

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель аналізу спектральних даних астрономічних
об'єктів
(тема)

Виконав:
студент II курсу, групи СПм-22-3
Ніколаєнко Д.С.
(прізвище, ініціали)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Філімончук Т.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ
(підпис) Коваленко А.А.
(прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____ комп'ютерної інженерії та управління _____

Кафедра _____ електронних обчислювальних машин _____

Рівень вищої освіти _____ другий (магістерський) _____

Спеціальність _____ 123 «Комп'ютерна інженерія» _____
(код і повна назва)

Тип програми _____ освітньо-наукова _____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____ Системне програмування _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

“ _____ ” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту _____ Ніколаєнку Дмитру Сергійовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Модель аналізу спектральних даних астрономічних об'єктів _____

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст _____

2. Термін подання студентом роботи до екзаменаційної комісії _____ 15 червня 2024 р. _____

3. Вхідні дані до роботи _____ 1) методи обробки спектральних даних астрономічних об'єктів;
_____ 2) програмні середовища для реалізації алгоритмів паралельної функціональності;
_____ 3) бібліотеки математичних перетворень; 4) бібліотеки машинного зору. _____

4. Перелік питань, що потрібно опрацювати у роботі _____

- _____ 1) аналіз предметної області;
- _____ 2) ознайомлення з актуальним програмних забезпеченням;
- _____ 3) розробка моделі аналізу спектральних даних за допомогою паралельних обчислень;
- _____ 4) реалізація моделі у середовищі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____

Слайд-презентація – 15 слайдів _____

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	02.04.24-08.04.24	
2	Ознайомлення з актуальних програмним забезпеченням	09.04.24-16.04.24	
3	Розробка моделі аналізу спектральних даних за допомогою розподілених обчислень	17.04.24-22.04.24	
4	Реалізація моделі у середовищі	23.04.24-06.05.24	
5	Проведення експериментів	07.05.24-23.05.24	
6	Оформлення матеріалів кваліфікаційної роботи	24.05.24-03.06.24	
7	Подання кваліфікаційної роботи керівникові та її попередній захист	04.06.24-07.06.24	
8	Подання кваліфікаційної роботи на рецензування	08.06.24-12.06.24	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____
(підпис)

доц. Філімончук Т.В.
(посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 66 с., 5 рис., 2 дод., 20 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЬ, СПЕКТРАЛЬНИЙ АНАЛІЗ, МОДУЛЬ, ВЕЛИКІ ДАНІ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ, ШВИДКОДІЯ.

Метою даної роботи є побудова моделі обробки спектральних даних астрономічних об'єктів з використанням паралельних обчислень.

Об'єктом дослідження є процес обробки спектральних даних, які було отримано від астрономічних об'єктів.

Предметом дослідження є існуючі на даний час моделі та методи обробки спектральних даних.

Результати – запропонована модель обробки спектральних даних астрономічних об'єктів, яка рекомендує використовувати паралельний алгоритм обробки та спеціалізоване програмне забезпечення, а саме середовище CUDA та бібліотеку OpenCV для зменшення часу обробки великих масивів даних та більш раціонального використання потужностей обчислювальних ресурсів.

Область застосування: сфери астрономії, фізики, інформаційних технологій.

ABSTRACT

Master's thesis: 66 pages, 5 figures, 2 appendices, 20 sources.

INFORMATION SYSTEM, MODEL, SPECTRAL ANALYSIS,
MODULE, BIG DATA, PARALLEL COMPUTING, SPEED.

The method of this work is the construction of a model for processing spectral data of astronomical objects using parallel computing.

The object of research is the process of processing spectral data obtained from astronomical objects.

The subject of the study is currently existing models and methods of spectral data processing.

The results are a proposed model for processing spectral data of astronomical objects, which recommends the use of a parallel processing algorithm and specialized software, namely the CUDA environment and the OpenCV library to reduce the processing time of large data sets and more rational use of computing resources.

