідно до довгостроковості їх праці над завданням. Який представляємо у вигляді:

$$\text{Reliability} = R (CL, RL, \text{Time}) \quad (3.14)$$

де CL - рівень завантаженості каналів системи;

RL - рівень завантаженості ресурсів системи;

Time – прогнозований час виконання.

Таким чином маємо математичну модель нашої системи у вигляді:

$$\text{Parameters} = \text{PDCS} (DCS_{speed}, R (CL, RL, \text{Time}), MSP (BW, S, N), S (Am, Pr, n)) \quad (3.15)$$

3.2 Опис архітектури тестової системи

Будемо розглядати нашу модель розподіленої комп'ютерної системи з точки зору оптимізації управління ресурсами на основі межцентричного підходу. Такий підхід є дуже ефективним для оптимізації потрібних ресурсів системи, а також дозволяє дуже ефективно розподіляти завдання рівномірно між компонентами системи. Наша РКС при такому підході буде мати ієрархічний вигляд, де на першому рівні в нас буде клієнт, який подає задачу, а на останньому рівні ми будемо отримувати результати виконання: такі як час потрачений на опрацювання, завантаженість ресурсів вузлів, статус виконання, тощо. Усі вони допоможуть нам для оптимізації нашої системи та її ресурсів згідно до задач, які подаються. Загальний вигляд системи буде наступним:

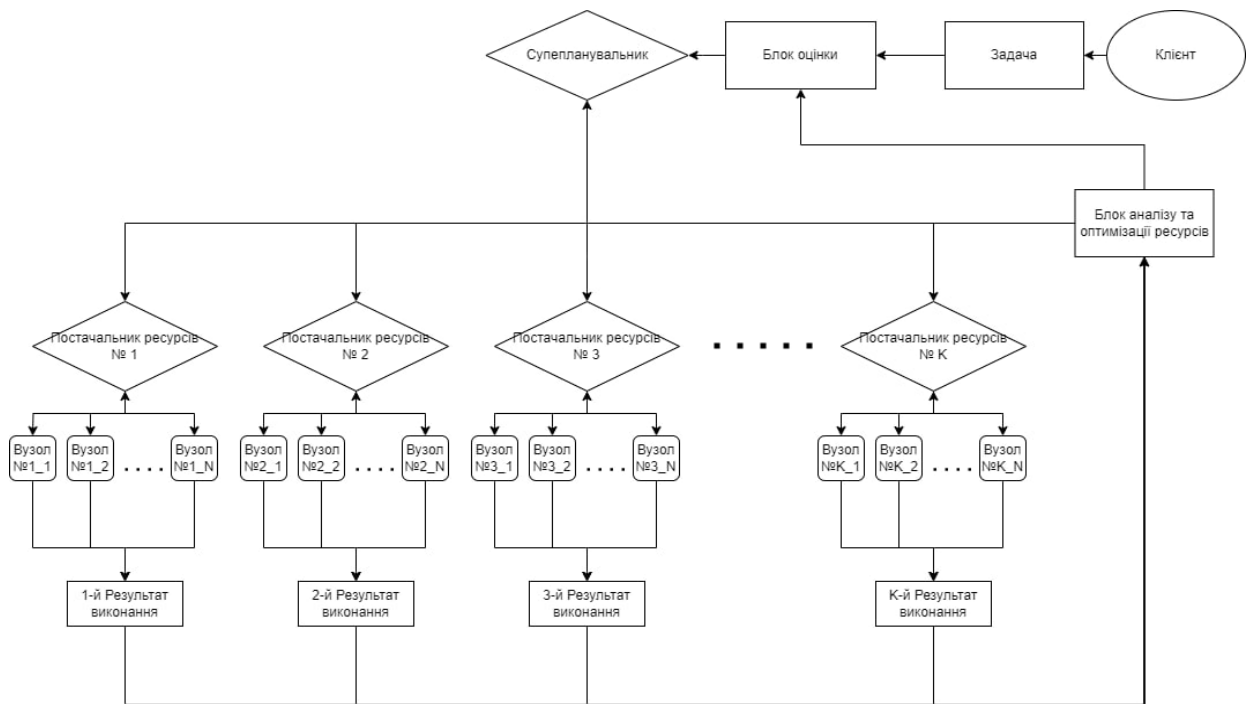


Рис. 3.4. Блок схема вигляду РКС на основі межцентричного підходу.

Як можемо побачити з рисунка 3.4, маємо такі ієрархічні рівні:

- 1) рівень обробки задач;
- 2) рівень Планувальника та Постачальників ресурсів;
- 3) рівень обчислень;
- 4) рівень обробки та аналізу результатів.

Також слід зазначити, що так як мета дослідження полягає в оптимізації ресурсів РКС, то ми лише імітуємо її роботу, з цілю знаходження ефективного використання ресурсів для різного обсягу задач та часу їх виконання, та реальних задач в систему на подається, а отримані результати використовуються тільки для аналізу. Далі визначимо, що уявляє собою кожен блок окремо. Починається все з блоку на якому ми отримуємо задачу та одразу її аналізуємо. Так як в нас є базові параметри ресурсів системи та її можливостей, то ми будемо використовувати статистичну модель, для обрахування задачі на 5 рівнів складності: незначна, проста, середня, важка, дуже важка. Після чого задачі на основі попереднього розподілу почергово

подаються до нашого блоку суперпланувальника, який вже розподіляє задачі, по блокам з постачальниками ресурсів в залежності від:

- 1) Можливостей постачальника та загальної потужності ресурсів його вузлів.
- 2) Наявності в постачальника задачі, яка ще виконується.
- 3) Ефективність виконання даної задачі, даним вузлом.

Далі переходимо до блоків постачальників ресурсів, які уявляють собою підпорядковані системи, кожна з яких працює паралельно з розподіленням ресурсів по декількома вузлам.

Після розподілення задач по постачальникам, вони розподіляють їх виконання рівномірно між вузлами, які насамперед також працюють паралельно. Кожен вузол в свою чергу має свої параметри, на основі яких обчислюється загальна потужність вузла щодо обчислень конкретного обсягу даних за одиницю часу. Спочатку ми задаємо ці параметри в системі власноруч, а далі корегуємось даними з блоку аналізу, який отримує такі дані:

- час виконання задачі;
- завантаженість каналів;
- складність обсягу даних;
- завантаженість ресурсів вузла;
- інформація щодо метода захисту даних.

Таким чином ми отримуємо нові дані оцінок параметрів системи, та можемо прийняти міри щодо можливого масштабування системи, або ж навпаки зменшення кількості вузлів, за їх необхідності, або обидва випадки одночасно, коли наприклад у систему поступають задачі лише перших трьох рівнів складності, та обслуговування постачальників ресурсів с високим рівнем потужності та його вузлів для обчислень не буде необхідним, а доцільніше буде створити на їх основі два нових постачальники для оптимізації часу виконання, або ж розподілити його вузли до інших постачальників.

3.3 Програмна реалізація

Модель оптимізації управління ресурсами на основі її оцінок було створено за допомогою мови програмування Python версії 3.9. Використані бібліотеки для реалізації програмного продукту:

1. PyQt5 та PyQt-tools – ці дві бібліотеки були необхідні для створення інтерфейсу. PyQt є набором зв'язків Python для Qt, який надає можливість створення графічного інтерфейсу користувача (GUI) за допомогою функцій і класів. Використовуючи PyQt, було створено вікна, кнопки, тексти, повзунки, графіки та багато іншого. PyQt-tools, з іншого боку, є набором інструментів, які надають додаткові можливості для розробки додатків. Також, PyQt-tools включає в себе різноманітні утиліти, які спрощують процес розробки, відлагодження та налагодження PyQt додатків.
2. networkx є потужним інструментом для роботи з мережевими структурами в РКС мережових системах. Завдяки цієї бібліотеки ми можемо зручно моделювати, аналізувати та візуалізувати мережеві дані, що допомагає розуміти взаємозв'язки та властивості системи. Ми можемо відображати графи у вигляді графічних об'єктів, змінювати їх кольори, розміри та стилі відображення. Також можливо відображати ваги ребр та атрибути вузлів. У межах дослідження, ця бібліотека була використана для моделювання вигляду систем з метою зручнішого аналізу її компонентів.
3. numpy та math – необхідні бібліотеки для обчислень. Включають у себе функції тригонометричних, логарифмічних, експоненціальних та інших математичних операцій. А також дозволяють працювати з масивами, включаючи математичні операції, операції лінійної алгебри, трансформації даних та статистичний аналіз.
4. matplotlib - це основна бібліотека для створення графіків та візуалізації даних у Python. Вона надає широкий спектр функцій та класів для створення різних типів графіків, таких як лінійні графіки, стовпчикові діаграми, кругові діаграми, діаграми розсіювання, теплові карти та інші. А також її модуль matplotlib.colors, який містить різноманітні класи та функції, пов'язані з кольорами в бібліотеці matplotlib.

Програмний продукт було вирішено візуалізувати за допомогою функціонала бібліотек PyQt5 Та PyQt-tools. Це відкриває нам доступ до асортименту з багатьох віджетів, кнопок, контекстних меню, блоків вибору параметрів тощо. При завантаженні програми перед нами впливає головне вікно: “Project X”, його можна побачити на рисунку 3.5.

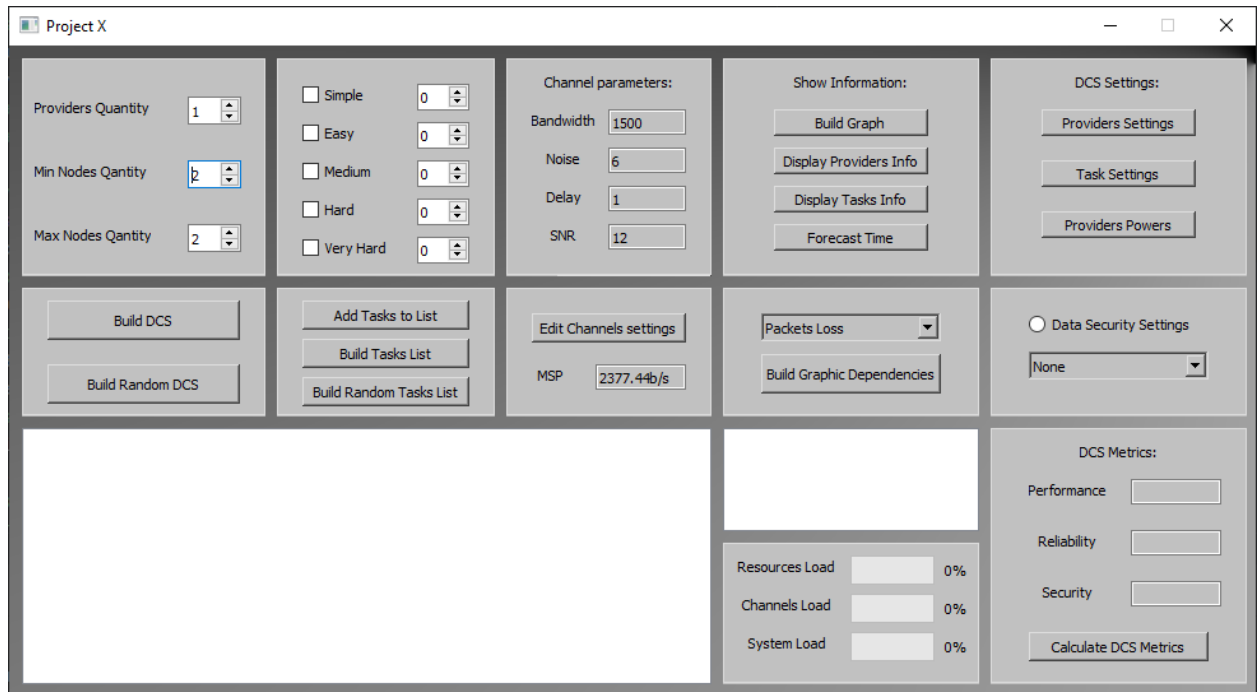


Рис. 3.5. Головне вікно програми.

Тут ми можемо побачити багато різних кнопок та інших елементів інтерфейсу. Позначимо який елемент за що відповідає:

- Кнопки “Build Random DCS” та “Build DCS” відповідають за створення Розподіленої комп’ютерної системи з випадковими параметрами або ж з нуля з використанням параметрів, що ми можемо змінити вище у відповідних SpinBox “Providers Quantity”, який відповідає за кількість постачальників ресурсів у системі, та “Max_ Nodes Quantity” та “Min Nodes Quantity”, які відповідають максимальній та мінімальній кількості вузлів у кожного постачальника відповідно.
- Кнопки “Add Tasks to List”, “Build Tasks List” та “Build Random Tasks List” відповідають за створення списку запитів випадково, або ж використовуючи параметри зверху.

- Кнопка “Edit Channels settings” змінює початкові параметри каналів системи.
- Кнопка “Build Graph” будує та моделює на основі згенерованої мережі постачальників та їх вузлів. Після чого виводить їх у вигляду графу, з відповідними кольорами у вузлів щодо їх потужності.
- Кнопки “Display Providers Info” та “Display Tasks Info” виводять інформацію про постачальників та їх вузли або про завдання відповідно.
- Кнопка “Forecast Time” прогнозує час на виконання змодельованою системою згенерованого списку завдань-запитів. А також виводить інформацію про завантаженість системи та її компонентів на три ProgressBar знизу.
- Кнопка “Provider Settings” дозволяє нам змінювати конфігурації компонентів на рівні постачальників. При виборі постачальника ми зможемо його видалити, або додати нового, або змінити його параметри та вузли для моделювання певних сценаріїв. Що насамперед відкриває нам багато можливостей щодо масштабування системи.
- Кнопка “Tasks Settings” дозволяє нам редагувати список запитів видаляючи їх, або змінюючи їх параметри.
- Кнопка “Providers Powers” дозволяє напряду встановлювати ліміти на використання постачальниками їх ресурсів, збільшуючи їх швидкість, або ж навпаки зменшуючи.
- Перемикач “Data Security Settings”, та відповідає comboBox за вибір методу захисту даних зі списку нижче.
- Кнопка “Build Graphic Dependencies” відповідає за побудову графіку відношень параметрів та результатів.
- Кнопка “Calculate DCS Metrics” відповідає за розрахунок оцінок показників системи.

3.4 Дослідження впливу параметрів на оцінки розподіленої комп'ютерної системи (PKC)

Для того щоб дослідити вплив параметрів на оцінки системи спочатку, згенеруємо випадковим чином нашу систему, та побудуємо та виведемо її граф. Для побудови випадкової системи використовуємо параметри “Providers Quantity” = 3, а їх кількість у межах від “Min Nodes Quantity” = 2 до “Max Nodes Quantity” = 5 та натискаємо на “Build DCS”. Далі натискаємо на “Build Graph” та в результаті отримуємо таку систему у вигляді графу на рисунку 3.6.

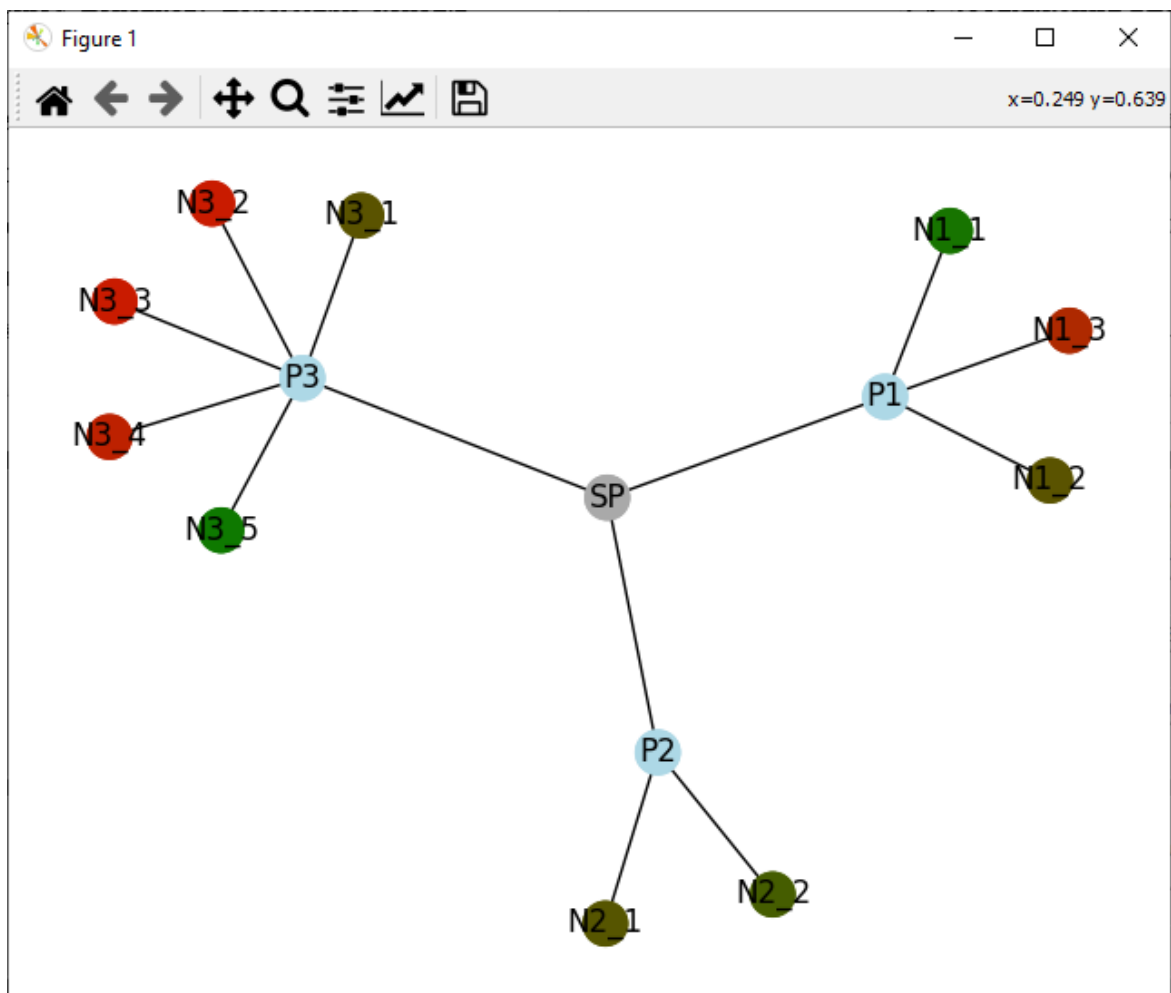


Рис. 3.6. Згенерована з параметрами (P:3, Min:2, Max:5) PKC у вигляді графу.

Далі для того щоб запустити процес обчислень, згенеруємо для нашої системи 5 найпростіших, 5 простих та 5 середніх задач, та виведемо

інформацію про ці завдання та про постачальників з їх ресурсами, за допомогою кнопок “Display Providers Info” та “Display Tasks Info”.

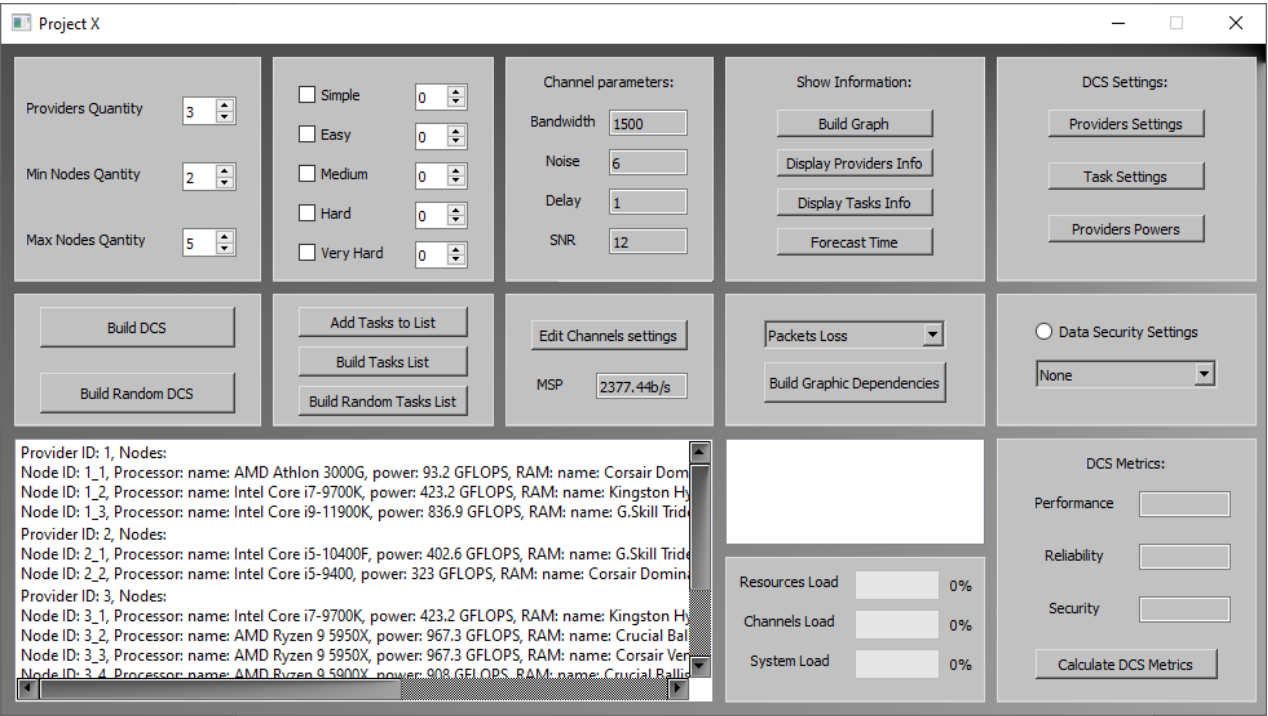


Рис. 3.7. Головне вікно з інформацією про постачальників та їх ресурси.