Field of application: spheres of astronomy, physics, information technologies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	7
ВСТУП	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	9
1.1 Актуальні засоби обробки.....	9
2 РОЗРОБКА МОДЕЛІ АНАЛІЗУ СПЕКТРАЛЬНИХ ДАНИХ.....	25
2.1 Програмна архітектура CUDA. Інструментарій розробки.....	25
2.2 Аналоги CUDA від AMD	27
2.3 Машинне навчання	29
2.4 Бібліотека OpenCV.....	31
2.5 Бібліотека CFITSIO	32
3 РЕАЛІЗАЦІЯ АЛГОРИТМУ АНАЛІЗУ СПЕКТРАЛЬНИХ ДАНИХ.....	38
3.1 Доступ до спектральних даних	38
3.2 Підготовка робочого середовища.....	40
3.2.1 Встановлення CUDA	40
3.2.2 Встановлення OpenCV.....	41
3.2.3 Встановлення бібліотеки CFITSIO.....	42
3.3 Реалізація алгоритму спектральних даних	43
3.3.1 Метод послідовного аналізу даних на CPU	44
3.3.2 Метод паралельного аналізу даних на GPU	44
3.3.3 Методи візуалізації результатів.....	45
3.3.4 Головний метод програми.....	47
3.4 Результати роботи алгоритму	48
ВИСНОВКИ.....	51
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	53
ДОДАТОК А Графічний матеріал кваліфікаційної роботи.....	55
ДОДАТОК Б Лістинг коду	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс програмування застосунків (англ., Application Programming Interface)

CPU – центральний процесор (англ., Central Processing Unit)

CUDA – уніфікована архітектура пристрою (англ., Compute Unified Device Architecture)

GPU – графічний процесор (англ., Graphics Processing Unit)

IDE – інтегроване середовище розробки (англ., Integrated Development Environment)

IRAF – програма аналізу зображень (англ., Image Reduction and Analysis Facility)

ВСТУП

Людство завжди прагнуло дослідити та проаналізувати небо, виявити закономірності, явища небесних об'єктів, дослідити самі об'єкти. З плином часу прості дослідження неба стали повноцінною наукою – астрономією. Астрономія вже давно перестала бути виключно наукою оптичних спостережень. Минуле століття дало великий поштовх розвитку всіх наукових сфер, які використовуються в наш час як інструментарій для досліджень неба та його явищ: фізика, хімія, географія, комп'ютерні технології.

Створення радіотелескопу в середині 20-го століття дало великий поштовх розвитку астрономії, так як тепер науковці мають змогу бачити випромінювання об'єктів не тільки в оптичному діапазоні, а й в інших. Проте, науковці стикаються з іншою проблемою: обробкою даних, які було отримано. Одними з типів даних, які отримуються з радіотелескопів є спектральні дані. У зв'язку з цим, сучасна астрономія все більше спирається на передові комп'ютерні технології та інноваційні методи обробки даних [1-5]. Використання графічних процесорів (GPU) стало революційним кроком у цьому напрямку. GPU дозволяють проводити обчислення з високим ступенем паралелізму, що кардинально змінює підходи до обробки великих обсягів даних, що отримані від небесних спостережень.

Для оптимізації процесу аналізу спектральних даних використовуються комп'ютерні технології [6-7]. На даний момент науковці досліджують існуючі моделі, програми та алгоритми обробки, намагаючись створити новий, більш оптимізований підхід до аналізу даних.

Метою кваліфікаційної роботи є побудова моделі аналізу спектральних даних астрономічних об'єктів за допомогою паралельних обчислень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальні засоби обробки

На даний час найпопулярнішим середовищем для аналізу та обробки спектральних даних астрономічних об'єктів є середовище Image Reduction and Analysis Facility (IRAF) [8], яке було створено у 80-х роках для швидкої та оптимальної обробки даних, отриманих від астрономічних об'єктів. Мовою програмування середовища IRAF є Python, а для математичних обчислень використовується бібліотека Astropy [9]. Середовище являє собою потужний програмний пакет, розроблений для використання в астрономії та астрофізиці. IRAF надає зручні інструменти для виконання широкого спектру задач: від базових обчислень та обробки зображень до більш складних операцій, таких як управління астрономічними координатами, часом та моделюванням.