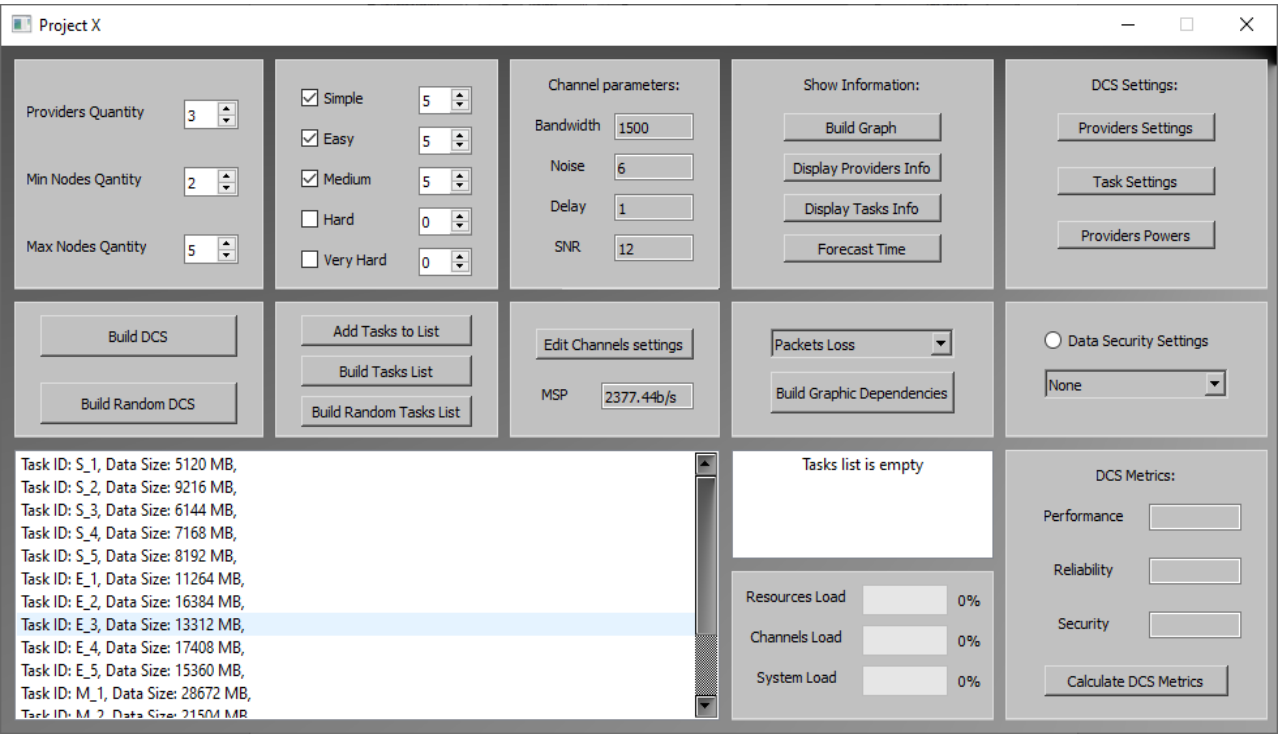


Рис. 3.8. Головне вікно з виводом інформації про запити та їх обсяг.

Результати виконання можемо побачити на рисунку 3.7 та рисунку 3.8. Після чого обчислимо середній час їх виконання та завантаженість системи, натиснувши на кнопку “Forecast Time”. А також отримаємо оцінки системи на основі цих даних натиснувши кнопку “Calculate DCS Metrics”:

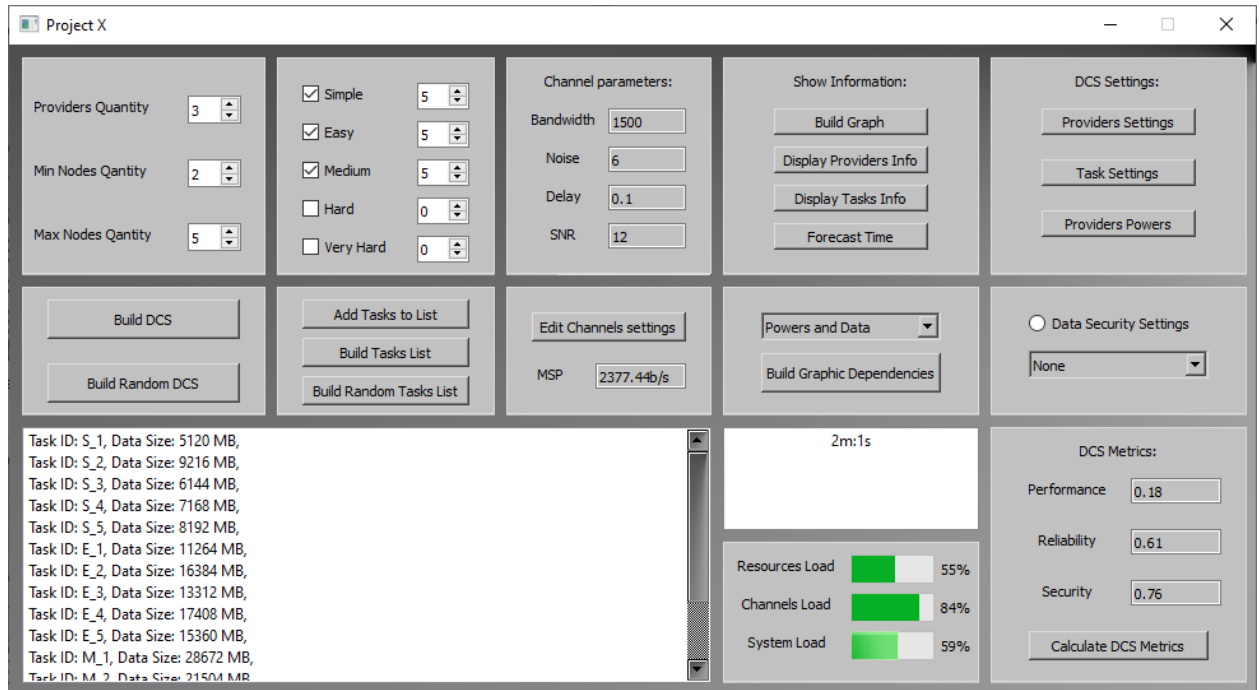


Рис. 3.9. Головне вікно з виводом прогнозованого часу та оцінками системи.

З рисунка 3.9 можемо побачити, що при завантаженні постачальників та їх ресурсів усього на 55 відсотків, завантаженість каналів занадто висока. Також отримали оцінки. Оцінка Performance = 0.18, яка повідомляє про те, що не ефективно використовувати постачальників з такими ресурсами у такій системі з такими задачами. Також отримали середні показники надійності Reliability = 0.59. Але лише 76 відсотків даних було не пошкоджено під час втручання до системи, про що каже нам показник Security = 0.76. Оскільки, ми не використали ніяких заходів безпеки, така велика втрата даних є зрозумілою, але для наглядності наведемо графік втрати ресурсів у часі, для цього натискаємо “Build Graphic Dependencies”, вибравши попередньо таку залежність “Packet Loss” у списку, та отримаємо результат на рисунку 3.10. Також побудуємо графік щодо часу виконання кожного завдання відповідно до кількості вузлів кожного постачальника, вибравши залежність “Power and

Data”, та натиснувши “ Build Graphic Dependencies ”, результатом є графік на рисунку 3.11.

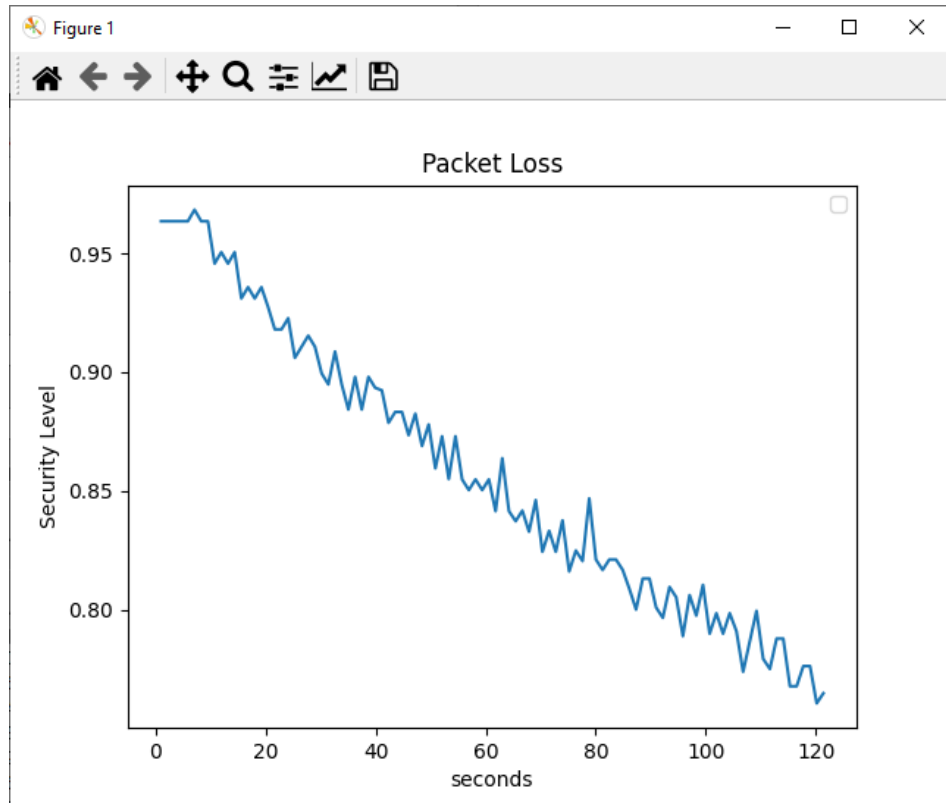


Рис. 3.10 Графік втрати даних без використання заходів безпеки.

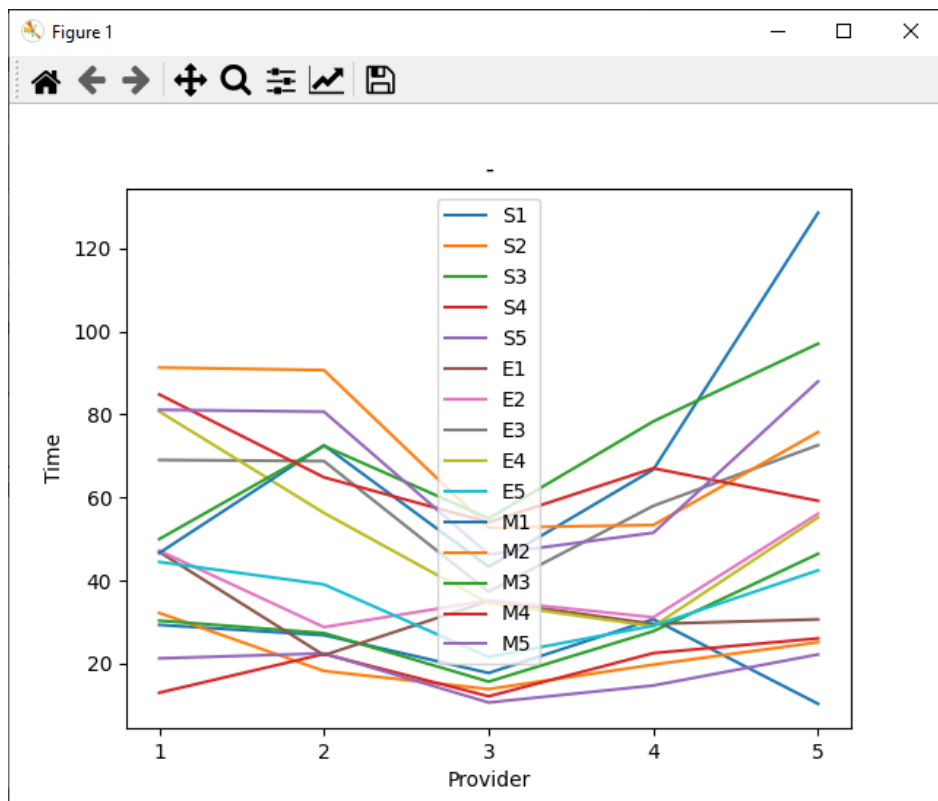


Рис. 3.11 Графік виконання різних завдань в залежності від кількості вузлів.

Тепер для того щоб підвищити ефективність системи, додаємо декілька завдань складності тяжка, 10 штук, та дуже складна, 5 штук. Одразу розрахуємо прогнозований час, завантаженість ресурсів та оцінки системи.

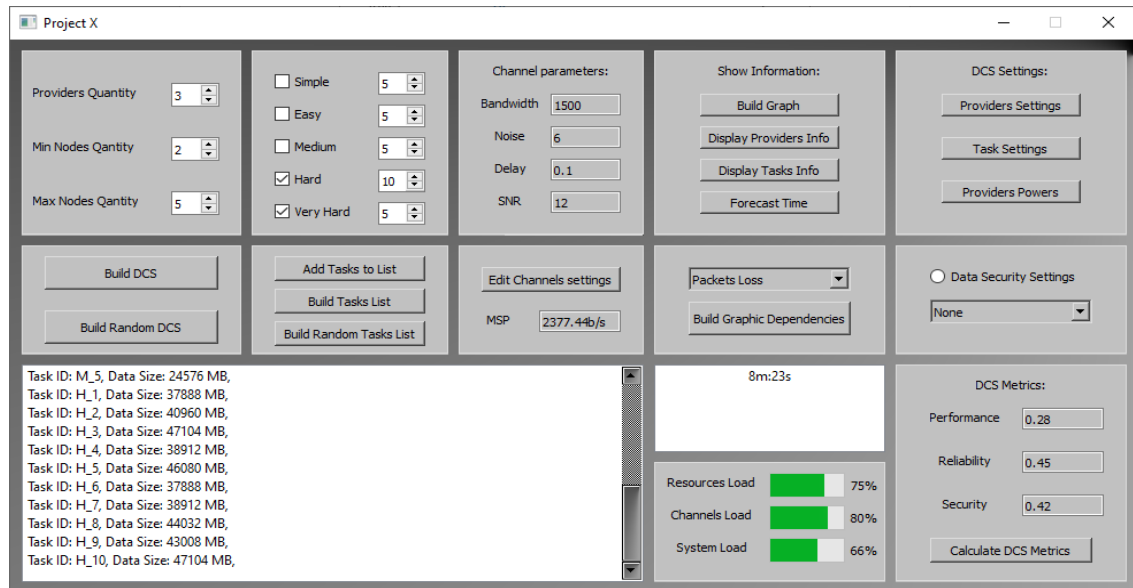


Рис 3.12 Головне вікно з виводом показників, після додавання завдань.

З рисунку 3.12 бачимо, що хоч оцінка продуктивності Performance = 0.28 стала краще, але метрики надійності системи, а особливо її захищеності погіршилися Reliability = 0.45 та Security = 0.42. Також побудуємо графік втрати ресурсів у часі для поточного стану системи на рисунку 3.13.

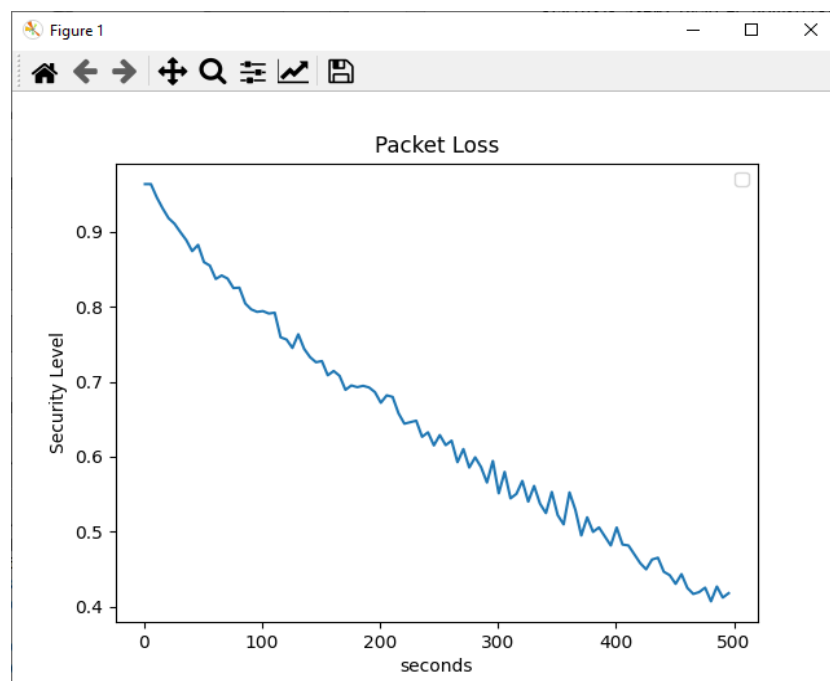


Рис. 3.13 Втрати даних без використання заходів безпеки після змін запитів.

Можемо побачити, як з ростом часу виконання завдань, збільшуються втрати даних, тому виберемо метод захисту: моніторинг активності, який полягає у тому, що під час моніторингу, з вірогідністю 95 відсотків атака буде повністю заблокована, але витрачає більше ресурсів системи у такому випадку. А також додаймо ще одного постачальника ресурсів, для цього натискаємо “Providers Settings”, після чого впливе вікно, як на рисунку 3.14.

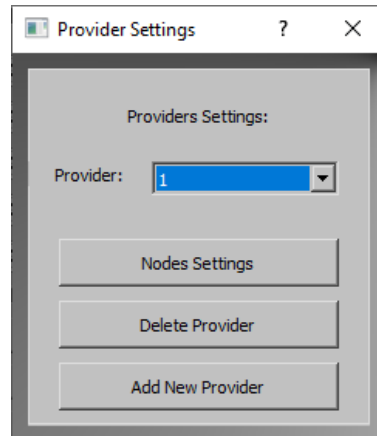


Рис. 3.14 Вікно для конфігурації ресурсів постачальників та їх вузлів.

Додаємо двох провайдерів, натиснувши кнопку “Add New Provider” двічі. Також виведемо новий побудований граф про них, як на рисунку 3.15, натиснувши “Draw Graph”.

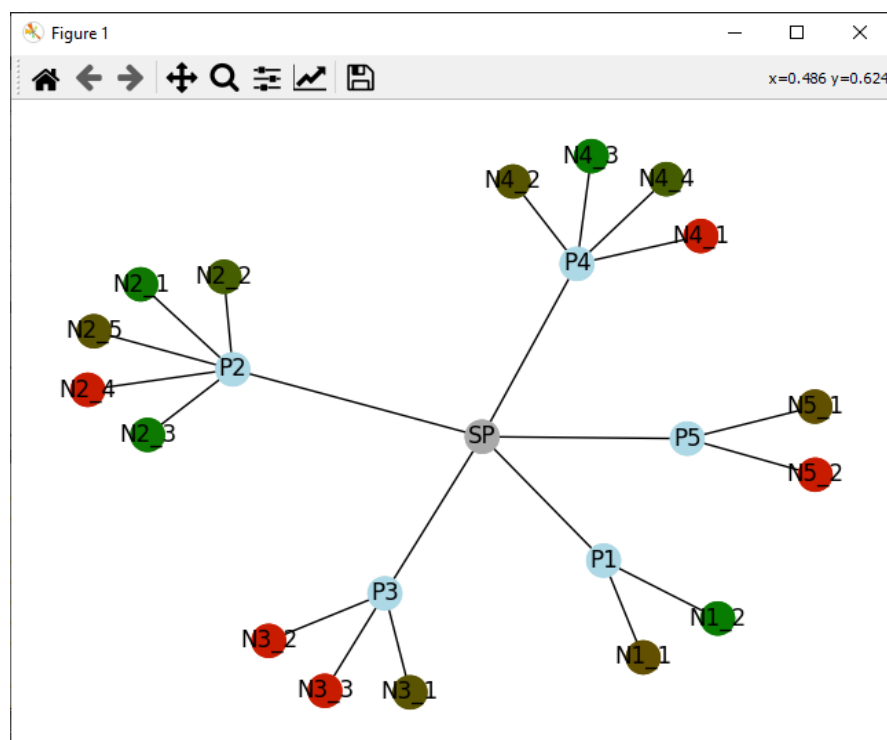


Рис. 3.15 Граф РКС після додавання двох нових постачальників..