Бібліотека Astropy надає розширені можливості для аналізу спектральних даних, що є невід'ємною частиною сучасних астрономічних досліджень. Вона дозволяє вченим легко обробляти, аналізувати та інтерпретувати інформацію, яка була отримана в наслідок астрономічних спостережень.

Середовище IRAF до сих пір підтримується, незважаючи на використання послідовних алгоритмів обробки. Такий підхід накладає значні обмеження на швидкодію обробки даних, особливо в контексті «любительської» астрономії. Необхідна велика потужність від комп'ютерної системи, щоб обробляти такий великий потік даних в реальному часі. Обробка одного файлу FITS на доволі посередніх комп'ютерах може зайняти більше кількох хвилин, що є доволі довгим процесом, враховуючи те, що один такий файл містить дані про об'єкт в конкретний момент часу.

Головною задачею моделі, що пропонується, є прискорення швидкості обробки даних у форматі FITS: максимально швидко пройтися по масиву да-

них, застосувати до кожного елементу математичні функції (аналогічно бібліотеці *Astropy*) та зробити це з використанням розподілених обчислень.

1.2 Спектральний аналіз

Спектральний аналіз є однією з ключових методик у науках про Землю, астрономії, фізиці та хімії. Він дозволяє вивчати властивості матеріалів та об'єктів через їхні спектри, тобто розподіл інтенсивності випромінювання або поглинання по різних довжинах хвиль. Ця методика надає детальну інформацію про хімічний склад, фізичні властивості та динамічні процеси в досліджуваних об'єктах. Спектральний аналіз використовується в широкому спектрі наук та технологій, включаючи астрономію, де він є незамінним інструментом для вивчення космічних об'єктів.

Спектр – це графік або діаграма (рисунок 1.1), що показує, як інтенсивність світла (або іншого випромінювання) змінюється в залежності від довжини хвилі або частоти. Світло, яке ми бачимо, є лише невеликою частиною електромагнітного спектру, який включає також радіохвилі, інфрачервоне, ультрафіолетове, рентгенівське та гамма-випромінювання. Кожен тип випромінювання має свої унікальні властивості та використовується для різних цілей у науці та техніці.

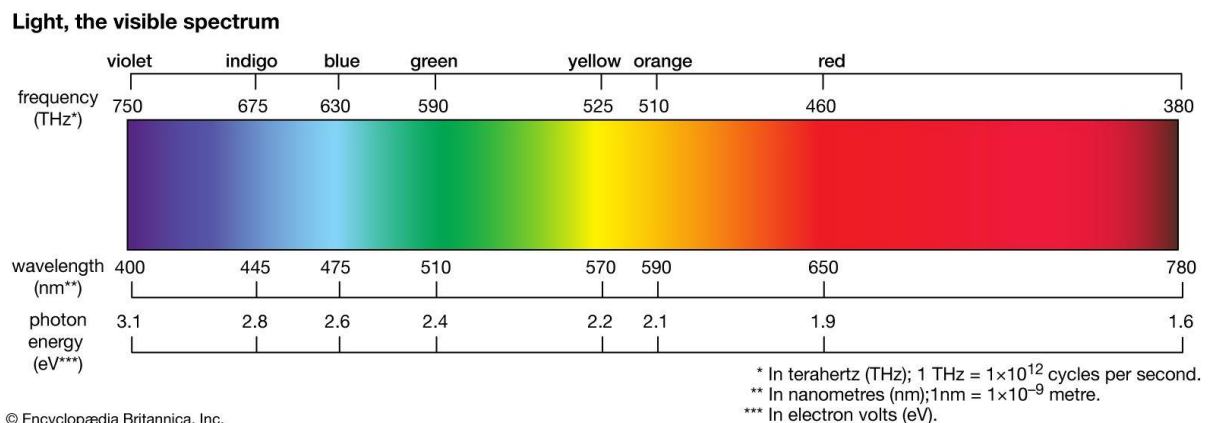


Рисунок 1.1 – Спектр видимого світла

Спектральний аналіз зазвичай проводиться за допомогою спектрометрів – приладів, які розділяють випромінювання на його складові