Після чого маємо такі показники та оцінки після прогнозування часу та завантаженості ресурсів:

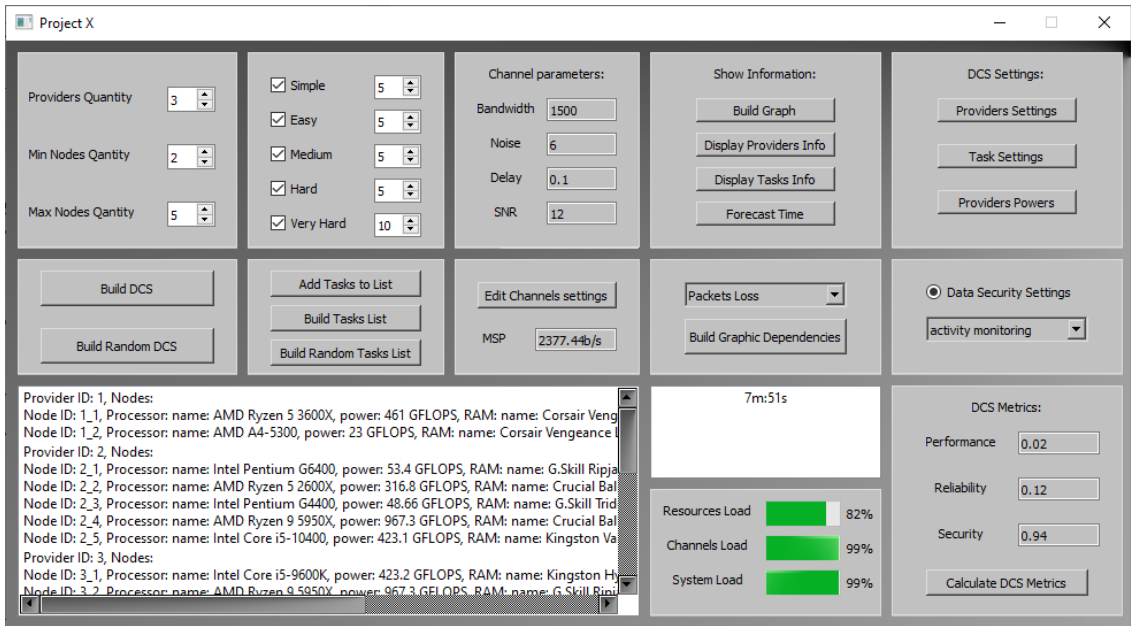


Рис. 3.16. Головне вікно з виводом показників, після додавання постачальників

З рисунку 3.16 бачимо, що усі показники окрім безпеки, дуже погані, це відбувається через те, що канали повністю забиті, та багато ресурсів працюють без завдання. Тому збільшуємо такі параметри каналів системи, як Bandwidth до 1800, SNR (Signal-to-ratio) до 18, та зменшуємо шум до 3, а delay до 0.01. Результат можна передивитись на рисунку 3.17.

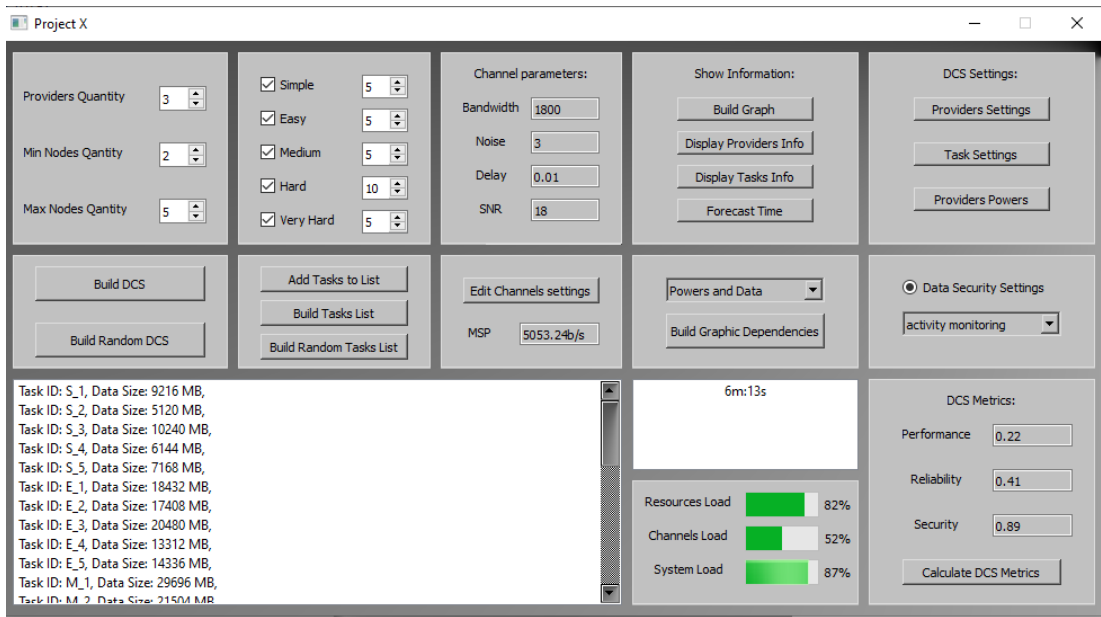


Рис. 3.17. Головне вікно після змін параметрів каналу.

Тепер переглянувши ще раз граф на рисунку 3.15, побачимо, що є дуже багато слабких вузлів, які відображаються відтінками ярко зеленого, це вузли: N1_2, N2_1, N2_3, N4_3. Тому змінюємо їх конфігурації натискаючи “Providers Settings”. Після чого у новому вікні обираємо потрібний нам провайдер та натискаємо “Nodes Settings”.

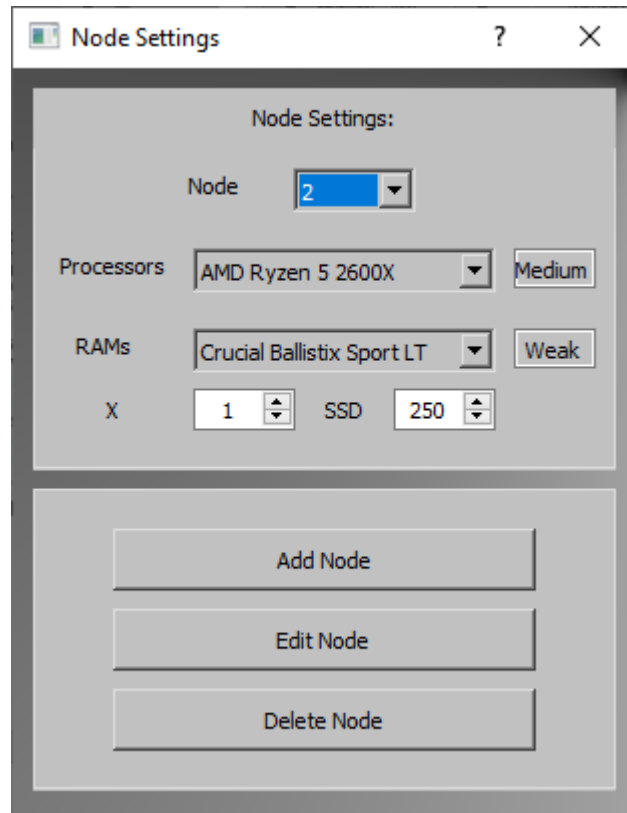


Рис. 3.18. Додаткове вікно для змін налаштування вузлів.

У новому вікні обираємо номер вузла, як на рисунку 3.18, та бачимо що в нього слабка оперативна пам'ять, та її кількість теж мала, також на заважало б підвибити розмір SSD носія. Змінивши усі слабкі вузли, шляхом натискання на comboBox з назвами відповідних компонент та вибору інших зі списку, а також змінюючи значення у відповідних spinBox, знов побудуємо граф для нашої ПК. Отримуємо результат на рисунку 3.19.

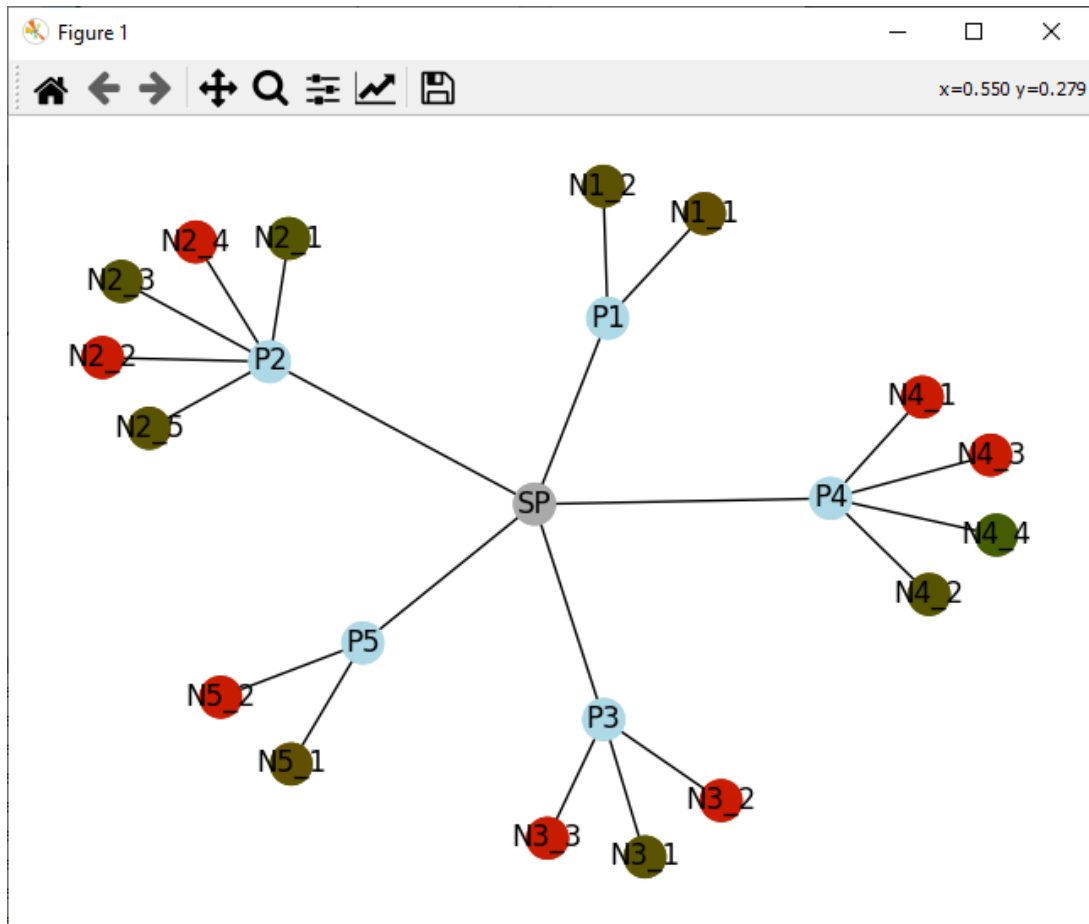


Рис. 3.19. РКС у вигляді графу після редагування вузлів постачальників.

Після цього повертаємось до головного вікна та натискаємо на “Providers Powers”, та відкривається нове вікно, де ми можемо за допомогою повзунка редагувати ліміти потужностей кожного окремого постачальника. Змінимо потужності усіх постачальників до ~80 відсотків, як на рисунку 3.20.

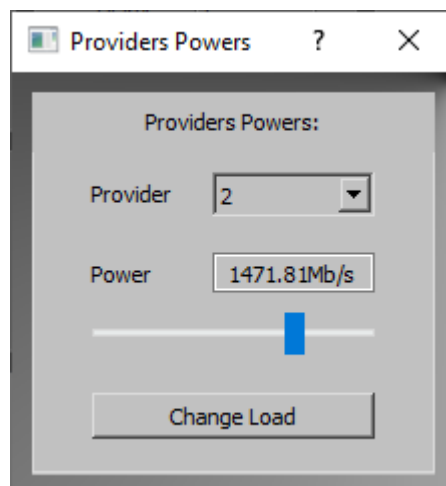


Рис. 3.20 Додаткове вікно для редагування потужностей.

Після чого повернемося до головного вікна, та після запуску прогнозу та розрахунку оцінок отримуємо результати, як на рисунку 3.21.

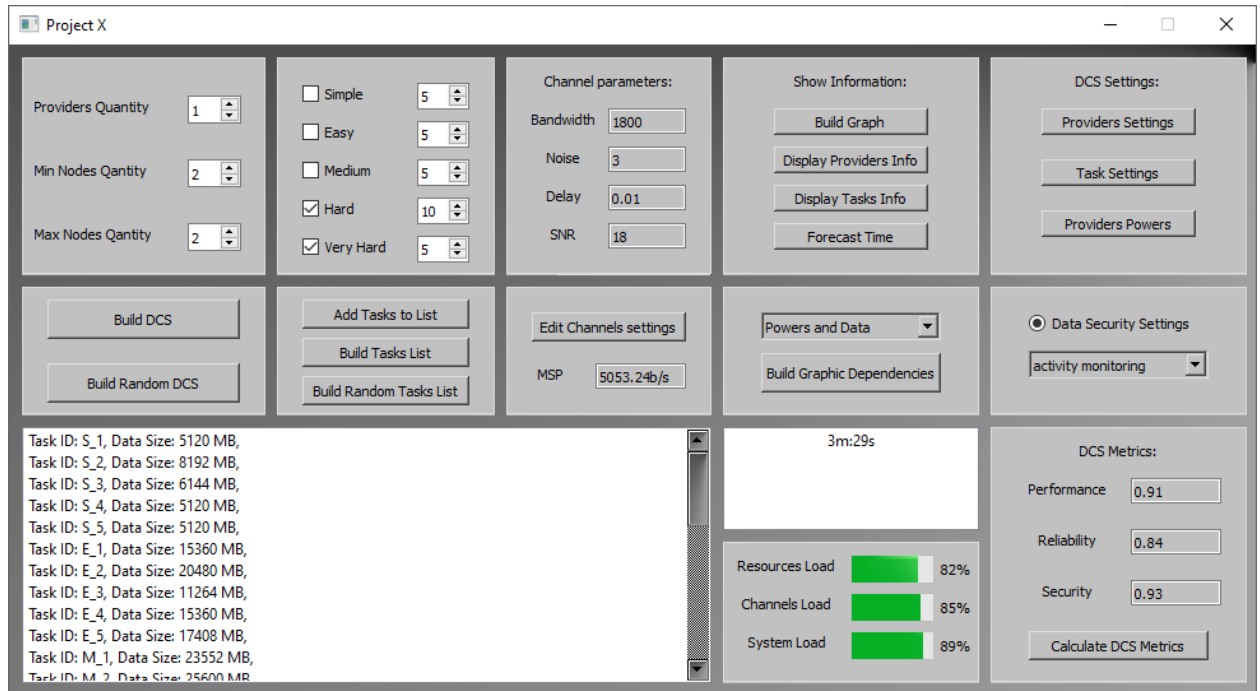


Рис. 3.21. Головне вікно після оптимізації усіх ресурсів системи.

З цього можемо зробити висновки, що при масштабуванні системи для виконання більш складних запитів, потрібно збільшувати й максимальну пропускну здатність системи, зменшуючи шум та затримку, а також збільшуючи кількість сигналів та ширину смуги. Також гарною оптимізацією є заміна слабких компонентів вузлів постачальників, а також корегування їх потужностей відповідно до можливостей системи. Ще необхідним є застосування додаткових рівнів безпеки, тому що при збільшенні часу роботи нашої системи відповідно до нових задач, збільшилась й кількість атак, та втрата даних досягала більше половини. Відповідно показникам, можна сказати, що мета оптимізації розподілу ресурсів виконана відповідно до отриманих з моделі оцінок.

3.4 Висновки до розділу 3

У цьому розділі була представлена оптимізаційна багатофакторна модель для оцінки показників РКС. Також за допомогою мови програмування Python та її бібліотек було створено програмний продукт, який імітує роботу цієї моделі, враховуючи її показники. Завдяки функціоналу цієї програми ми можемо на рівні компонентів масштабувати систему, та дивитись на зміни її показників, тим самим оптимізуючи використання її ресурсів. Також тут можна змодельовати граф системи з заданими параметрами, де вузлами графу є постачальники та їх ресурси, а ребрами – канали даних, по яких передається інформація. Також на тестовому прикладі було показано як можна оптимізувати систему згідно складності запитів. Окрім цього, на його основі було зроблено аналіз результатів та висновки щодо впливу вибору параметрів на показники системи на кожному кроці.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні оптимізаційних моделей оцінки показників розподілених комп'ютерних систем.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – якісний аналіз даних.

F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1 :

а) Python.

б) Java.

Функція F_2 :

а) Застосування вбудованих функцій.

б) Створення своїх обчислень значень.

Функція F_3 :

а) Використання шаблонних графіків.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

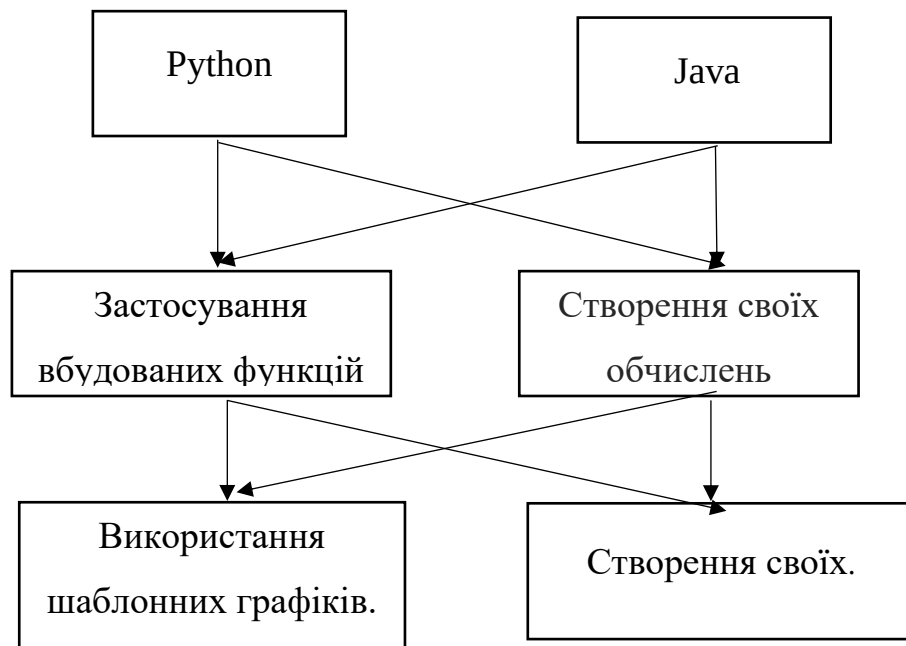


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Має широкий спектр бібліотек для обчислень, дуже зручний синтаксис, побудови графіків та моделювання мережевих систем у вигляді графів.	Може бути повільним в порівнянні з іншими мовами програмування. Не доступний на усіх пристроях.
	B	Має широкий спектр бібліотек для роботи з графіками та математичними моделями. Буде працювати в будь-якій системі.	Представляє собою ООП підхід. Що потребує більшої уваги до роботи з об'єктами та ініціалізаціям функцій та змінних.
F_2	A	Легкість використання. Також зручна та швидка реалізація загальних операцій без опису багатьох алгоритмів.	Може не мати потрібного функціоналу для опису задач.
	B	Гнучкість створення задач з урахуванням багатьох факторів.	Потребує більше часу. Іноді немає простих засобів реалізації.
F_3	A	Вже створений красивий графік з урахуванням потреб користувачів	Іноді не можливо реалізувати задумане. Потребує або генерації даних або тестову вибірку для більшої точності.
	B	Зручніше при виконанні персональних досліджень	Потребує багато часу для реалізації, досвіду у візуалізації

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 :

Варіант А є більш доступним, та не потребує багатого досвіду для роботи та програмування, є більшість потрібних функцій, не важкий функціонал для створення інтерфейсів. Хоча варіант Б теж має багато подібних функцій, але програє у легкості використання, тому відкидаємо варіант Б

Функція F_2 :

У програмі використовуємо обидва варіанта і А і Б одразу.