довжини хвиль та вимірюють інтенсивність кожної з них. Спектрометри можуть бути оптичними, які працюють з видимим світлом, або працювати в інших діапазонах, таких як інфрачервоне чи ультрафіолетове випромінювання. Результати вимірювань спектрометра зазвичай представляються у вигляді спектральних ліній або безперервних спектрів, які можуть бути використані для визначення хімічного складу, температури, щільності та інших фізичних параметрів об'єктів, що досліджуються.

У астрономії спектральний аналіз використовується для вивчення зірок, планет, галактик та інших небесних тіл. Коли світло від зірки або іншого об'єкта проходить через спектрометр, воно розділяється на спектр, який містить інформацію про елементи, що знаходяться у верхніх шарах зірки. Кожен хімічний елемент має унікальні спектральні лінії, які можна ідентифікувати та використовувати для визначення складу зірки. Наприклад, присутність ліній водню в спектрі зірки свідчить про наявність водню в її атмосфері.

Крім того, спектральний аналіз дозволяє визначити швидкість руху об'єктів завдяки ефекту Доплера. Коли об'єкт рухається від спостерігача, його спектральні лінії зміщуються в бік червоного кінця спектру (червоне зміщення), а коли наближається – у бік синього кінця (синє зміщення). Це зміщення дозволяє вимірювати швидкість руху зірок, галактик та інших об'єктів, що є ключовим інструментом у дослідженні розширення Всесвіту та руху галактик.

У хімії та фізиці спектральний аналіз використовується для ідентифікації речовин та визначення їхніх концентрацій. Наприклад, у хімічних лабораторіях атомна спектроскопія використовується для аналізу складу зразків шляхом збудження атомів та вимірювання випромінювання, яке вони випускають. Це дозволяє точно визначити кількість кожного елемента в зразку, що є важливим для контролю якості та дослідження нових матеріалів.

Одним з важливих аспектів аналізу є використання спектральних ліній

для визначення фізичних умов в об'єктах. Інтенсивність та форма спектральних ліній залежать від температури, тиску, магнітного поля та інших параметрів середовища, в якому вони утворюються. Аналізуючи ці лінії, можна отримати інформацію про фізичні умови в різних частинах зірок, планетних атмосфер, галактик та інших космічних об'єктів.

Спектр також має важливе значення у вивченні міжзоряного середовища. Молекули та іони в міжзоряному просторі випромінюють та поглинають світло на специфічних довжинах хвиль, що дозволяє визначати їхній склад та розподіл у галактиках. Це є ключовим інструментом для розуміння процесів зореутворення, еволюції галактик та хімічного збагачення Всесвіту.

У технічних додатках спектральний аналіз використовується в різних галузях, включаючи медицину, екологію, матеріалознавство та багато інших. Наприклад, в медицині спектральна діагностика використовується для виявлення патологій в органах та тканинах, аналізу крові та інших біологічних зразків. У екології спектральний аналіз дозволяє виявляти забруднювачі у воді, повітрі та ґрунті, а в матеріалознавстві – вивчати властивості нових матеріалів та покриттів.

1.3 Паралельні обчислення

Паралельні обчислення є важливою складовою сучасних наукових досліджень та технологій. Вони дозволяють значно підвищити швидкість і ефективність обробки великих обсягів даних та виконання складних обчислювальних задач шляхом одночасного виконання декількох процесів. Цей підхід є особливо корисним у галузях, які потребують інтенсивних обчислень, таких як астрономія, фізика, хімія, біоінформатика та інженерія.

Суть паралельних обчислень полягає у розподілі задачі на менші підзадачі, які можуть виконуватися одночасно на декількох процесорах або ядрах процесора. Це дозволяє ефективніше використовувати обчислювальні ресурси та скорочувати час виконання задачі. Паралельні обчислення можуть

бути реалізовані на різних рівнях: від багатоядерних процесорів у персональних комп'ютерах до великих суперкомп'ютерів та розподілених обчислювальних систем.