Функція F_3 :

За недостатністю досвіду у графічному моделюванні, обираємо А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_1a - F_2a - F_3a$$

$$F_1a - F_2b - F_3a$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- $X1$ – швидкодія мови програмування;
- $X2$ – об'єм пам'яті для обчислень та збереження даних;
- $X3$ – час навчання даних;
- $X4$ – потенційний об'єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	$X1$	оп/мс	35	65	110
Об'єм пам'яті	$X2$	Мб	150	95	65
Час попередньої обробки даних	$X3$	мс	80	60	50
Потенційний об'єм програмного коду	$X4$	кількість рядків коду	1500	1000	850

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

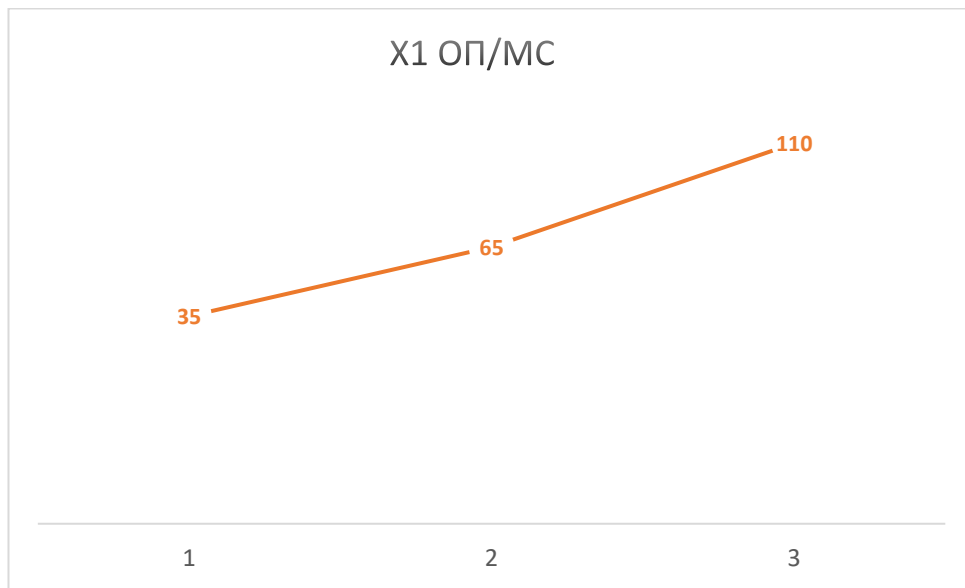


Рисунок 4.2 – X1, швидкодія мови програмування

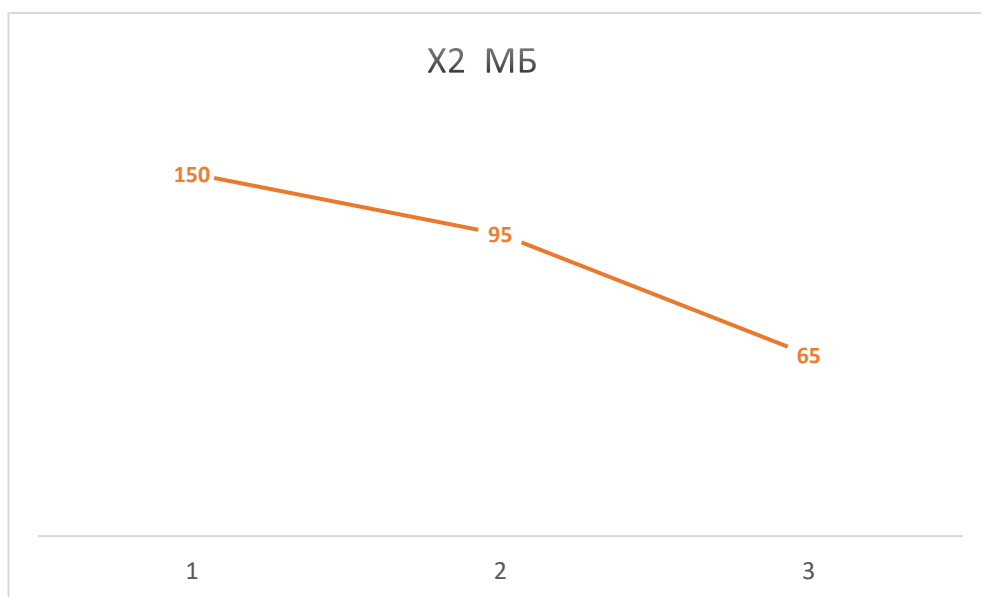


Рисунок 4.3 – X2, об'єм пам'яті

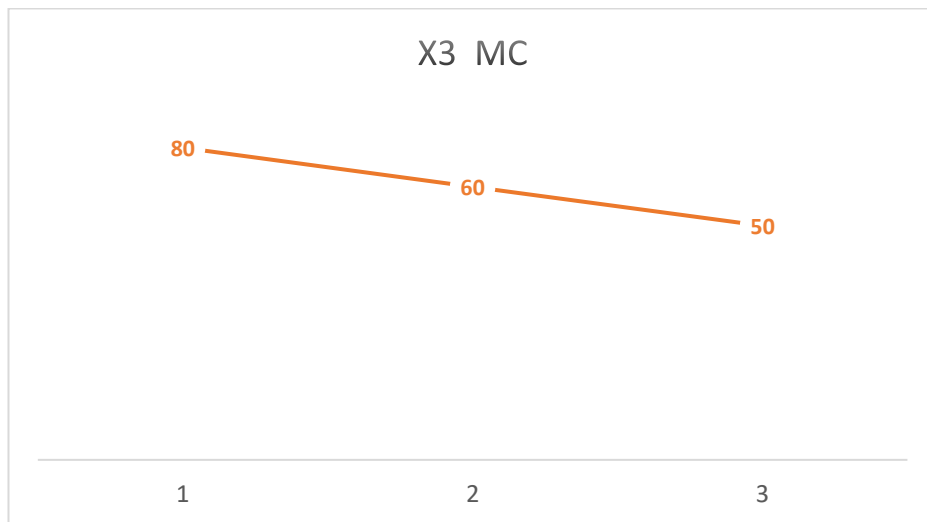


Рисунок 4.4 – X3, час попередньої обробки даних



Рисунок 4.5 – X4, потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при

знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	1	2	2	1	1	10	-6,75	45,56
X2	Об'єм пам'яті	Мб	3	2	4	3	4	2	3	21	4,25	18,06
X3	Час попередньої обробки даних	мс	2	2	1	1	1	3	2	12	-4,75	22,56
X4	Потенційний об'єм програмного коду	Кількість рядків коду	4	2	3	4	4	3	4	24	7,25	52,56
	Разом		10	10	10	10	10	10	10	67		138,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 67, \quad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 16,75 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всіх параметрах повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 181. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 138,75}{7^2(4^3 - 4)} = 0,738 > W_k = 0,56. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,56.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	=	<	<	<	<	<	<	0,5
X1 і X3	<	=	=	>	>	<	<	<	0,5
X1 і X4	<	=	<	<	<	<	<	<	0,5
X2 і X3	>	=	>	>	>	<	>	>	1,5
X2 і X4	<	=	>	<	=	<	<	<	0,5
X3 і X4	<	=	<	<	<	=	<	<	0,5

Числове значення, що визначає ступінь переваги i -го параметра над j -тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості K_{ei} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i} \quad (4.7)$$

$$b_i = \sum_{j=1}^N a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{Bi}} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \quad (4.9)$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер.	
	1	2	3	4	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2
X1	1	0,5	0,5	0,5	2,5	0,16	6,25	0,09	10,94	0,04
X2	1,5	1	1,5	0,5	4,5	0,28	20,25	0,29	65,81	0,27
X3	1,5	0,5	1	0,5	3,5	0,22	12,25	0,18	30,63	0,13
X4	1,5	1,5	1,5	1	5,5	0,34	30,25	0,44	136,13	0,56
Всього:					16	1	69	1	243,5	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X_2 (Об'єм пам'яті), X_3 (час попередньої обробки даних) та X_4 (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X_1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

K_{ei} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації
основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	115	26	0,16	4,16
F3	A	X2	73	35	0,28	9,8
	Б	X3	50	14	0,22	3,08
F4	A	X4	65	21	0,34	7,14

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \dots + K_{\text{ТУ}}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4,16 + 9,8 + 7,14 = 21,1$$

$$K_{K2} = 4,16 + 3,08 + 7,14 = 14,38$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_P \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

K_P – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 37$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання:

$K_{\Pi} = 1.8$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.9$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 37 \cdot 1.8 \cdot 0.9 = 59,94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_P = 29$ людино-днів, $K_{\Pi} = 0.9$, $K_{СК} = 1$, $K_{СТ} = 0.8$:

$$T_2 = 29 \cdot 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (59,94 + 20.88 + 4.8 + 20.88) \cdot 8 = 852 \text{ людино-годин.}$$

$$T_{II} = (59,94 + 20.88 + 6.91 + 20.88) \cdot 8 = 868,88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 21000 грн., один аналітик в області даних з окладом 23000. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн.,} \quad (4.14)$$

де M – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$C_q = \frac{21000 + 21000 + 23000}{3 \cdot 21 \cdot 8} = 128,96 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{зп} = C_q \cdot T_i \cdot K_d, \quad (4.16)$$

де C_q – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{зп} = 128,96 \cdot 852 \cdot 1.2 = 131848,70 \text{ грн.}$$

$$\text{II. } C_{зп} = 128,96 \cdot 868.88 \cdot 1.2 = 134460,91 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$\text{I. } C_{від} = C_{зп} \cdot 0.22 = 144025,488 \cdot 0.22 = 29006,71 \text{ грн.}$$

$$\text{II. } C_{від} = C_{зп} \cdot 0.22 = 146878,95 \cdot 0.22 = 29581,40 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_m)

Так як одна ЕОМ обслуговує одного програміста з окладом 23000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{г} = 12 \cdot M \cdot K_3 = 12 \cdot 21000 \cdot 0,2 = 50400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3П} = C_{Г} \cdot (1 + K_3) = 50400 \cdot (1 + 0.2) = 60480 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{3П} \cdot 0.22 = 60480 \cdot 0.22 = 13305,6 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$C_A = K_{ТМ} \cdot K_A \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 10000 = 3500 \text{ грн.,}$$

де $K_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{ТМ} \cdot Ц_{ПР} \cdot K_P = 1.4 \cdot 10000 \cdot 0.08 = 1120 \text{ грн.,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\begin{aligned} T_{ЕФ} &= (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 104 - 12 - 16) \cdot 8 \cdot 0.35 = \\ &= 627,2 \text{ години,} \end{aligned}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_C \cdot K_3 \cdot C_{\text{ЕН}} = 627,2 \cdot 0,3 \cdot 0,3 \cdot 4,87 = 274,9 \text{ грн.},$$

де N_C – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = C_{\text{ПР}} \cdot 0,67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_A + C_P + C_{\text{ЕЛ}} + C_H, \quad (4.17)$$

$$C_{\text{ЕКС}} = 66240 + 14572,8 + 3500 + 1120 + 274,9 + 6700 = 92407,7 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 92407,7 / 627,2 = 137,47 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-\Gamma} \cdot T, \quad (4.18)$$

$$\text{I. } C_M = 137,47 \cdot 852 = 117124,44 \text{ грн.}$$

$$\text{II. } C_M = 137,47 \cdot 868,88 = 119444,93 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67, \quad (4.19)$$

$$\text{I. } C_H = 131848,70 \cdot 0,67 = 88338,63 \text{ грн.}$$

$$\text{II. } C_H = 134460,91 \cdot 0,67 = 90088,81 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H, \quad (4.20)$$

$$\text{I. } C_{ПП} = 131848,70 + 29006,71 + 117124,44 + 88338,63 = 366318,48 \text{ грн.}$$

$$\text{II. } C_{ПП} = 134460,91 + 29581,40 + 119444,93 + 90088,81 = 373576,05 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{kj} / C_{\Phi j}, \quad (4.21)$$

$$K_{\text{ТЕР}1} = 21,1 / 366318,48 = 5,760 \cdot 10^{-5},$$

$$K_{\text{ТЕР}2} = 14,38 / 373576,05 = 3,849 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}1} = 5,760 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{ТЕР}} = 5,760 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір програмного продукту – Python;
- реалізація важливої постановки з допомогою вбудованих функцій;
- використання стандартного інтерфейсу для побудови значень.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидко реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до розділу 4

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У розділі 1 було розглянуто основні поняття з приводу того, що взагалі таке розподілена комп'ютерна система (РКС). Також визначали найбільш важливі її особливості та розглянули існуючі типи таких систем, взявши декілька реальних прикладів для аналізу. Також виявили більш значні метрики, такі як: продуктивність, доступність, надійність, масштабованість та витривалість та як вони впливають на РКС. Знаючи усі ці особливості ми можемо ефективніше використовувати ресурси, забезпечувати взаємодію компонентів, неперервну роботу та масштабувати систему для зростаючих потреб. Також завдяки існуючим моделям та засобам оцінки РКС з використанням баз даних, ми упевнились, що використання баз даних у процесі оцінювання (РКС) дозволяє збирати та зберігати дані про роботу компонентів системи та їх взаємодію, таких як завантаженість компонентів, швидкість передачі даних, час відповіді на запити, використання ресурсів процесору та пам'яті, мережевий трафік та інші важливі метри. Окрім цього, ми можемо проводити аналіз цих даних в реальному часі, що дозволяє виявляти потенційні проблеми та усувати їх, сприяючи підвищенню ефективності та надійності системи.

У розділі 2 ми спочатку розглянули існуючі загальні підходи щодо оптимізації. Були розглянуті такі основні підходи, як розподілене планування ресурсів, оптимізація мережевої комунікації, масштабування, оптимізація алгоритмів та обробки даних, використання кешування та реплікація даних, а також керування завантаженістю ресурсів, допомагають підвищити продуктивність та ефективність розподіленої системи. Після чого ми розглянули вже існуючі моделі, усі з яких напряду або ж опосередковано використовують в основі розглянуті раніше підходи щодо оптимізації. Окрім цього, загальний огляд таких моделей оптимізації РКС показує, що існують різні підходи та приклади їх застосування в сучасному світі. Моделі, такі як

оптимізація завантаження, оптимізація індексів, оптимізація запитів, оптимізація розподіленого зберігання даних та оптимізація масштабування, допомагають досягти максимальної продуктивності та ефективності розподіленої системи. Ці моделі використовуються в відомих нам різних сучасних системах, таких як Google MapReduce, Oracle Database, Apache Cassandra, Amazon S3, Google File System, Twitter і Facebook's Haystack, для оптимізації різних аспектів системи та забезпечення надійності та швидкодії.

Після чого ми порівняли усі описані нами методи оптимізації, за такими параметрами: умови завантаженості, обсяг даних, складність задачі, а також виділили більш вагомі переваги та недоліки цих моделей. Також за результатом цього міні дослідження можемо зробити такі висновки: модель оптимізації завантаження (Load Balancing) ефективна при високій завантаженості системи та розподілі завдань між вузлами, забезпечуючи високу доступність. Модель оптимізації запитів (Query Optimization) покращує продуктивність складних запитів та оптимізує ресурси, але вимагає аналізу та оптимізації запитів. Модель оптимізації масштабування (Scalability Optimization) забезпечує гнучкість системи для росту навантаження та оптимізацію розподілу ресурсів, але вимагає налагодження та налаштування для масштабування.

Також ми ще визначили що управління ресурсами у РКС включає в себе механізми: моніторингу ресурсів, планування ресурсів, оптимізації завантаження, масштабування та енергоспоживання. Окрім цього, ці механізми в свою чергу дозволяють ефективно використовувати ресурси, розподіляти завдання, балансувати навантаження, масштабувати систему та знижувати споживання енергії. Наприклад, такі відомі системи, як Kubernetes, Apache Hadoop, Apache Spark та PowerNap є демонстрацією таких механізмів, які допомагають досягти максимальної продуктивності та ефективності у РКС.

У розділі 3 була представлена оптимізаційна багатофакторна модель для оцінки показників РКС. Також за допомогою мови програмування Python та її бібліотек було створено програмний продукт, який імітує роботу цієї

моделі, враховуючи її показники. Завдяки функціоналу цієї програми ми можемо на рівні компонентів масштабувати систему, та дивитись на зміни її показників, тим самим оптимізуючи використання її ресурсів. Також тут можна змодельовати граф системи з заданими параметрами, де вузлами графу є постачальники та їх ресурси, а ребрами – канали даних, по яких передається інформація. Також на тестовому прикладі було показано як можна оптимізувати систему згідно складності запитів. Окрім цього, на його основі було зроблено аналіз результатів та висновки щодо впливу вибору параметрів на показники системи на кожному кроці.

У розділі 4 було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту на мові програмування Python.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Andrew S. Tanenbaum, Maarten van Steen, Distributed Systems Principles and Paradigms, 2006, 707 p.
2. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Database Systems: The Complete Book June 5, 2008, 1248 p.
3. V. Mukhin, Y. Kornaga, V. Bondarenko, V. Zavgorodnii, O. Herasymenko and O. Sholokhov, "Mathematical Model for Heterogeneous Databases Parameters Estimation in Distributed Systems with Dynamic Structure," 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2020, pp. 158-161, doi: 10.1109/ATIT50783.2020.9349331.
4. Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, High Performance MySQL: Optimization, Backups, and Replication, Third edition (April 17, 2012), 826 p.
5. Viktor Mayer-Schönberger, Kenneth Cukier, Big Data: A Revolution That Will Transform How We Live, Work, and Think, Reprint edition (March 4, 2014), 272 p.
6. Sadashiv N., Kumar Dilip S.M. Cluster, Grid and Cloud computing: A detailed comparison. The 6th International Conference on Computer Science & Education (ICCSE 2011), August 3–5, 2011, 477 p.
7. Ніколов А ., Кластерні системи високої готовності, 2005 - № 8.
8. George Coulouris, Jean Dollimore, im Kindberg, Gordon Blair, Distributed Systems: Concepts and Design, 5th edition (April 27, 2011), 1080 p.
9. Розподілена система керування – Вікіпедія [URL]: https://uk.wikipedia.org/wiki/Розподілена_система_керування
10. Raghu Ramakrishnan. Database Management Systems 3rd edition (August 14, 2002), 1104 p.
11. В. В. Пасічник, В. А. Резніченко, Організація баз даних та знань / – К.: Видавнича група BHV, 2006. – 384 с.

12. MySQL DataBase documentation [URL]:
<https://www.mysql.com/>
13. Lizhe Wang, Wei Jie, Jinjun Chen, Grid Computing: Infrastructure, Service, and Applications 1st Edition (October 12, 2017), 528 p.
14. Качурів В . Сучасні розподілені файлові системи для Linux: Основні відомості (переклад) // Комп'ютера . - 2002. - № 9.
15. Sandro Fiore, Giovanni Aloisio, Grid and Cloud Database Management 2011th edition (July 28, 2011), 363 p.
16. James D. McCabe, Network Analysis, Architecture, and Design, 3rd edition (June 27, 2007), 496 p.
17. Корнага Я. І. Моделі та методи організації та управління гетерогенними розподіленими базами даних з динамічною структурою на основі мережцентричного підходу : дис. докт. техн. наук : 05.13.06 / Корнага Ярослав Ігорович – Київ, 2020. – 328 с.
18. Python PyQt5 and PyQt-tools [URL]:
<https://doc.qt.io/qtforpython-6/>
19. NetworkX – NetworkX documentation [URL]:
<https://networkx.org/>
20. Matplotlib: Visualization with Python [URL]:
<https://matplotlib.org/>
21. Math: Mathematical functions with Python [URL]:
<https://docs.python.org/3/library/math.html>

ДОДАТОК А

```

import random
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import math
import matplotlib.colors as mcolors
from datetime import datetime, timedelta
from mpl_toolkits.mplot3d import Axes3D

from matplotlib.cm import ScalarMappable

class Memory:
    def __init__(self, name, frequency, memory):
        self.name = name
        self.frequency = frequency
        self.memory = memory

    def __str__(self):
        return f"name: {self.name}, frequency: {self.frequency} MHz, memory: {self.memory} GB"

class Processor:
    def __init__(self, name, power):
        self.name = name
        self.power = power

    def __str__(self):
        return f"name: {self.name}, power: {self.power} GFLOPS"

class Node:
    def __init__(self, node_id, processor, RAM_quant, ram, memory, pid=None):
        self.pid = pid
        self.node_id = node_id
        self.processor = processor
        self.RAM_quant = RAM_quant
        self.ram = ram
        self.memory = memory
        self.node_power = calculate_node_power(self)

    def set_power_new(self):
        self.node_power = calculate_node_power(self)

    def __str__(self):
        return f"Node ID: {self.pid}_{self.node_id}, Processor: {self.processor}, RAM: {self.ram} x {self.RAM_quant}, " \
            f"SSD: {self.memory} GB"

class Channel:
    def __init__(self, bandwidth, signal, noise):
        self.bandwidth = bandwidth

```

```

        self.signal = signal
        self.noise = noise
        self.msp = self.calc_msp()

    def calc_msp(self):
        msp = self.bandwidth * math.log2(1 + self.signal / self.noise)
        return msp

class Provider:
    def __init__(self, provider_id, nodes):
        self.provider_id = provider_id
        self.nodes = nodes
        self.power_level = 60
        self.power_avg = calculate_provider_power_avg(self)

    def set_power_new(self):
        self.power_avg = calculate_provider_power_avg(self)

    def set_power(self, power_lvl_new):
        self.power_level = power_lvl_new
        self.power_avg = calculate_provider_power_avg(self)

    def __str__(self):
        node_info = "\n".join(str(node) for node in self.nodes)
        return f"ID: {self.provider_id}, Nodes:\n{node_info}"

class Task:
    def __init__(self, task_id, data_size, task_diff, task_did=None):
        self.task_id = task_id
        self.data_size = data_size
        self.task_diff = task_diff
        self.task_did = task_did

    def __str__(self):
        return f"Task ID: {self.task_did}_{self.task_id}, Data Size: {self.data_size} MB,"

def process_time_calc(providers, tasks, msp, security):
    ex = 1
    data = 0
    time_prog = 0
    providers_powers = calculate_providers_avg_power(providers)
    for task in tasks:
        data += task.data_size
    if security == 'None':
        data = data
        time_prog = data / min(providers_powers, msp)
    elif security == 'encryption':
        data *= 1.1
        time_prog = data / min(providers_powers, msp)
    elif security == 'backup and update':
        data *= 1.1
        time_prog = data / min(providers_powers, msp)
    n = time_prog / 10
    for i in range(int(n)):
        rand = random.random()
        if rand <= 0.2:
            n = n

```

```

        else:
            time_prog += 5

    elif security == 'activity monitoring':
        ex = 1.05
        time_prog = data / min(providers_powers, msp) * ex

    return time_prog, data

def process_time_calc_for_stat(task, provider, msp, security):
    ex = 1
    data = 0
    time_prog = 0
    providers_powers = provider.power_avg
    nre = math.log2(random.uniform(1, 5))
    data = task.data_size
    data = data * random.uniform(0.85, 1)
    fie = math.cos(random.uniform(0, 1)) * math.tan(random.uniform(0, 1))
    if security == 'None':
        data = data
        time_prog = data / min(providers_powers, msp)
    elif security == 'encryption':
        data *= 1.1
        time_prog = data / min(providers_powers, msp)
    elif security == 'backup and update':
        data *= 1.1
        time_prog = data / min(providers_powers, msp)
        n = time_prog / 10
        for i in range(int(n)):
            rand = random.random()
            if rand <= 0.2:
                n = n
            else:
                time_prog += 5

    elif security == 'activity monitoring':
        ex = 1.05
        time_prog = data / min(providers_powers, msp) * ex
    print(time_prog)

    return time_prog

def calculate_node_power(node):
    processor_power = node.processor.power
    ram_power = node.RAM_quant * node.ram.frequency
    ssd_power = node.memory
    node_power = (0.7 * processor_power) + math.log2(0.2 * ram_power) +
math.log2(0.1 * ssd_power)
    return node_power

def calculate_node_powers_avg(nodes):
    i = 0
    power_avg = 0
    for node in nodes:
        power_avg += calculate_node_power(node)
        i += 1
    power_avg /= i

```

```

    return power_avg

def calculate_performance(providers, delay, time, data, quant, msp, tasks):
    n = 0
    for provider in providers:
        n += len(provider.nodes)
    c = (n * (n - 1)) / 4
    pr_avg = calculate_providers_avg_power(providers)
    labda = 1 - pr_avg / msp
    if labda <= 0.15:
        labda = 0.2
    # tdc = data / (n * pr_avg) + 0.5 * c * (100/delay) * labda ** 2
    single = data / len(tasks)
    tsing = single / (pr_avg / len(providers))
    tdc = (single / pr_avg) * labda ** 2 + 0.5 * c * delay
    dc = tsing / tdc
    dc *= 0.01
    return dc

def calculate_provider_power_avg(provider):
    i = 1
    per = provider.power_level
    provider_avg = 0
    for node in provider.nodes:
        provider_avg += calculate_node_power(node)
        i -= 0.02
    provider_avg *= i
    if 1 <= per <= 30:
        provider_avg *= (per / 100) * random.uniform(0.75, 0.85)
    elif 31 <= per <= 60:
        provider_avg *= (per / 100) * random.uniform(0.95, 1.05)
    elif 61 <= per <= 84:
        provider_avg *= (per / 100) * random.uniform(1.15, 1.25)
    elif 84 <= per <= 99:
        provider_avg *= (per / 100) * random.uniform(1.35, 1.5)

    return provider_avg

def calculate_security(security, time, providers, tasks):
    n = int(time / 5)
    if n <= 0:
        n = 1
    data = 0
    level = 0
    m = 0
    for prov in providers:
        m += int(len(prov.nodes))
    p = len(tasks)
    for task in tasks:
        data += task.data_size
    if security is None or security == 'None':
        k = 1
        for i in range(int(n)):
            rand = random.random()
            if rand <= 0.2:
                data = data
            else:
                k -= 0.005

```

```

        coef = (data * k) / data
        level = coef * (1 - (data / (data / math.sqrt(n * m))) / 100)
        if level < 0:
            level = 0
    elif security == 'encryption':
        level = 1 - (data / (data * random.uniform(12, 15) / math.sqrt(n * m)))
/ 100
    elif security == 'backup and update':
        level = 100
    elif security == 'activity monitoring':
        for i in range(int(n)):
            rand = random.random()
            if rand <= 0.05:
                n = n
            else:
                n -= 1
        if n <= 0:
            level = 100
        else:
            level = 1 - (data / (data / math.sqrt(n * m))) / 100
    return level

def plot_sec(security, time, providers, tasks):
    y = []
    xxx = np.linspace(1, time, 100).tolist()
    for x in xxx:
        y_item = calculate_security(security, x, providers, tasks)
        y.append(y_item)
    plt.plot(xxx, y)
    plt.xlabel('seconds')
    plt.ylabel('Security Level')
    plt.title('Packet Loss')
    plt.legend()
    plt.show()
    print(max(xxx))

def calculate_providers_avg_power(providers):
    pn_avg = 0
    i = 1
    for prov in providers:
        pn_avg += prov.power_avg
        i -= 0.03
    pn_avg *= i
    return pn_avg

def calculate_time_given(st_time, end_time):
    time_difference = (end_time - st_time).total_seconds()
    # Розрахунок годин, хвилин і секунд
    return time_difference

def calculate_reliability(time, providers, rl, cl, sl):
    level_break = 0
    if 120 <= time <= 300:

```



```

        level_break = 0.9
    elif 301 <= time <= 600:
        level_break = 1
    elif 601 <= time <= 900:
        level_break = 1.2
    elif time >= 900:
        level_break = 1.4
    else:
        level_break = 0.6
    rel = 1 - level_break * (((r1 / 100) ** 2 + (c1 / 100) ** 2 + (s1 / 100) **
2) / 3)
    return rel

```

```

def generate_time(time, tp, fora=None):
    if tp == 'h':
        new_time_start = fora * 360 + time
        now = datetime.now() + timedelta(seconds=new_time_start)
        end_time = now + timedelta(hours=fora)
    elif tp == 'm':
        new_time_start = fora * 60 + time
        now = datetime.now() + timedelta(seconds=new_time_start)
        end_time = now + timedelta(seconds=fora)
    else:
        print('Input error')
    if fora is None:
        new_time_start = time * random.randint(2, 4)
        new_time_end = time * random.randint(5, 8)
        now = datetime.now() + timedelta(seconds=new_time_start)
        end_time = now + timedelta(seconds=new_time_end)
    # Генеруємо випадкову дату у вказаних межах
    random_date = random.uniform(now.timestamp(), end_time.timestamp())
    random_datetime = datetime.fromtimestamp(random_date)

    return random_datetime

```

```

def generate_tasks(tasks, task_quant, task_diff, data_size=None):
    for task_id in range(1, task_quant + 1):
        if task_diff == 'Simple':
            # dcs_power = calculate_providers_avg_power(providers)
            data_size = random.randint(5, 10)
            data_size *= 1024
            # process_time = data_size/dcs_power
            task_did = 'S'
        elif task_diff == 'Easy':
            # dcs_power = calculate_providers_avg_power(providers)
            data_size = random.randint(11, 20)
            data_size *= 1024
            # process_time = data_size / dcs_power
            task_did = 'E'
        elif task_diff == 'Medium':
            # dcs_power = calculate_providers_avg_power(providers)
            data_size = random.randint(21, 30)
            data_size *= 1024
            # process_time = data_size / dcs_power
            task_did = 'M'
        elif task_diff == 'Hard':
            # dcs_power = calculate_providers_avg_power(providers)
            data_size = random.randint(31, 50)
            data_size *= 1024

```

```

        # process_time = data_size / dcs_power
        task_diff = 'H'
    elif task_diff == 'Very Hard':
        # dcs_power = calculate_providers_avg_power(providers)
        data_size = random.randint(51, 70)
        data_size *= 1024
        # process_time = data_size / dcs_power
        task_diff = 'VH'
    tasks.append(Task(task_id, data_size, task_diff, task_diff))

return tasks

def generate_random_tasks():
    tasks = []
    for task_id in range(random.randint(1, 5)):
        diff = random.choice(['Simple', 'Easy', 'Medium', 'Hard', 'Very Hard'])
        generate_tasks(tasks, random.randint(2, 10), diff)
    return tasks

def Graph_builder(providers):
    G = nx.Graph()

    G.add_node("SP", color='darkgray') # Вузол "SP" має темно-сірий колір

    min_node_power = float('inf')
    max_node_power = float('-inf')

    for pr in providers:
        G.add_node(f'P{pr.provider_id}', color='lightblue') # Всі постачальники
        # мають блакитний колір
        G.add_edge("SP", f'P{pr.provider_id}')

        for node in pr.nodes:
            node_power = calculate_node_power(node)
            min_node_power = min(min_node_power, node_power)
            max_node_power = max(max_node_power, node_power)

            node_color = get_node_color(node_power, min_node_power,
max_node_power)

            G.add_node(f'N{pr.provider_id}_{node.node_id}', color=node_color)
            G.add_edge(f'P{pr.provider_id}',
f'N{pr.provider_id}_{node.node_id}')

    pos = nx.spring_layout(G)

    node_colors = [G.nodes[node]['color'] for node in G.nodes]
    nx.draw(G, pos, with_labels=True, node_color=node_colors)
    nx.draw_networkx_nodes(G, pos, node_color=node_colors)
    plt.show()

def get_node_color(node_power, min_power, max_power):
    cmap = mcolors.LinearSegmentedColormap.from_list('green_to_red', ['green',
'red'])

    if min_power == max_power:
        normalized_power = node_power / (900 - 20)
    else:

```

```

        normalized_power = node_power / (900 - 20)

        node_color = cmap(normalized_power)

    return node_color

processors = []

# Слабкі процесори (5 видів)
weak_processors_dict = [
    {'назва': 'Intel Pentium G4400', 'Потужність': 48.66},
    {'назва': 'AMD Athlon 3000G', 'Потужність': 93.2},
    {'назва': 'Intel Celeron G5900', 'Потужність': 48.4},
    {'назва': 'AMD A4-5300', 'Потужність': 23},
    {'назва': 'Intel Pentium G6400', 'Потужність': 53.4},
]
weak_processors = []
for proc in weak_processors_dict:
    processor = Processor(proc['назва'], proc['Потужність'])
    weak_processors.append(processor)
    processors.append(processor)

medium_processors_dict = [
    {'назва': 'Intel Core i5-10400', 'Потужність': 423.1},
    {'назва': 'Intel Core i9-10900K', 'Потужність': 460.8},
    {'назва': 'AMD Ryzen 5 5600X', 'Потужність': 408},
    {'назва': 'Intel Core i7-9700K', 'Потужність': 423.2},
    {'назва': 'Intel Core i5-10400F', 'Потужність': 402.6},
    {'назва': 'AMD Ryzen 5 3600X', 'Потужність': 461},
    {'назва': 'Intel Core i7-8700K', 'Потужність': 453.8},
    {'назва': 'AMD Ryzen 7 2700X', 'Потужність': 384.3},
    {'назва': 'Intel Core i5-9600K', 'Потужність': 423.2},
    {'назва': 'AMD Ryzen 5 2600X', 'Потужність': 316.8},
    {'назва': 'Intel Core i5-9400', 'Потужність': 323},
    {'назва': 'Intel Core i5-9400F', 'Потужність': 344.7},
]
medium_processors = []
for proc in medium_processors_dict:
    processor = Processor(proc['назва'], proc['Потужність'])
    medium_processors.append(processor)
    processors.append(processor)

high_processors_dict = [
    {'назва': 'Intel Core i9-11900K', 'Потужність': 836.9},
    {'назва': 'AMD Ryzen 9 5900X', 'Потужність': 908},
    {'назва': 'Intel Core i9-12900K', 'Потужність': 1180},
    {'назва': 'AMD Ryzen 9 5950X', 'Потужність': 967.3},
    {'назва': 'AMD Ryzen 9 5950X', 'Потужність': 967.3},
]

high_processors = []
for proc in high_processors_dict:
    processor = Processor(proc['назва'], proc['Потужність'])
    high_processors.append(processor)
    processors.append(processor)

RAM_memory = []
weak_RAM_dict = [
    {'назва': 'Kingston ValueRAM', 'частота': 2133, 'пам`ять': 4},
    {'назва': 'Crucial Ballistix Sport LT', 'частота': 2400, 'пам`ять': 8}
]

```

```

]

weak_RAM = []
for ram in weak_RAM_dict:
    memory = Memory(ram['назва'], ram['частота'], ram['пам`ять'])
    weak_RAM.append(memory)
    RAM_memory.append(memory)

medium_RAM_dict = [
    {'назва': 'Corsair Vengeance LPX', 'частота': 3000, 'пам`ять': 16},
    {'назва': 'G.Skill Ripjaws V', 'частота': 3200, 'пам`ять': 16},
    {'назва': 'Kingston HyperX Fury', 'частота': 2666, 'пам`ять': 16},
    {'назва': 'Corsair Dominator Platinum', 'частота': 3200, 'пам`ять': 32},
    {'назва': 'Crucial Ballistix Elite', 'частота': 3600, 'пам`ять': 32},
    {'назва': 'Corsair Vengeance RGB Pro', 'частота': 3600, 'пам`ять': 32},
]
medium_RAM = []
for ram in medium_RAM_dict:
    memory = Memory(ram['назва'], ram['частота'], ram['пам`ять'])
    medium_RAM.append(memory)
    RAM_memory.append(memory)

high_RAM_dict = [
    {'назва': 'Kingston HyperX Predator', 'частота': 3200, 'пам`ять': 64},
    {'назва': 'G.Skill Trident Z Neo', 'частота': 3600, 'пам`ять': 64}
]

high_RAM = []
for ram in high_RAM_dict:
    memory = Memory(ram['назва'], ram['частота'], ram['пам`ять'])
    high_RAM.append(memory)
    RAM_memory.append(memory)

def convert_seconds_to_hms(seconds):
    hours = seconds // 3600
    minutes = (seconds % 3600) // 60
    seconds = (seconds % 3600) % 60

    return hours, minutes, seconds

def generate_nodes(node_quant, IDs=None):
    nodes = []
    for node_id in range(1, node_quant + 1):
        nprocessor = random.choice(processors)
        nram = random.choice(RAM_memory)
        nmemory = random.randint(64, 512)
        nodes.append(Node(node_id, nprocessor, random.randint(1, 4), nram,
nmemory, IDs))

    return nodes

def check_container_proc(pr_name):
    proc_ind = 'None'
    for prcs in processors:
        if prcs.name == pr_name:
            if prcs in weak_processors:
                proc_ind = 'Weak'
            if prcs in medium_processors:

```

```

        proc_ind = 'Medium'
    if prcs in high_processors:
        proc_ind = 'High'
    return proc_ind

def check_container_ram(ram_name):
    ram_ind = 'None'
    for rams in RAM_memory:
        if rams.name == ram_name:
            if rams in weak_RAM:
                ram_ind = 'Weak'
            if rams in medium_RAM:
                ram_ind = 'Medium'
            if rams in high_RAM:
                ram_ind = 'High'
    return ram_ind

def generate_providers(provider_quant, min_node_quant, max_node_quant):
    providers = []
    for provider_id in range(1, provider_quant + 1):
        r_nodes = generate_nodes(random.randint(min_node_quant, max_node_quant),
provider_id)
        providers.append(Provider(provider_id, r_nodes))
    return providers

def generate_random_providers():
    providers = []
    for provider_id in range(1, random.randint(2, 10)):
        min_nodes = random.randint(2, 3)
        max_nodes = random.randint(3, 5)
        r_nodes = generate_nodes(random.randint(min_nodes, max_nodes),
provider_id)
        providers.append(Provider(provider_id, r_nodes))
    return providers

```

ДОДАТОК Б

```

import random
from PyQt5 import QtCore, QtGui, QtWidgets
import Blocks as blocks
from node_settings import Ui_Dialog as ns_Ui
from provider_settings import Ui_Dialog as ps_Ui
from tasks_settings import Ui_TaskSettings as TSTUi
from providers_powers import Ui_Dialog as PPUi
import matplotlib.pyplot as plt
import numpy as np

class Ui_MainWindow(object):

    def __init__(self):
        self.providers = []
        self.tasks = []
        self.channel = None
        self.delay = 1
        self.msp = 2377.44
        self.security = 'None'
        self.time = 0
        self.tasks_data = 0
        self.rl = 0
        self.cl = 0
        self.sl = 0

    def openWindow_provider_settings(self):
        self.window_provider_settings = QtWidgets.QDialog()
        self.ui = ps_Ui()
        self.ui.setupUi(self.window_provider_settings)
        self.window_provider_settings.setWindowTitle('Provider Settings')
        self.window_provider_settings.show()
        self.providers = sorted(self.providers, key=lambda prov: prov.provider_id)
        self.ui.providers = self.providers
        for provider in self.providers:
            self.ui.comboBox.addItem(str(provider.provider_id))
        self.ui.pushButton.clicked.connect(self.openWindow_node_settings)
        self.ui.pushButton_2.clicked.connect(self.on_click_2_ps)
        self.ui.pushButton_3.clicked.connect(self.on_click_3_ps)

    def on_click_2_ps(self):
        item = self.ui.comboBox.currentText()
        for provider in self.ui.providers:
            if int(item) == provider.provider_id:
                index = self.ui.providers.index(provider)
                self.ui.comboBox.removeItem(index)
                self.ui.providers.remove(provider)

    def on_click_3_ps(self):
        if len(self.ui.providers) > 0:
            ids = []
            for provider in self.ui.providers:
                ids.append(provider.provider_id)
            new_id = 1
            while new_id in ids:

```

```

        new_id += 1
        r_nodes2 = blocks.generate_nodes(random.randint(2, 5), new_id)
        self.ui.providers.append(blocks.Provider(new_id, r_nodes2))
        self.ui.providers = sorted(self.ui.providers, key=lambda prov:
prov.provider_id)
        self.ui.comboBox.clear()
        for provider in self.providers:
            self.ui.comboBox.addItem(str(provider.provider_id))
    else:
        r_nodes1 = blocks.generate_nodes(random.randint(2, 5), 1)
        self.ui.providers.append(blocks.Provider(1, r_nodes1))

def openWindow_node_settings(self):
    self.window_node_settings = QtWidgets.QDialog()
    self.uins = ns_Ui()
    self.uins.setupUi(self.window_node_settings)
    self.window_node_settings.setWindowTitle('Node Settings')
    self.window_node_settings.show()
    if len(self.ui.providers) > 0:
        self.uins.providers = self.ui.providers
        index = int(self.ui.comboBox.currentText()) - 1
        self.uins.index = index
        self.uins.nodes = self.ui.providers[index].nodes
        for item in self.ui.RAMs:
            self.uins.comboBox_2.addItem(str(item.name))
        for item in self.ui.processors:
            self.uins.comboBox_3.addItem(str(item.name))
        for node in self.providers[index].nodes:
            self.uins.comboBox_4.addItem(str(node.node_id))
        for node in self.providers[index].nodes:
            if node.node_id == int(self.uins.comboBox_4.currentText()):
                p_ind = blocks.check_container_proc(node.processor.name)
                p_ram = blocks.check_container_ram(node.ram.name)
                self.uins.comboBox_2.setCurrentText(str(node.ram.name))
                self.uins.comboBox_3.setCurrentText(str(node.processor.name))
                self.uins.spinBox_2.setValue(node.memory)
                self.uins.spinBox.setValue(node.RAM_quant)
                self.uins.lineEdit.setText(p_ind)
                self.uins.lineEdit_2.setText(p_ram)
    else:
        self.window_node_settings.close()

def open_Window_task_settings(self):
    self.wind_ts = QtWidgets.QDialog()
    self.uits = TSTUi()
    self.uits.setupUi(self.wind_ts)
    self.wind_ts.setWindowTitle('Task Settings')
    self.wind_ts.show()
    self.uits.tasks = self.tasks
    for task in self.tasks:
        self.uits.comboBox.addItem(str(task.task_did) + str(task.task_id))

def open_Window_providers_power(self):
    if len(self.providers) > 0:
        self.wind_pp = QtWidgets.QDialog()
        self.uipp = PPUi()
        self.uipp.setupUi(self.wind_pp)
        self.wind_pp.setWindowTitle('Providers Powers')
        self.wind_pp.show()
        self.providers = sorted(self.providers, key=lambda prov: prov.provider_id)

```

```

        self.uipp.providers = self.providers
        for provider in self.providers:
            self.uipp.comboBox_2.addItem(str(provider.provider_id))
        index = int(self.uipp.comboBox_2.currentText()) - 1
        self.uipp.index = index

self.uipp.lineEdit_2.setText(str(round(blocks.calculate_provider_power_avg(self.providers[index]), 2)) + 'Mb/s')
    else:
        self.listWidget_2.clear()
        item = QtWidgets.QListWidgetItem('Provider List is empty')
        item.setTextAlignment(QtCore.Qt.AlignCenter)
        self.listWidget_2.addItem(item)

def setupUi(self, MainWindow):
    MainWindow.setObjectName("ProjectX")
    MainWindow.setFixedSize(978, 510)
    MainWindow.setStyleSheet(
        "background-color: qconicalgradient(cx:1, cy:0.0284091, angle:15,
stop:0.210227 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255, 255));")
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.ProvidersConfig = QtWidgets.QGroupBox(self.centralwidget)
    self.ProvidersConfig.setGeometry(QtCore.QRect(10, 10, 191, 171))
    self.ProvidersConfig.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                       "background-color: rgb(193, 193, 193);")
    self.ProvidersConfig.setTitle("")
    self.ProvidersConfig.setAlignment(QtCore.Qt.AlignCenter)
    self.ProvidersConfig.setFlat(False)
    self.ProvidersConfig.setCheckable(False)
    self.ProvidersConfig.setObjectName("ProvidersConfig")
    self.spinBox = QtWidgets.QSpinBox(self.ProvidersConfig)
    self.spinBox.setGeometry(QtCore.QRect(130, 80, 42, 22))
    self.spinBox.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.spinBox.setObjectName("spinBox")
    self.spinBox.setMinimum(2)
    self.spinBox_2 = QtWidgets.QSpinBox(self.ProvidersConfig)
    self.spinBox_2.setGeometry(QtCore.QRect(130, 30, 42, 22))
    self.spinBox_2.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.spinBox_2.setObjectName("spinBox_2")
    self.spinBox_2.setMinimum(1)
    self.label = QtWidgets.QLabel(self.ProvidersConfig)
    self.label.setGeometry(QtCore.QRect(10, 30, 91, 16))
    self.label.setObjectName("label")
    self.label_2 = QtWidgets.QLabel(self.ProvidersConfig)
    self.label_2.setGeometry(QtCore.QRect(10, 80, 101, 16))
    self.label_2.setObjectName("label_2")
    self.label_3 = QtWidgets.QLabel(self.ProvidersConfig)
    self.label_3.setGeometry(QtCore.QRect(10, 130, 91, 16))
    self.label_3.setObjectName("label_3")
    self.spinBox_3 = QtWidgets.QSpinBox(self.ProvidersConfig)
    self.spinBox_3.setGeometry(QtCore.QRect(130, 130, 42, 22))
    self.spinBox_3.setStyleSheet("background-color: rgb(255, 255, 255);")
    self.spinBox_3.setObjectName("spinBox_3")
    self.spinBox_3.setMinimum(2)
    self.gridGroupBox_3 = QtWidgets.QGroupBox(self.centralwidget)
    self.gridGroupBox_3.setGeometry(QtCore.QRect(210, 10, 171, 171))
    self.gridGroupBox_3.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                       "background-color: rgb(193, 193, 193);")
    self.gridGroupBox_3.setFlat(False)