Одним з ключових аспектів паралельних обчислень є правильний розподіл задач між процесорами та забезпечення їхньої синхронізації. Це може бути досягнуто за допомогою різних моделей паралельного програмування, таких як моделі з розподіленою пам'яттю та моделі з загальною пам'яттю. У моделях з розподіленою пам'яттю кожен процесор має свою власну пам'ять, і обмін даними між процесорами здійснюється через комунікаційні мережі. У моделях з загальною пам'яттю всі процесори мають доступ до спільної пам'яті, що спрощує обмін даними, але вимагає складнішої синхронізації.

У контексті астрономічних досліджень паралельні обчислення є особливо корисними для обробки великих обсягів даних, які генеруються сучасними телескопами та іншими науковими інструментами. Наприклад, обробка зображень великого поля зору, аналіз спектральних даних та моделювання астрофізичних процесів можуть займати значний час, якщо виконувати їх послідовно. Використання паралельних обчислень дозволяє значно скоротити час обробки таких даних та отримувати результати у більш стислі терміни.

Однією з популярних технологій для реалізації паралельних обчислень є використання графічних процесорів (GPU). GPU були розроблені для обробки графічних даних, але їхня архітектура з великою кількістю ядер також ідеально підходить для паралельних обчислень. Сучасні GPU можуть мати сотні або навіть тисячі ядер, що дозволяє виконувати масивні паралельні обчислення з високою продуктивністю. Використання GPU для наукових обчислень стає все більш популярним завдяки таким платформам, як CUDA від NVIDIA та OpenCL, які надають засоби для програмування паралельних обчислень на GPU.

Хмарні платформи, такі як Amazon Web Services, Google Cloud Platform

та Microsoft Azure, надають доступ до великих обчислювальних ресурсів через інтернет. Це дозволяє дослідникам використовувати високопродуктивні обчислювальні ресурси для своїх задач без необхідності інвестувати у власне обладнання. Хмарні обчислення забезпечують гнучкість у використанні ресурсів, що дозволяє масштабувати обчислювальні потужності відповідно до потреб проєкту.

Паралельні обчислення використовуються для обробки даних великих наукових проєктів, таких як проєкт з розшифровки геному людини або проєкти з дослідження космічного мікрохвильового фонового випромінювання. Ці проєкти генерують величезні обсяги даних, які потребують ефективної обробки та аналізу. Використання паралельних обчислень дозволяє виконувати такі задачі швидше та з меншою кількістю помилок.

Проте паралельні обчислення мають свої виклики та обмеження. Одним з основних викликів є складність програмування паралельних алгоритмів. Необхідно правильно розподілити задачі між процесорами, забезпечити ефективну синхронізацію та обмін даними між ними. Це вимагає додаткових зусиль від програмістів та може бути складним для тих, хто не має досвіду у паралельному програмуванні. Крім того, паралельні алгоритми можуть бути менш ефективними, якщо задачі не можуть бути легко розпаралелені або якщо існують значні залежності між підзадачами.

Іншим викликом є забезпечення ефективного використання обчислювальних ресурсів. Наприклад, наявність великої кількості ядер у процесорі не завжди гарантує високу продуктивність, якщо не вдається повністю завантажити всі ядра. Це може бути пов'язано з обмеженнями на обмін даними або необхідністю синхронізації між процесорами. Крім того, ефективність паралельних обчислень може залежати від конкретної архітектури апаратного забезпечення та конфігурації системи. Сама система також може впливати на швидкодію обчислень через свої налаштування доступу.

1.4 Технологія зберігання даних Big Data

Наш світ дуже складний, ця складність призводить до генерації постійно зростаючої кількості даних, які потрібно зберігати та аналізувати. Швидкість генерації даних не має ознак уповільнення. Різноманітність даних поширюється на нові області, які ніколи не були доступні для аналізу. Взаємодія між людьми, які використовують медіа-платформи, автоматизація процесів та агрегація даних, що надходять з різних джерел, створюють Інтернет речей (IoT), який не лише підключає датчики до існуючих речей, а створює ринок для нових підключених речей. Усі ці зв'язані речі генерують дані, що створює приголомшливу кількість великих даних, які називаються Big Data [10].