```



```

self.gridGroupBox_3.setObjectName("gridGroupBox_3")
self.checkBox = QtWidgets.QCheckBox(self.gridGroupBox_3)
self.checkBox.setGeometry(QtCore.QRect(20, 20, 82, 17))
self.checkBox.setObjectName("checkBox")
self.checkBox_2 = QtWidgets.QCheckBox(self.gridGroupBox_3)
self.checkBox_2.setGeometry(QtCore.QRect(20, 50, 82, 17))
self.checkBox_2.setObjectName("checkBox_2")
self.checkBox_3 = QtWidgets.QCheckBox(self.gridGroupBox_3)
self.checkBox_3.setGeometry(QtCore.QRect(20, 80, 82, 17))
self.checkBox_3.setObjectName("checkBox_3")
self.checkBox_4 = QtWidgets.QCheckBox(self.gridGroupBox_3)
self.checkBox_4.setGeometry(QtCore.QRect(20, 110, 82, 17))
self.checkBox_4.setObjectName("checkBox_4")
self.checkBox_5 = QtWidgets.QCheckBox(self.gridGroupBox_3)
self.checkBox_5.setGeometry(QtCore.QRect(20, 140, 81, 17))
self.checkBox_5.setObjectName("checkBox_5")
self.spinBox_5 = QtWidgets.QSpinBox(self.gridGroupBox_3)
self.spinBox_5.setGeometry(QtCore.QRect(110, 20, 41, 21))
self.spinBox_5.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox_5.setObjectName("spinBox_5")
self.spinBox_6 = QtWidgets.QSpinBox(self.gridGroupBox_3)
self.spinBox_6.setGeometry(QtCore.QRect(110, 50, 41, 21))
self.spinBox_6.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox_6.setObjectName("spinBox_6")
self.spinBox_7 = QtWidgets.QSpinBox(self.gridGroupBox_3)
self.spinBox_7.setGeometry(QtCore.QRect(110, 80, 41, 21))
self.spinBox_7.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox_7.setObjectName("spinBox_7")
self.spinBox_8 = QtWidgets.QSpinBox(self.gridGroupBox_3)
self.spinBox_8.setGeometry(QtCore.QRect(110, 110, 41, 21))
self.spinBox_8.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox_8.setObjectName("spinBox_8")
self.spinBox_9 = QtWidgets.QSpinBox(self.gridGroupBox_3)
self.spinBox_9.setGeometry(QtCore.QRect(110, 140, 41, 21))
self.spinBox_9.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox_9.setObjectName("spinBox_9")
self.listWidget = QtWidgets.QListWidget(self.centralwidget)
self.listWidget.setGeometry(QtCore.QRect(10, 300, 541, 201))
self.listWidget.setStyleSheet("background-color: rgb(255, 255, 255);")
self.listWidget.setObjectName("listWidget")
self.gridGroupBox_2 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_2.setGeometry(QtCore.QRect(560, 10, 201, 171))
self.gridGroupBox_2.setStyleSheet("background-color: rgb(52, 52, 52);\n"
    "background-color: rgb(193, 193, 193);")

self.gridGroupBox_2.setFlat(False)
self.gridGroupBox_2.setObjectName("gridGroupBox_2")
self.pushButton_4 = QtWidgets.QPushButton(self.gridGroupBox_2)
self.pushButton_4.setGeometry(QtCore.QRect(40, 40, 121, 21))
self.pushButton_4.setObjectName("pushButton_4")
self.pushButton_5 = QtWidgets.QPushButton(self.gridGroupBox_2)
self.pushButton_5.setGeometry(QtCore.QRect(40, 70, 121, 21))
self.pushButton_5.setObjectName("pushButton_5")
self.pushButton_7 = QtWidgets.QPushButton(self.gridGroupBox_2)
self.pushButton_7.setGeometry(QtCore.QRect(40, 100, 121, 21))
self.pushButton_7.setObjectName("pushButton_7")
self.pushButton_8 = QtWidgets.QPushButton(self.gridGroupBox_2)
self.pushButton_8.setGeometry(QtCore.QRect(40, 130, 121, 21))
self.pushButton_8.setObjectName("pushButton_8")
self.label_10 = QtWidgets.QLabel(self.gridGroupBox_2)
self.label_10.setGeometry(QtCore.QRect(50, 10, 101, 16))
self.label_10.setAlignment(QtCore.Qt.AlignCenter)

```

```

self.label_10.setObjectName("label_10")
self.gridGroupBox_4 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_4.setGeometry(QtCore.QRect(10, 190, 191, 101))
self.gridGroupBox_4.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_4.setFlat(False)
self.gridGroupBox_4.setObjectName("gridGroupBox_4")
self.pushButton = QtWidgets.QPushButton(self.gridGroupBox_4)
self.pushButton.setGeometry(QtCore.QRect(20, 10, 151, 31))
self.pushButton.setObjectName("pushButton")
self.pushButton_2 = QtWidgets.QPushButton(self.gridGroupBox_4)
self.pushButton_2.setGeometry(QtCore.QRect(20, 60, 151, 31))
self.pushButton_2.setObjectName("pushButton_2")
self.gridGroupBox_5 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_5.setGeometry(QtCore.QRect(210, 190, 171, 101))
self.gridGroupBox_5.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_5.setFlat(False)
self.gridGroupBox_5.setObjectName("gridGroupBox_5")
self.pushButton_6 = QtWidgets.QPushButton(self.gridGroupBox_5)
self.pushButton_6.setGeometry(QtCore.QRect(20, 40, 131, 23))
self.pushButton_6.setObjectName("pushButton_6")
self.pushButton_9 = QtWidgets.QPushButton(self.gridGroupBox_5)
self.pushButton_9.setGeometry(QtCore.QRect(20, 70, 131, 23))
self.pushButton_9.setObjectName("pushButton_9")
self.pushButton_10 = QtWidgets.QPushButton(self.gridGroupBox_5)
self.pushButton_10.setGeometry(QtCore.QRect(20, 10, 131, 23))
self.pushButton_10.setObjectName("pushButton_10")
self.gridGroupBox_6 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_6.setGeometry(QtCore.QRect(390, 10, 161, 171))
self.gridGroupBox_6.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_6.setFlat(False)
self.gridGroupBox_6.setObjectName("gridGroupBox_6")
self.pushButton_14 = QtWidgets.QPushButton(self.gridGroupBox_6)
self.pushButton_14.setGeometry(QtCore.QRect(40, 170, 121, 21))
self.pushButton_14.setObjectName("pushButton_14")
self.label_4 = QtWidgets.QLabel(self.gridGroupBox_6)
self.label_4.setGeometry(QtCore.QRect(30, 10, 101, 16))
self.label_4.setObjectName("label_4")
self.lineEdit = QtWidgets.QLineEdit(self.gridGroupBox_6)
self.lineEdit.setGeometry(QtCore.QRect(80, 40, 61, 20))
self.lineEdit.setObjectName("lineEdit")
self.lineEdit.setText(str(1500))
self.lineEdit_2 = QtWidgets.QLineEdit(self.gridGroupBox_6)
self.lineEdit_2.setGeometry(QtCore.QRect(80, 70, 61, 20))
self.lineEdit_2.setObjectName("lineEdit_2")
self.lineEdit_2.setText(str(6))
self.lineEdit_3 = QtWidgets.QLineEdit(self.gridGroupBox_6)
self.lineEdit_3.setGeometry(QtCore.QRect(80, 100, 61, 20))
self.lineEdit_3.setObjectName("lineEdit_3")
self.lineEdit_3.setText(str(1))
self.label_5 = QtWidgets.QLabel(self.gridGroupBox_6)
self.label_5.setGeometry(QtCore.QRect(20, 40, 51, 16))
self.label_5.setObjectName("label_5")
self.label_6 = QtWidgets.QLabel(self.gridGroupBox_6)
self.label_6.setGeometry(QtCore.QRect(20, 70, 51, 16))
self.label_6.setAlignment(QtCore.Qt.AlignCenter)
self.label_6.setObjectName("label_6")
self.label_7 = QtWidgets.QLabel(self.gridGroupBox_6)
self.label_7.setGeometry(QtCore.QRect(20, 100, 51, 16))

```

```

self.label_7.setAlignment(QtCore.Qt.AlignCenter)
self.label_7.setObjectName("label_7")
self.label_8 = QtWidgets.QLabel(self.gridGroupBox_6)
self.label_8.setGeometry(QtCore.QRect(20, 130, 51, 16))
self.label_8.setAlignment(QtCore.Qt.AlignCenter)
self.label_8.setObjectName("label_8")
self.lineEdit_4 = QtWidgets.QLineEdit(self.gridGroupBox_6)
self.lineEdit_4.setGeometry(QtCore.QRect(80, 130, 61, 20))
self.lineEdit_4.setObjectName("lineEdit_4")
self.lineEdit_4.setText(str(12))
self.gridGroupBox_7 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_7.setGeometry(QtCore.QRect(390, 190, 161, 101))
self.gridGroupBox_7.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_7.setFlat(False)
self.gridGroupBox_7.setObjectName("gridGroupBox_7")
self.lineEdit_5 = QtWidgets.QLineEdit(self.gridGroupBox_7)
self.lineEdit_5.setGeometry(QtCore.QRect(70, 60, 71, 20))
self.lineEdit_5.setReadOnly(True)
self.lineEdit_5.setObjectName("lineEdit_5")
self.lineEdit_5.setText(str(2377.44) + 'b/s')

self.label_9 = QtWidgets.QLabel(self.gridGroupBox_7)
self.label_9.setGeometry(QtCore.QRect(10, 60, 51, 16))
self.label_9.setAlignment(QtCore.Qt.AlignCenter)
self.label_9.setObjectName("label_9")
self.pushButton_11 = QtWidgets.QPushButton(self.gridGroupBox_7)
self.pushButton_11.setGeometry(QtCore.QRect(20, 20, 121, 23))
self.pushButton_11.setObjectName("pushButton_11")
self.gridGroupBox_8 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_8.setGeometry(QtCore.QRect(560, 190, 201, 101))
self.gridGroupBox_8.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_8.setFlat(False)
self.gridGroupBox_8.setObjectName("gridGroupBox_8")
self.comboBox = QtWidgets.QComboBox(self.gridGroupBox_8)
self.comboBox.setGeometry(QtCore.QRect(30, 20, 141, 22))
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem('Packets Loss')
self.comboBox.addItem('Powers and Data')
self.pushButton_12 = QtWidgets.QPushButton(self.gridGroupBox_8)
self.pushButton_12.setGeometry(QtCore.QRect(30, 52, 141, 31))
self.pushButton_12.setObjectName("pushButton_12")
self.listWidget_2 = QtWidgets.QListWidget(self.centralwidget)
self.listWidget_2.setGeometry(QtCore.QRect(560, 300, 201, 81))
self.listWidget_2.setStyleSheet("background-color: rgb(255, 255, 255);")
self.listWidget_2.setObjectName("listWidget_2")
self.gridGroupBox_9 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_9.setGeometry(QtCore.QRect(560, 390, 201, 111))
self.gridGroupBox_9.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")

self.gridGroupBox_9.setFlat(False)
self.gridGroupBox_9.setObjectName("gridGroupBox_9")
self.progressBar = QtWidgets.QProgressBar(self.gridGroupBox_9)
self.progressBar.setGeometry(QtCore.QRect(100, 70, 101, 23))
self.progressBar.setProperty("value", 00)
self.progressBar.setObjectName("progressBar")
self.progressBar_2 = QtWidgets.QProgressBar(self.gridGroupBox_9)
self.progressBar_2.setGeometry(QtCore.QRect(100, 40, 101, 23))
self.progressBar_2.setProperty("value", 0)
self.progressBar_2.setObjectName("progressBar_2")

```

```

self.progressBar_3 = QtWidgets.QProgressBar(self.gridGroupBox_9)
self.progressBar_3.setGeometry(QtCore.QRect(100, 10, 101, 23))
self.progressBar_3.setProperty("value", 0)
self.progressBar_3.setObjectName("progressBar_3")
self.label_12 = QtWidgets.QLabel(self.gridGroupBox_9)
self.label_12.setGeometry(QtCore.QRect(10, 10, 81, 16))
self.label_12.setAlignment(QtCore.Qt.AlignCenter)
self.label_12.setObjectName("label_12")
self.label_13 = QtWidgets.QLabel(self.gridGroupBox_9)
self.label_13.setGeometry(QtCore.QRect(10, 40, 81, 16))
self.label_13.setAlignment(QtCore.Qt.AlignCenter)
self.label_13.setObjectName("label_13")
self.label_14 = QtWidgets.QLabel(self.gridGroupBox_9)
self.label_14.setGeometry(QtCore.QRect(10, 70, 81, 16))
self.label_14.setAlignment(QtCore.Qt.AlignCenter)
self.label_14.setObjectName("label_14")
self.gridGroupBox_10 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_10.setGeometry(QtCore.QRect(770, 10, 201, 171))
self.gridGroupBox_10.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")
self.gridGroupBox_10.setFlat(False)
self.gridGroupBox_10.setObjectName("gridGroupBox_10")
self.label_11 = QtWidgets.QLabel(self.gridGroupBox_10)
self.label_11.setGeometry(QtCore.QRect(50, 10, 101, 16))
self.label_11.setAlignment(QtCore.Qt.AlignCenter)
self.label_11.setObjectName("label_11")
self.pushButton_13 = QtWidgets.QPushButton(self.gridGroupBox_10)
self.pushButton_13.setGeometry(QtCore.QRect(40, 40, 121, 21))
self.pushButton_13.setObjectName("pushButton_13")
self.pushButton_15 = QtWidgets.QPushButton(self.gridGroupBox_10)
self.pushButton_15.setGeometry(QtCore.QRect(40, 80, 121, 21))
self.pushButton_15.setObjectName("pushButton_15")
self.pushButton_16 = QtWidgets.QPushButton(self.gridGroupBox_10)
self.pushButton_16.setGeometry(QtCore.QRect(40, 120, 121, 21))
self.pushButton_16.setObjectName("pushButton_16")
self.gridGroupBox_11 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_11.setGeometry(QtCore.QRect(770, 190, 201, 101))
self.gridGroupBox_11.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")
self.gridGroupBox_11.setFlat(False)
self.gridGroupBox_11.setObjectName("gridGroupBox_11")
self.comboBox_2 = QtWidgets.QComboBox(self.gridGroupBox_11)
self.comboBox_2.setGeometry(QtCore.QRect(30, 50, 141, 22))
self.comboBox_2.setObjectName("comboBox_2")
self.comboBox_2.addItem('None')
self.comboBox_2.addItem('encryption')
self.comboBox_2.addItem('backup and update')
self.comboBox_2.addItem('activity monitoring')
self.radioButton = QtWidgets.QRadioButton(self.gridGroupBox_11)
self.radioButton.setGeometry(QtCore.QRect(30, 20, 141, 17))
self.radioButton.setObjectName("radioButton")
self.gridGroupBox_12 = QtWidgets.QGroupBox(self.centralwidget)
self.gridGroupBox_12.setGeometry(QtCore.QRect(770, 300, 201, 201))
self.gridGroupBox_12.setStyleSheet("background-color: rgb(52, 52, 52);\n"
                                   "background-color: rgb(193, 193, 193);")
self.gridGroupBox_12.setFlat(False)
self.gridGroupBox_12.setObjectName("gridGroupBox_12")
self.label_15 = QtWidgets.QLabel(self.gridGroupBox_12)
self.label_15.setGeometry(QtCore.QRect(70, 10, 61, 16))
self.label_15.setObjectName("label_15")
self.lineEdit_6 = QtWidgets.QLineEdit(self.gridGroupBox_12)

```

```

self.lineEdit_6.setGeometry(QtCore.QRect(110, 40, 71, 20))
self.lineEdit_6.setReadOnly(True)
self.lineEdit_6.setObjectName("lineEdit_6")
self.lineEdit_7 = QtWidgets.QLineEdit(self.gridGroupBox_12)
self.lineEdit_7.setGeometry(QtCore.QRect(110, 80, 71, 20))
self.lineEdit_7.setReadOnly(True)
self.lineEdit_7.setObjectName("lineEdit_7")
self.lineEdit_8 = QtWidgets.QLineEdit(self.gridGroupBox_12)
self.lineEdit_8.setGeometry(QtCore.QRect(110, 120, 71, 20))
self.lineEdit_8.setReadOnly(True)
self.lineEdit_8.setObjectName("lineEdit_8")
self.pushButton_17 = QtWidgets.QPushButton(self.gridGroupBox_12)
self.pushButton_17.setGeometry(QtCore.QRect(30, 160, 141, 23))
self.pushButton_17.setObjectName("pushButton_17")
self.label_16 = QtWidgets.QLabel(self.gridGroupBox_12)
self.label_16.setGeometry(QtCore.QRect(30, 40, 61, 16))
self.label_16.setAlignment(QtCore.Qt.AlignCenter)
self.label_16.setObjectName("label_16")
self.label_17 = QtWidgets.QLabel(self.gridGroupBox_12)
self.label_17.setGeometry(QtCore.QRect(30, 80, 61, 16))
self.label_17.setAlignment(QtCore.Qt.AlignCenter)
self.label_17.setObjectName("label_17")
self.label_18 = QtWidgets.QLabel(self.gridGroupBox_12)
self.label_18.setGeometry(QtCore.QRect(30, 120, 61, 16))
self.label_18.setAlignment(QtCore.Qt.AlignCenter)
self.label_18.setObjectName("label_18")
MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

self.pushButton.clicked.connect(self.on_click)
self.pushButton_2.clicked.connect(self.on_click_2)
self.pushButton_4.clicked.connect(self.on_click_4)
self.pushButton_5.clicked.connect(self.on_click_5)
self.pushButton_5.clicked.connect(self.on_click_5)
self.pushButton_6.clicked.connect(self.on_click_6)
self.pushButton_7.clicked.connect(self.on_click_7)
self.pushButton_8.clicked.connect(self.on_click_8)
self.pushButton_9.clicked.connect(self.on_click_9)
self.pushButton_11.clicked.connect(self.on_click_11)
self.pushButton_12.clicked.connect(self.on_click_12)
self.pushButton_13.clicked.connect(self.open_window_provider_settings)
self.pushButton_15.clicked.connect(self.open_window_task_settings)
self.pushButton_16.clicked.connect(self.open_window_providers_power)
self.radioButton.toggled.connect(self.change_security)
self.comboBox_2.currentIndexChanged.connect(self.on_combo_2)
self.pushButton_17.clicked.connect(self.on_click_17)

def change_security(self):
    self.security = self.comboBox_2.currentText()

def on_click(self):
    if self.providers is not None or len(self.providers) > 0:
        self.providers.clear()
        providers = blocks.generate_providers(int(self.spinBox_2.value()),
int(self.spinBox.value()),
int(self.spinBox_3.value()))
    self.providers = providers

```

```

def on_combo_2(self):
    if self.radioButton.isChecked() is True:
        self.security = self.comboBox_2.currentText()
    else:
        pass

def on_click_2(self):
    if len(self.providers) > 0:
        self.providers.clear()
    providers = blocks.generate_random_providers()
    self.providers = providers

def on_click_4(self):
    if len(self.providers) > 0:
        for provider in self.providers:
            for node in provider.nodes:
                node.set_power_new()
            provider.set_power_new()
        blocks.Graph_builder(self.providers)
    else:
        self.listWidget.clear()
        self.listWidget_2.clear()
        item = QtWidgets.QListWidgetItem('Provider List is empty')
        item.setTextAlignment(QtCore.Qt.AlignCenter)
        self.listWidget_2.addItem(item)

def on_click_5(self):
    if len(self.providers) > 0:
        self.providers = sorted(self.providers, key=lambda prov: prov.provider_id)
        self.listWidget.clear()
        for provider in self.providers:
            self.listWidget.addItem(str('Provider ') + str(provider))
    else:
        self.listWidget_2.clear()
        self.listWidget.clear()
        item = QtWidgets.QListWidgetItem('Provider List is empty')
        item.setTextAlignment(QtCore.Qt.AlignCenter)
        self.listWidget_2.addItem(item)

def on_click_6(self):
    self.tasks.clear()
    if len(self.tasks) <= 0:
        self.tasks = []
    if self.checkBox.isChecked() is True:
        blocks.generate_tasks(self.tasks, int(self.spinBox_5.value()), 'Simple')
    if self.checkBox_2.isChecked() is True:
        blocks.generate_tasks(self.tasks, int(self.spinBox_6.value()), 'Easy')
    if self.checkBox_3.isChecked() is True:
        blocks.generate_tasks(self.tasks, int(self.spinBox_7.value()), 'Medium')
    if self.checkBox_4.isChecked() is True:
        blocks.generate_tasks(self.tasks, int(self.spinBox_8.value()), 'Hard')
    if self.checkBox_5.isChecked() is True:
        blocks.generate_tasks(self.tasks, int(self.spinBox_9.value()), 'Very
Hard')

def on_click_7(self):
    if len(self.tasks) > 0:
        self.listWidget.clear()
        for task in self.tasks:
            self.listWidget.addItem(str(task))
    else:

```

```

self.listWidget_2.clear()
self.listWidget.clear()
item = QtWidgets.QListWidgetItem('Tasks list is empty')
item.setTextAlignment(QtCore.Qt.AlignCenter)
self.listWidget_2.addItem(item)

def on_click_8(self):
    if len(self.providers) > 0:
        self.listWidget_2.clear()
        for provider in self.providers:
            for node in provider.nodes:
                node.set_power_new()
                provider.set_power_new()
        self.time, data = blocks.process_time_calc(self.providers, self.tasks,
self.msp, self.security)
        hours, minutes, seconds = blocks.convert_seconds_to_hms(self.time)
        if hours != 0:
            item = QtWidgets.QListWidgetItem(str(int(hours)) + 'h:' +
str(int(minutes)) + 'm:' + str(int(seconds)) + 's')
            elif minutes != 0:
                item = QtWidgets.QListWidgetItem(str(int(minutes)) + 'm:' +
str(int(seconds)) + 's')
            else:
                item = QtWidgets.QListWidgetItem(str(int(seconds)) + 's')
            item.setTextAlignment(QtCore.Qt.AlignCenter)
            self.listWidget_2.addItem(item)
        # self.progressBar
        # self.progressBar_2
        avg_power = blocks.calculate_providers_avg_power(self.providers)
        cl = avg_power/self.msp * 100
        if cl > 99:
            cl = 99
        self.progressBar_2.setValue(cl)
        p = 0
        i = 0
        for prov in self.providers:
            p += prov.power_level
            i += 1
        if 600 >= self.time >= 301:
            rl = p / i * 1.1
        elif 900 >= self.time >= 601:
            rl = p / i * 1.2
        elif self.time >= 901:
            rl = p / i * 1.3
        elif 300 >= self.time >= 120:
            rl = p / i * 0.8
        else:
            rl = p/i * 0.7
        it = 0
        it = 1 + 0.05 * len(self.providers)
        rl *= it
        if rl > 99:
            rl = 99

        self.progressBar_3.setValue(rl)
        sl = 0
        if self.security is None or self.security == 'None':
            sl = ((rl+cl) * 0.85) / 2
        elif self.security == 'encryption':
            sl = ((rl + cl) * 1.1) / 2
        elif self.security == 'backup and update':

```

```

        sl = ((r1 + c1) * 1.2) / 2
    elif self.security == 'activity monitoring':
        sl = ((r1 + c1) * 1.3) / 2
    if sl > 99:
        sl = 99
    self.progressBar.setValue(sl)
    self.tasks_data = data
    self.sl = sl
    self.cl = c1
    self.rl = r1
else:
    self.listWidget_2.clear()
    self.listWidget.clear()
    item = QtWidgets.QListWidgetItem('Providers list is empty')
    item.setTextAlignment(QtCore.Qt.AlignCenter)
    self.listWidget_2.addItem(item)

def on_click_9(self):
    if len(self.tasks) <= 0:
        self.tasks = []
    self.tasks = blocks.generate_random_tasks()

def on_click_11(self):
    BW = int(self.lineEdit.text())
    N = int(self.lineEdit_2.text())
    D = float(self.lineEdit_3.text())
    SNR = int(self.lineEdit_4.text())
    self.delay = D
    channel = blocks.Channel(BW, SNR, N)
    msp = channel.calc_msp()
    self.msp = msp
    self.lineEdit_5.setText(str(round(self.msp, 2)) + 'b/s')

def on_click_12(self):
    if self.comboBox.currentText() == 'Powers and Data':
        task_times = []
        tasks_ids = []
        providers_id = []
        for provider in self.providers:
            providers_id.append(provider.provider_id)
            task1 = []
            for task in self.tasks:
                time = blocks.process_time_calc_for_stat(task, provider, self.msp,
self.security)
                task1.append(time)
            task_times.append(task1)
        for task in self.tasks:
            tasks_ids.append(str(task.task_did)+str(task.task_id))
        for task in task_times:
            print(task)
        plt.plot(providers_id, task_times, label=tasks_ids)
        plt.xlabel('Provider')
        plt.ylabel('Time')
        plt.title('-')
        plt.legend()
        plt.xticks(providers_id)
        plt.show()
    elif self.comboBox.currentText() == 'Packets Loss':
        blocks.plot_sec(self.security, self.time, self.providers, self.tasks)

```



```

def on_click_17(self):
    quant = len(self.tasks)
    tdc = blocks.calculate_performance(self.providers, self.delay, self.time,
self.tasks_data, quant, self.msp,
                                self.tasks)
    self.lineEdit_6.setText(str(round(tdc, 2)))
    rel = blocks.calculate_reliability(self.time, self.providers, self.rl,
self.cl, self.sl)
    self.lineEdit_7.setText(str(round(rel, 2)))
    sec = blocks.calculate_security(self.security, self.time, self.providers,
self.tasks)
    self.lineEdit_8.setText(str(round(sec, 2)))

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
    self.label.setText(_translate("MainWindow", "Providers Quantity"))
    self.label_2.setText(_translate("MainWindow", "Min Nodes Quantity"))
    self.label_3.setText(_translate("MainWindow", "Max Nodes Quantity"))
    self.checkBox.setText(_translate("MainWindow", "Simple"))
    self.checkBox_2.setText(_translate("MainWindow", "Easy"))
    self.checkBox_3.setText(_translate("MainWindow", "Medium"))
    self.checkBox_4.setText(_translate("MainWindow", "Hard"))
    self.checkBox_5.setText(_translate("MainWindow", "Very Hard"))
    self.pushButton_4.setText(_translate("MainWindow", "Build Graph"))
    self.pushButton_5.setText(_translate("MainWindow", "Display Providers Info"))
    self.pushButton_7.setText(_translate("MainWindow", "Display Tasks Info"))
    self.pushButton_8.setText(_translate("MainWindow", "Forecast Time"))
    self.label_10.setText(_translate("MainWindow", "Show Information:"))
    self.pushButton.setText(_translate("MainWindow", "Build DCS"))
    self.pushButton_2.setText(_translate("MainWindow", "Build Random DCS"))
    self.pushButton_6.setText(_translate("MainWindow", "Build Tasks List"))
    self.pushButton_9.setText(_translate("MainWindow", "Build Random Tasks List"))
    self.pushButton_10.setText(_translate("MainWindow", "Add Tasks to List"))
    self.pushButton_14.setText(_translate("MainWindow", "Forecast Time"))
    self.label_4.setText(_translate("MainWindow", "Channel parameters:"))
    self.label_5.setText(_translate("MainWindow", "Bandwidth"))
    self.label_6.setText(_translate("MainWindow", "Noise"))
    self.label_7.setText(_translate("MainWindow", "Delay"))
    self.label_8.setText(_translate("MainWindow", "SNR"))
    self.label_9.setText(_translate("MainWindow", "MSP"))
    self.pushButton_11.setText(_translate("MainWindow", "Edit Channels
settings"))
    self.pushButton_12.setText(_translate("MainWindow", "Build Graphic
Dependencies"))
    self.label_12.setText(_translate("MainWindow", "Resources Load"))
    self.label_13.setText(_translate("MainWindow", "Channels Load"))
    self.label_14.setText(_translate("MainWindow", "System Load"))
    self.label_11.setText(_translate("MainWindow", "DCS Settings:"))
    self.pushButton_13.setText(_translate("MainWindow", "Providers Settings"))
    self.pushButton_15.setText(_translate("MainWindow", "Task Settings"))
    self.pushButton_16.setText(_translate("MainWindow", "Providers Powers"))
    self.radioButton.setText(_translate("MainWindow", "Data Security Settings"))
    self.label_15.setText(_translate("MainWindow", "DCS Metrics:"))
    self.pushButton_17.setText(_translate("MainWindow", "Calculate DCS Metrics"))
    self.label_16.setText(_translate("MainWindow", "Performance"))
    self.label_17.setText(_translate("MainWindow", "Reliability"))
    self.label_18.setText(_translate("MainWindow", "Security"))

```

```

if __name__ == "__main__":

```

```

import sys

app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.setWindowTitle('Project X')
MainWindow.show()
sys.exit(app.exec_())

from PyQt5 import QtCore, QtGui, QtWidgets
import Blocks as blocks
import random
class Ui_Dialog(object):

    def __init__(self):
        self.index = 0
        self.providers = []
        self.processors = blocks.processors
        self.RAMs = blocks.RAM_memory
        self.nodes = []

    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(311, 374)
        Dialog.setStyleSheet("background-color: qconicalgradient(cx:1,
cy:0.0284091, angle:15, stop:0.210227 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255,
255));\n"
        "")
        self.gridGroupBox = QtWidgets.QGroupBox(Dialog)
        self.gridGroupBox.setGeometry(QtCore.QRect(10, 10, 291, 191))
        self.gridGroupBox.setStyleSheet("background-color: rgb(52, 52, 52);\n"
"background-color: rgb(193, 193, 193);")
        self.gridGroupBox.setObjectName("gridGroupBox")
        self.comboBox_2 = QtWidgets.QComboBox(self.gridGroupBox)
        self.comboBox_2.setGeometry(QtCore.QRect(80, 120, 151, 22))
        self.comboBox_2.setObjectName("comboBox_2")
        self.comboBox_3 = QtWidgets.QComboBox(self.gridGroupBox)
        self.comboBox_3.setGeometry(QtCore.QRect(80, 80, 151, 22))
        self.comboBox_3.setObjectName("comboBox_3")
        self.spinBox_2 = QtWidgets.QSpinBox(self.gridGroupBox)
        self.spinBox_2.setGeometry(QtCore.QRect(180, 150, 51, 21))
        self.spinBox_2.setStyleSheet("background-color: rgb(255, 255, 255);")
        self.spinBox_2.setAlignment(QtCore.Qt.AlignCenter)
        self.spinBox_2.setMinimum(64)
        self.spinBox_2.setMaximum(1024)
        self.spinBox_2.setProperty("value", 233)
        self.spinBox_2.setObjectName("spinBox_2")
        self.label_2 = QtWidgets.QLabel(self.gridGroupBox)
        self.label_2.setGeometry(QtCore.QRect(140, 150, 31, 20))
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.gridGroupBox)
        self.label_3.setGeometry(QtCore.QRect(10, 120, 51, 16))
        self.label_3.setAlignment(QtCore.Qt.AlignCenter)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.gridGroupBox)
        self.label_4.setGeometry(QtCore.QRect(10, 80, 61, 16))
        self.label_4.setAlignment(QtCore.Qt.AlignCenter)

```

```

self.label_4.setObjectName("label_4")
self.comboBox_4 = QtWidgets.QComboBox(self.gridGroupBox)
self.comboBox_4.setGeometry(QtCore.QRect(130, 40, 61, 22))
self.comboBox_4.setObjectName("comboBox_4")
self.label_5 = QtWidgets.QLabel(self.gridGroupBox)
self.label_5.setGeometry(QtCore.QRect(0, -1, 291, 31))
self.label_5.setAlignment(QtCore.Qt.AlignCenter)
self.label_5.setObjectName("label_5")
self.label_6 = QtWidgets.QLabel(self.gridGroupBox)
self.label_6.setGeometry(QtCore.QRect(60, 40, 61, 16))
self.label_6.setAlignment(QtCore.Qt.AlignCenter)
self.label_6.setObjectName("label_6")
self.label = QtWidgets.QLabel(self.gridGroupBox)
self.label.setGeometry(QtCore.QRect(30, 150, 21, 21))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setObjectName("label")
self.spinBox = QtWidgets.QSpinBox(self.gridGroupBox)
self.spinBox.setGeometry(QtCore.QRect(80, 150, 51, 21))
self.spinBox.setStyleSheet("background-color: rgb(255, 255, 255);")
self.spinBox.setAlignment(QtCore.Qt.AlignCenter)
self.spinBox.setMinimum(1)
self.spinBox.setMaximum(6)
self.spinBox.setObjectName("spinBox")
self.lineEdit = QtWidgets.QLineEdit(self.gridGroupBox)
self.lineEdit.setGeometry(QtCore.QRect(240, 80, 41, 20))
self.lineEdit.setAlignment(QtCore.Qt.AlignCenter)
self.lineEdit.setObjectName("lineEdit")
self.lineEdit_2 = QtWidgets.QLineEdit(self.gridGroupBox)
self.lineEdit_2.setGeometry(QtCore.QRect(240, 120, 41, 20))
self.lineEdit_2.setAlignment(QtCore.Qt.AlignCenter)
self.lineEdit_2.setObjectName("lineEdit_2")
self.gridGroupBox_9 = QtWidgets.QGroupBox(Dialog)
self.gridGroupBox_9.setGeometry(QtCore.QRect(10, 210, 291, 151))
self.gridGroupBox_9.setStyleSheet("background-color: rgb(52, 52, 52);\n"
"background-color: rgb(193, 193, 193);")
self.gridGroupBox_9.setFlat(False)
self.gridGroupBox_9.setObjectName("gridGroupBox_9")
self.pushButton_3 = QtWidgets.QPushButton(self.gridGroupBox_9)
self.pushButton_3.setGeometry(QtCore.QRect(40, 20, 211, 31))
self.pushButton_3.setObjectName("pushButton_3")
self.pushButton_2 = QtWidgets.QPushButton(self.gridGroupBox_9)
self.pushButton_2.setGeometry(QtCore.QRect(40, 60, 211, 31))
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_4 = QtWidgets.QPushButton(self.gridGroupBox_9)
self.pushButton_4.setGeometry(QtCore.QRect(40, 100, 211, 31))
self.pushButton_4.setObjectName("pushButton_4")

self.retranslateUi(Dialog)
QtCore.QMetaObject.connectSlotsByName(Dialog)
self.comboBox_4.currentIndexChanged.connect(self.on_combo_4)
self.pushButton_3.clicked.connect(self.on_click_3)
self.pushButton_2.clicked.connect(self.on_click_2)
self.pushButton_4.clicked.connect(self.on_click_4)
self.comboBox_3.currentIndexChanged.connect(self.on_combo_3)
self.comboBox_2.currentIndexChanged.connect(self.on_combo_2)

def on_click_2(self):
    if len(self.nodes) >= 2:
        for proc in self.processors:
            if proc.name == self.comboBox_3.currentText():
                processor = proc

```

```

        for mem in self.RAMs:
            if mem.name == self.comboBox_2.currentText():
                ram = mem
        ssd = int(self.spinBox_2.value())
        ram_q = int(self.spinBox.value())
        item = self.comboBox_4.currentText()
        for node in self.nodes:
            if int(item) == node.node_id:
                index = self.nodes.index(node)
                self.comboBox_4.removeItem(index)
                self.nodes.remove(node)
            node_n = blocks.Node(int(item), processor, ram_q, ram, ssd,
self.index + 1)
            self.nodes.append(node_n)
            self.comboBox_4.addItem(str(node_n.node_id))
        else:
            pass

def on_click_3(self):
    for proc in self.processors:
        if proc.name == self.comboBox_3.currentText():
            processor = proc
    for mem in self.RAMs:
        if mem.name == self.comboBox_2.currentText():
            ram = mem
    ssd = int(self.spinBox_2.value())
    ram_q = int(self.spinBox.value())
    if len(self.nodes) > 0:
        ids = []
        for node in self.nodes:
            ids.append(node.node_id)
        new_id = 1
        while new_id in ids:
            new_id += 1
        node_n = blocks.Node(new_id, processor, ram_q, ram, ssd, self.index
+ 1)

        self.nodes.append(node_n)
        self.comboBox_4.addItem(str(node_n.node_id))

    else:
        node_n = blocks.Node(1, processor, ram_q, ram, ssd, self.index + 1)
        self.nodes.append(node_n)
        self.comboBox_4.addItem(str(node_n.node_id))

def on_combo_2(self):
    for ram in self.RAMs:
        if ram.name == self.comboBox_2.currentText():
            r_ind = blocks.check_container_ram(ram.name)
            self.lineEdit_2.setText(r_ind)
        else:
            pass

def on_combo_3(self):
    for proc in self.processors:
        if proc.name == self.comboBox_3.currentText():
            p_ind = blocks.check_container_proc(proc.name)
            self.lineEdit.setText(p_ind)
        else:
            pass

```

```

def on_combo_4(self):
    for node in self.nodes:
        if node.node_id == int(self.comboBox_4.currentText()):
            p_ind = blocks.check_container_proc(node.processor.name)
            r_ind = blocks.check_container_ram(node.ram.name)
            self.comboBox_2.setCurrentText(str(node.ram.name))
            self.lineEdit_2.setText(r_ind)
            self.comboBox_3.setCurrentText(str(node.processor.name))
            self.lineEdit.setText(p_ind)
            self.spinBox_2.setValue(node.memory)
            self.spinBox.setValue(node.RAM_quant)

def on_click_4(self):
    if len(self.nodes) >= 3:
        item = self.comboBox_4.currentText()
        for node in self.nodes:
            if int(item) == node.node_id:
                index = self.nodes.index(node)
                self.comboBox_4.removeItem(index)
                self.nodes.remove(node)
    else:
        pass

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.label_2.setText(_translate("Dialog", "SSD"))
    self.label_3.setText(_translate("Dialog", "RAMs"))
    self.label_4.setText(_translate("Dialog", "Processors"))
    self.label_5.setText(_translate("Dialog", "Node Settings:"))
    self.label_6.setText(_translate("Dialog", "Node"))
    self.label.setText(_translate("Dialog", "X"))
    self.lineEdit.setText(_translate("Dialog", "None"))
    self.lineEdit.setReadOnly(True)
    self.lineEdit_2.setText(_translate("Dialog", "None"))
    self.lineEdit_2.setReadOnly(True)
    self.pushButton_3.setText(_translate("Dialog", "Add Node"))
    self.pushButton_2.setText(_translate("Dialog", "Edit Node"))
    self.pushButton_4.setText(_translate("Dialog", "Delete Node"))

import Blocks as blocks
from PyQt5 import QtCore, QtGui, QtWidgets
import random

class Ui_Dialog(object):
    def __init__(self):
        self.providers = []
        self.processors = blocks.processors
        self.RAMs = blocks.RAM_memory

    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.setFixedSize(243, 251)
        Dialog.setStyleSheet("background-color: qconicalgradient(cx:1, cy:0.0284091,
angle:15, stop:0.210227 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255, 255));\n"
        "")
        self.gridGroupBox_9 = QtWidgets.QGroupBox(Dialog)

```

```

        self.gridGroupBox_9.setGeometry(QRect(10, 10, 221, 231))
        self.gridGroupBox_9.setStyleSheet("background-color: rgb(52, 52, 52);\n"
"background-color: rgb(193, 193, 193);")
        self.gridGroupBox_9.setFlat(False)
        self.gridGroupBox_9.setObjectName("gridGroupBox_9")
        self.label = QtWidgets.QLabel(self.gridGroupBox_9)
        self.label.setGeometry(QRect(40, 20, 141, 20))
        self.label.setAlignment(Qt.AlignCenter)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.gridGroupBox_9)
        self.label_2.setGeometry(QRect(0, 60, 81, 16))
        self.label_2.setAlignment(Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.comboBox = QtWidgets.QComboBox(self.gridGroupBox_9)
        self.comboBox.setGeometry(QRect(80, 60, 121, 22))
        self.comboBox.setObjectName("comboBox")
        self.pushButton_2 = QtWidgets.QPushButton(self.gridGroupBox_9)
        self.pushButton_2.setGeometry(QRect(20, 150, 181, 31))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton = QtWidgets.QPushButton(self.gridGroupBox_9)
        self.pushButton.setGeometry(QRect(20, 110, 181, 31))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_3 = QtWidgets.QPushButton(self.gridGroupBox_9)
        self.pushButton_3.setGeometry(QRect(20, 190, 181, 31))
        self.pushButton_3.setObjectName("pushButton_3")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.label.setText(_translate("Dialog", "Providers Settings:"))
    self.label_2.setText(_translate("Dialog", "Provider:"))
    self.pushButton_2.setText(_translate("Dialog", "Delete Provider"))
    self.pushButton.setText(_translate("Dialog", "Nodes Settings"))
    self.pushButton_3.setText(_translate("Dialog", "Add New Provider"))

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_TaskSettings(object):

    def __init__(self):
        self.tasks = []

    def setupUi(self, TaskSettings):
        TaskSettings.setObjectName("TaskSettings")
        TaskSettings.resize(190, 161)
        TaskSettings.setStyleSheet("background-color: qconicalgradient(cx:1,
cy:0.0284091, angle:15, stop:0.210227 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255,
255));")
        self.gridGroupBox_12 = QtWidgets.QGroupBox(TaskSettings)
        self.gridGroupBox_12.setGeometry(QRect(10, 10, 171, 141))
        self.gridGroupBox_12.setStyleSheet("background-color: rgb(52, 52,
52);\n"
"background-color: rgb(193, 193, 193);")