Не завжди можливо зібрати всі доступні дані в межах проєкту чи рішення. Обсяг даних, які можна зібрати, визначається можливостями датчиків, мереж, комп'ютерів та іншого обладнання. Це також визначається необхідністю, наприклад, перевірити правильність вирівнювання етикетки кожної пляшки, що рухається через високошвидкісну лінію розливу напоїв. У цьому випадку важливі дані кожної пляшки. Іншому датчику, наприклад, датчику вологості на кукурудзяному полі, не потрібно повідомляти про виміряне значення вологості кожні десять секунд: кожних п'яти-десяти хвилин може бути достатньо. Не всі зібрані дані можна використовувати в оригінальному вигляді тому що можливо були зібрані сторонні, неправильні або неправдиві дані. Щоб ці дані були корисними, їх потрібно очистити.

Очищення – це видалення непотрібних даних, зміна неправильних даних та заповнення відсутніх даних. Для очищення даних прийнято використовувати програмний код. Це досягається пошуком критеріїв або їх відсутністю та маніпулюванням даними, доки аномалії не зникнуть. Після очищення даних їх легше шукати, аналізувати та візуалізувати. За допомогою аналізу даних можна дізнатися цікаву інформацію та виявити тенденції. Це часто призводить до нових запитів, які ще не реалізовані. Якщо з деякого

набору даних не можна виявити жодної додаткової цінності, можна експериментувати з тим, як вони організовані та представлені.

Експоненціальне зростання даних створило нову сферу інтересів у технологіях і бізнесі під назвою «Великі дані». Загалом, набір даних або бізнес-проблема належить до класифікації великих даних, якщо дані настільки великі або складні, що їх неможливо зберігати, обробляти й аналізувати за допомогою традиційних підходів до зберігання та аналізу.

Особливості що відрізняють дані від великих даних:

- обсяг переданих та збережених даних: поточна задача полягає в тому, щоб знайти способи максимально ефективної обробки зростаючого обсягу даних;

- швидкість – це величина, з якою формуються дані. Наприклад, дані, отримані мільярдом акцій, що торгуються на Нью-Йоркській фондовій біржі, неможливо просто зберегти для подальшого аналізу;

- різноманітність – тип даних, який рідко знаходиться в стані, що ідеально готовий для обробки та аналізу. Велику частку таких даних становлять неструктуровані, які за оцінками, становлять від 70 до 90% даних у світі;

- правдивість – процес запобігання неточного опису наборів даних. Наприклад, люди можуть створити онлайн-акаунт та використовувати неправдиву контактну інформацію. Підвищена достовірність збору даних зменшує кількість необхідного очищення даних.

Багато компаній розуміють, що є сенс інвестувати в технології Big Data, щоб залишатися конкурентоспроможними на своєму ринку. Зазвичай доступ до даних обмежується кількома відповідальними людьми в організації. Компанії стрімко переходять на використання технологій Big Data для розвитку бізнес-аналітики. Парадигма великих даних складається з розподілу систем даних між горизонтально пов'язаними незалежними ресурсами для досягнення масштабованості, необхідної для ефективної обробки великих наборів даних. Це є горизонтальна масштабованість, яка

відрізняється від вертикальної тим, що вона не намагається додати більше процесорної потужності та пам'яті до існуючих машин. Ці інфраструктури дозволяють багатьом користувачам одночасно отримувати безперешкодний та безпечний доступ до даних. Одним із таких прикладів є тисячі онлайн-покупців або гравців мобільних ігор.