```

```

self.gridGroupBox_12.setFlat(False)
self.gridGroupBox_12.setObjectName("gridGroupBox_12")
self.comboBox = QtWidgets.QComboBox(self.gridGroupBox_12)
self.comboBox.setGeometry(QtCore.QRect(80, 40, 71, 22))
self.comboBox.setObjectName("comboBox")
self.label = QtWidgets.QLabel(self.gridGroupBox_12)
self.label.setGeometry(QtCore.QRect(0, 0, 171, 31))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.gridGroupBox_12)
self.label_2.setGeometry(QtCore.QRect(20, 40, 47, 21))
self.label_2.setObjectName("label_2")
self.pushButton = QtWidgets.QPushButton(self.gridGroupBox_12)
self.pushButton.setGeometry(QtCore.QRect(20, 100, 131, 23))
self.pushButton.setObjectName("pushButton")

self.retranslateUi(TaskSettings)
QtCore.QMetaObject.connectSlotsByName(TaskSettings)
self.pushButton.clicked.connect(self.on_click)

def on_click(self):
    item = self.comboBox.currentText()
    for task in self.tasks:
        if item == (str(task.task_id) + str(task.task_id)):
            index = self.tasks.index(task)
            self.comboBox.removeItem(index)
            self.tasks.remove(task)
def retranslateUi(self, TaskSettings):
    _translate = QtCore.QCoreApplication.translate
    TaskSettings.setWindowTitle(_translate("TaskSettings", "Dialog"))
    self.label.setText(_translate("TaskSettings", "Tasks Settings"))
    self.label_2.setText(_translate("TaskSettings", "Task ID"))
    self.pushButton.setText(_translate("TaskSettings", "Delete Task"))

import Blocks as blocks
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Dialog(object):

    def __init__(self):
        self.providers = []

    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.resize(221, 209)
        Dialog.setStyleSheet("background-color: qconicalgradient(cx:1, cy:0.0284091,
angle:15, stop:0.210227 rgba(0, 0, 0, 255), stop:1 rgba(255, 255, 255, 255));")
        self.ProvidersConfig = QtWidgets.QGroupBox(Dialog)
        self.ProvidersConfig.setGeometry(QtCore.QRect(10, 10, 201, 191))
        self.ProvidersConfig.setStyleSheet("background-color: rgb(52, 52, 52);\n"
"background-color: rgb(193, 193, 193);")
        self.ProvidersConfig.setTitle("")
        self.ProvidersConfig.setAlignment(QtCore.Qt.AlignCenter)
        self.ProvidersConfig.setFlat(False)
        self.ProvidersConfig.setCheckable(False)
        self.ProvidersConfig.setObjectName("ProvidersConfig")
        self.pushButton = QtWidgets.QPushButton(self.ProvidersConfig)

```

```

self.pushButton.setGeometry(QRect(30, 150, 141, 23))
self.pushButton.setObjectName("pushButton")
self.lineEdit = QtWidgets.QLineEdit(self.ProvidersConfig)
self.lineEdit.setGeometry(QRect(110, 230, 61, 20))
self.lineEdit.setObjectName("lineEdit")
self.comboBox_2 = QtWidgets.QComboBox(self.ProvidersConfig)
self.comboBox_2.setGeometry(QRect(90, 40, 81, 22))
self.comboBox_2.setObjectName("comboBox_2")
self.label = QtWidgets.QLabel(self.ProvidersConfig)
self.label.setGeometry(QRect(0, -1, 201, 31))
self.label.setAlignment(Qt.AlignCenter)
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.ProvidersConfig)
self.label_2.setGeometry(QRect(20, 40, 61, 21))
self.label_2.setAlignment(Qt.AlignCenter)
self.label_2.setObjectName("label_2")
self.label_4 = QtWidgets.QLabel(self.ProvidersConfig)
self.label_4.setGeometry(QRect(10, 80, 71, 20))
self.label_4.setAlignment(Qt.AlignCenter)
self.label_4.setObjectName("label_4")
self.horizontalSlider = QtWidgets.QSlider(self.ProvidersConfig)
self.horizontalSlider.setGeometry(QRect(30, 110, 141, 22))
self.horizontalSlider.setProperty("value", 60)
self.horizontalSlider.setOrientation(Qt.Horizontal)
self.horizontalSlider.setObjectName("horizontalSlider")
self.lineEdit_2 = QtWidgets.QLineEdit(self.ProvidersConfig)
self.lineEdit_2.setGeometry(QRect(90, 80, 81, 20))
self.lineEdit_2.setAlignment(Qt.AlignCenter)
self.lineEdit_2.setReadOnly(True)
self.lineEdit_2.setObjectName("lineEdit_2")

self.retranslateUi(Dialog)
QtCore.QMetaObject.connectSlotsByName(Dialog)
self.comboBox_2.currentIndexChanged.connect(self.on_combo_2)
self.pushButton.clicked.connect(self.on_click)

def on_click(self):
    index = int(self.comboBox_2.currentText()) - 1
    coef = self.horizontalSlider.value()
    self.providers[index].set_power(int(coef))
    self.lineEdit_2.setText(str(round(self.providers[index].power_avg, 2)) +
'Mb/s')

def on_combo_2(self):
    index = int(self.comboBox_2.currentText()) - 1

self.lineEdit_2.setText(str(round(blocks.calculate_provider_power_avg(self.providers[
index]), 2)) + 'Mb/s')

def retranslateUi(self, Dialog):
    _translate = QtCore.QCoreApplication.translate
    Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
    self.pushButton.setText(_translate("Dialog", "Change Load"))
    self.label.setText(_translate("Dialog", "Providers Powers:"))
    self.label_2.setText(_translate("Dialog", "Provider"))
    self.label_4.setText(_translate("Dialog", "Power"))

```


ДОДАТОК В

Оптимізаційна багатofакторна модель оцінки показників функціонування розподіленої комп'ютерної системи

Виконав:

студент групи КА-95

Новохатський Д.Є.

Науковий керівник:

проф. Яковлева А.П.

Предмет дослідження:

Підвищення ефективності в організації управління ресурсами розподіленої комп'ютерних систем (РКС), які включають в себе розподілені сервери, мережеву інфраструктуру, а також бази даних та їх взаємодію.

Мета роботи:

Аналіз існуючих методів оцінки функціонування розподілених комп'ютерних систем (РКС), а також моделей оптимізації управління її ресурсами. А також програмна реалізація тестового прикладу розробленої моделі для дослідження результатів впливу параметрів на її ефективність.

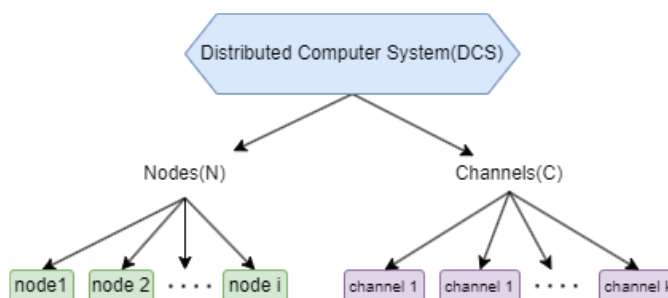
Використані моделі:

Було запропоновано та створено власну модель оцінки та її архітектуру на основі мережцентричного підходу для ефективного розподілу ресурсів системи відповідно до складності запитів, що вона виконує.

Основні етапи дослідження

- Визначення що таке розподілена комп'ютерна система (РКС), та які її типи існують, приклади використання в сучасному світі.
- Огляд особливостей РКС, а також опис існуючих моделей за засобів оцінки її функціонування.
- Розгляд та порівняння основних підходів та існуючих моделей для оптимізації. Аналіз механізмів управління ресурсами РКС.
- Формалізація загального вигляду РКС та визначення головних параметрів, що впливають на її функціонування.
- Побудова оптимізаційної моделі та опис архітектури тестової системи.
- Програмна реалізація та приклад оптимізації керування ресурсами системи на основі аналізу її показників та результатів.
- Аналіз отриманих результатів та висновки щодо ефективності застосування розробленої моделі. Оцінювання її переваг та недоліків, знайдених у ході дослідження.

Загальний вигляд розподіленої комп'ютерної системи (PKC/DCS)



$$\text{Distributed Computer System} = DCS(N, C)$$

$$\text{Nodes} = N(\text{node}_1, \text{node}_2, \text{node}_3, \dots, \text{node}_i)$$

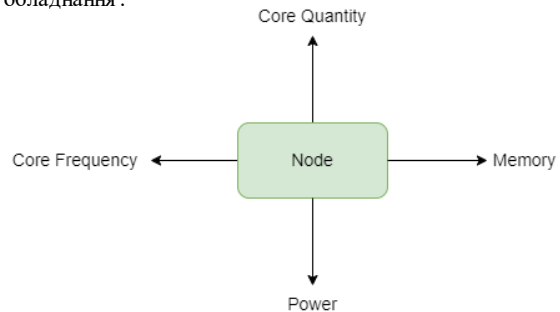
$$\text{Channels} = C(\text{channel}_1, \text{channel}_2, \text{channel}_3, \dots, \text{channel}_k)$$

Параметри вузлів та каналів

$$Nodes = N(node_1, node_2, node_3, \dots, node_i)$$

Кожен з вузлів складається з параметрів його обладнання:

- частота процесору (Core Frequency);
- кількість ядер (Core Quantity);
- потужність (Power);
- обсяг пам'яті (Memory).



де кожен окремий вузол має таку залежність:

$$node = n(CF, CQ, P, M)$$

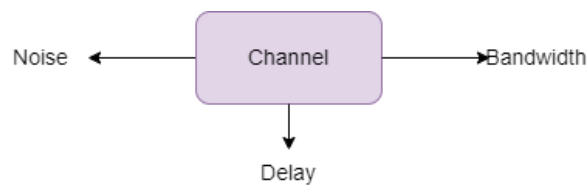
Параметри вузлів та каналів

$$Channels = C(channel_1, channel_2, channel_3, \dots, channel_k)$$

Кожен з каналів складається з параметрів його обладнання:

- Затримка (Delay);
- пропускна здатність (Bandwidth);
- Шум (Noise)

де кожен окремий канал має таку залежність: $channel = ch(NS, BW, D)$



Математична модель

$$PDCS(A, S, n) = F(DCS_{speed}, R(CL, RL, Time), MSP(BW, S, N), S(Am, Pr, n))$$

Захищеність системи

$$Security = S(Am, Pr, n)$$

де

Am – обсяг оброблюємих даних;

Pr – частина даних без втрати пакетів;

n – період циклу у секундах.

Надійність системи

$$Reliability = R(CL, RL, Time)$$

де

CL - рівень завантаженості каналів;

RL - рівень завантаженості ресурсів;

Time – прогнозований час виконання .

Оцінка продуктивності системи

Продуктивність системи будемо розглядати та обчислювати за формулою:

$$DCS_{speed} = \frac{t_s}{t_{DCS}}$$

де:

$$t_{DCS} = \frac{diff}{n\omega_{avg}} \cdot \lambda^2 + 0,5 * c * delay$$

$$c = \frac{n(n-1)}{4}$$

$$diff = \frac{Am}{\min(\omega_{avg}, MSP)}$$

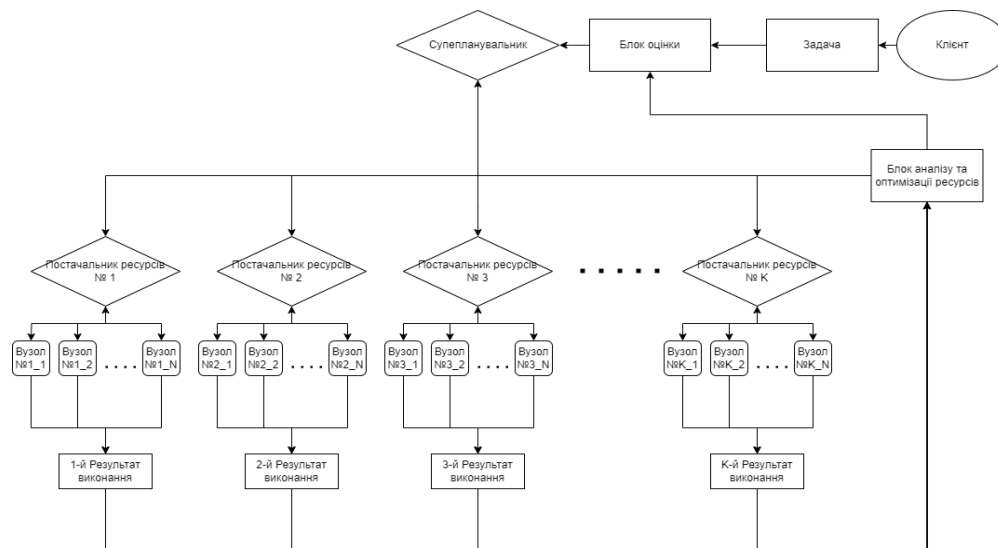
$$MSP = B * \log_2(1 + S/N)$$

$$t_s = \frac{dif}{\omega_{avg}}$$

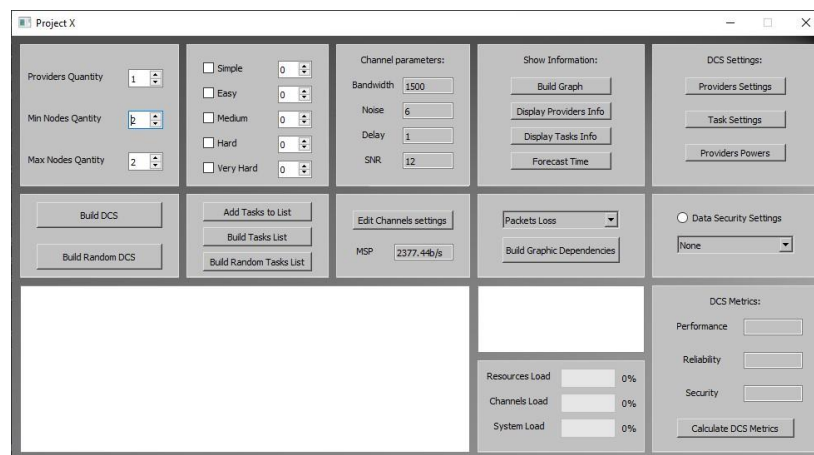
$$\omega_{avg} = \frac{\sum_{i=1}^n \omega_i}{n}$$

$$\omega_i = \frac{CQ_i * CF_i + M_i}{P_i}$$

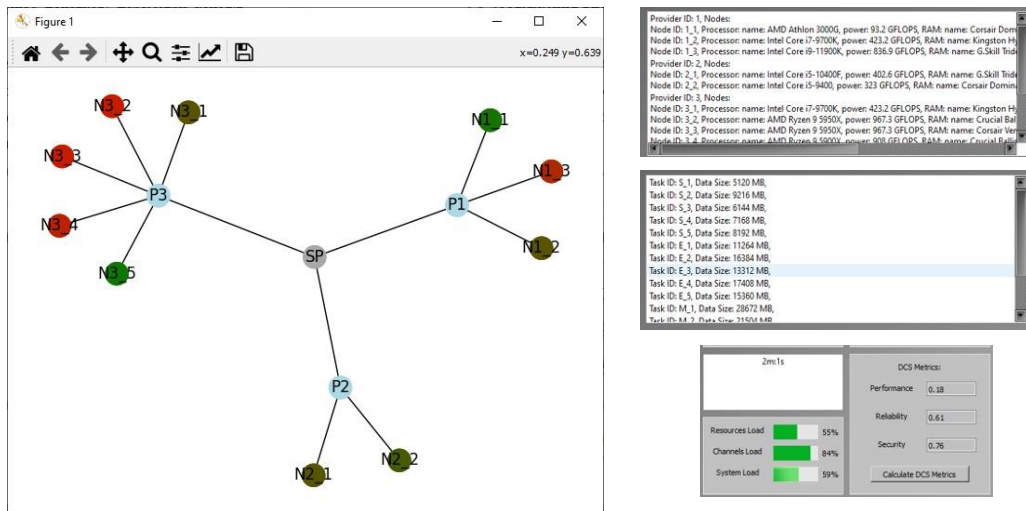
Блок-схема вигляду розподіленої комп'ютерної системи (РКС) на основі межцентричного підходу



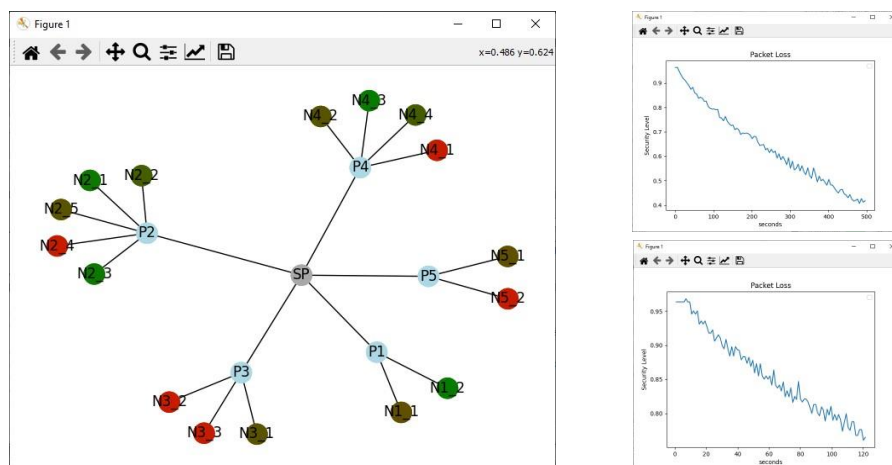
Огляд програмної реалізації



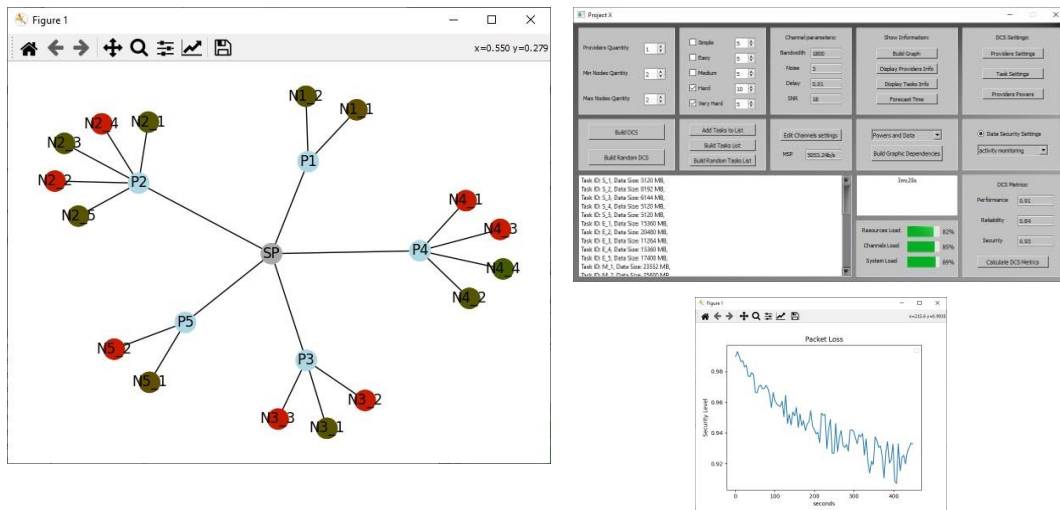
Дослідження впливу параметрів на тестовому прикладі



Дослідження впливу параметрів на тестовому прикладі



Дослідження впливу параметрів на тестовому прикладі



Висновки

В ході даного дослідження було оглянуто основні поняття розподілених комп'ютерних систем, їх типів, а також методів оцінювання її функціонування. Окрім цього, було розглянуто та зроблено порівняльний аналіз методів щодо оптимізації управління ресурсами в таких системах.

Також було побудовано власну модель та її архітектуру на основі мережцентричного підходу для оцінки функціонування розподіленої комп'ютерної системи. Після чого на тестовому прикладі проведено експериментальне дослідження впливу параметрів цієї системи на її оцінки.

Де з отриманих результатів можемо зробити висновки, що при масштабуванні системи для виконання більш складних запитів, потрібно збільшувати й максимальну пропускну здатність системи, зменшуючи шум та затримку, а також збільшуючи кількість сигналів та ширину смуги. Також гарною оптимізацією є заміна слабких компонентів вузлів постачальників, а також корегування їх потужностей відповідно до можливостей системи. Ще необхідним є застосування додаткових рівнів безпеки, тому що при збільшенні часу роботи нашої системи відповідно до нових задач, збільшилась й кількість атак, та втрата даних досягала більше половини. Відповідно показникам, можна сказати що на основі тестового прикладу, мета оптимізації розподілу ресурсів виконана відповідно до отриманих з моделі оцінок.

Подальші вдосконалення

- Розробка моделі черг з пріоритетами для оптимізації завантаженості ресурсів.
- Більш детальна архітектура для оцінювання метрик захищеності.
- Додавання параметрів, з метою більш точного оцінювання функціонування системи.
- Застосування нейронних мереж для підбору параметрів та збільшення виборки можливих компонентів системи.
- Розширення математичної моделі, додаванням нових оцінок, або вдосконаленням вже існуючих.

Дякую за увагу!