1.5 Середовище обробки даних IRAF

IRAF (Image Reduction and Analysis Facility) – це потужне програмне середовище, розроблене для обробки та аналізу астрономічних даних, яке було створене Національною обсерваторією оптичної астрономії США (NOAO) і включає безліч інструментів для редукції та аналізу астрономічних зображень (рисунок 1.2). IRAF став стандартом для багатьох астрономічних установ та дослідників у всьому світі. Цей інструмент забезпечує широкий спектр функцій для редукції даних, аналізу зображень та спектрів, що робить його незамінним у роботі астрономів.

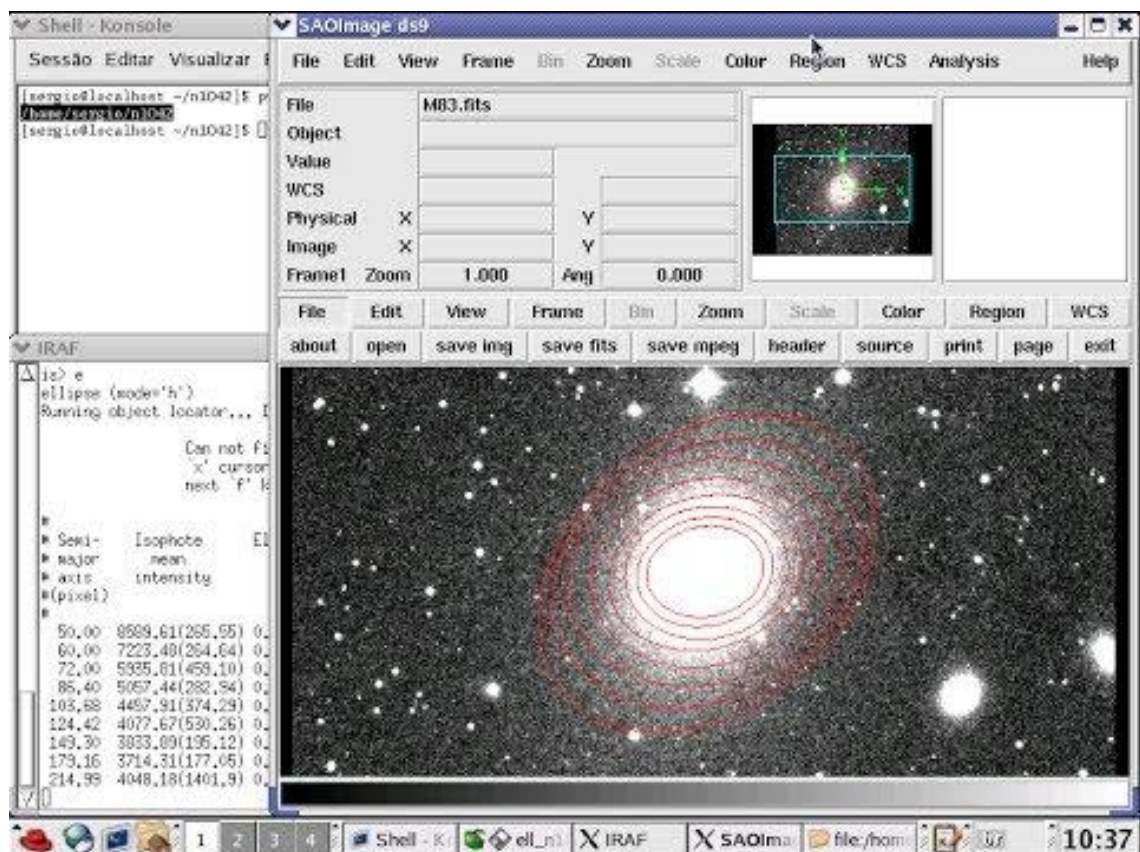


Рисунок 1.2 – Інтерфейс середовища IRAF

IRAF включає численні пакети та модулі, що дозволяють виконувати різні задачі, пов'язані з обробкою даних. Одним з головних аспектів IRAF є підтримка формату FITS (Flexible Image Transport System), який є стандартом для зберігання астрономічних зображень та спектральних даних. Формат FITS дозволяє зберігати як зображення, так і багатовимірні масиви даних, що робить його ідеальним для астрономічних спостережень.

Обробка спектральних даних у IRAF є однією з найважливіших функцій цього середовища. Спектральні дані дозволяють астрономам вивчати фізичні властивості об'єктів, такі як їхній склад, температура, швидкість та інші характеристики.

Середовище IRAF включає декілька спеціалізованих пакетів для роботи зі спектральними даними. Наприклад, пакет ONEDSPEC містить інструменти для обробки одновимірних спектрів. Він дозволяє виконувати редукцію спектрів, включаючи калібрування, нормалізацію, корекцію зсувів та багато іншого. Користувачі можуть також аналізувати спектральні лінії, визначати їхні параметри та будувати графіки спектрів.

Ще один важливий пакет – це TWODSPEC, який призначений для роботи з двовимірними спектральними даними, які було отримано за допомогою спектрографів. Цей пакет дозволяє виконувати складні обчислення та аналіз двовимірних спектрів, включаючи виправлення ефектів атмосферної дисперсії та інших спотворень.

IRAF також забезпечує засоби для інтерактивного аналізу спектральних даних. Користувачі можуть використовувати інтерфейс командного рядка для виконання різних операцій, а також графічні інтерфейси для візуалізації та аналізу даних, що дозволяє ефективно працювати з великими обсягами спектральних даних, виконувати комплексні обчислення та отримувати наочні результати.

Середовище IRAF забезпечує високу ступінь автоматизації, що допомагає користувачам створювати скрипти для автоматичного виконання рутинних задач. Це зручно для обробки великих наборів даних, що надходять

з обсерваторій та дозволяє економити час та зусилля дослідників.

IRAF також підтримує інтеграцію з іншими програмними інструментами та мовами програмування. Наприклад, користувачі можуть використовувати мови програмування, такі як Python, для створення додаткових скриптів та модулів, які розширюють можливості IRAF, що робить IRAF гнучким та потужним інструментом для астрономічних досліджень.

Завдяки своїм можливостям IRAF залишається актуальним для багатьох астрономів та дослідників у всьому світі. Підтримка формату FITS і можливості для обробки та аналізу спектральних даних роблять його незамінним у багатьох дослідницьких проєктах.

1.6 Бібліотека Astropy

Бібліотека Astropy є однією з найпотужніших та найпопулярніших бібліотек для астрономічних обчислень у середовищі Python. Вона була розроблена для спрощення та стандартизації наукових розрахунків, пов'язаних з астрономією та астрофізикою. Astropy об'єднує широкий спектр інструментів та модулів, які дозволяють астрономам та дослідникам ефективно працювати з великими обсягами даних, виконувати складні обчислення та аналізувати результати.

Бібліотека включає в себе різноманітні модулі, що дозволяють вирішувати широкий спектр задач, вона пропонує зручні засоби для роботи з астрономічними координатами та часом. Модуль `Astropy.Coordinates` дозволяє виконувати перетворення між різними системами координат, а `Astropy.Time` забезпечує підтримку високоточної роботи з часом. Крім того, модуль `Astropy.Units` дозволяє використовувати фізичні одиниці вимірювання та забезпечує їх автоматичне перетворення, що зручно для уникнення помилок, пов'язаних з неправильними одиницями під час обчислень.

Astropy підтримує різні формати астрономічних даних, включаючи

FITS.

Модуль `Astropy.Io.Fits` дозволяє читати, записувати та обробляти FITS-файли. Крім того, `Astropy.Table` забезпечує потужні засоби для роботи з табличними даними, дозволяючи легко створювати, редагувати та аналізувати таблиці, що є незамінним інструментом для обробки результатів спостережень.

`Astropy` включає модулі для моделювання астрономічних явищ та процесів. Наприклад, модуль `Astropy.Modeling` дозволяє створювати та підганяти моделі до спостережних даних. Також модуль `Astropy.Specutils` забезпечує засоби для роботи зі спектральними даними, включаючи інструменти для аналізу та візуалізації спектрів.

`Astropy` є проектом з відкритим кодом, що активно розвивається завдяки великій та активній спільноті розробників та користувачів. Спільнота `Astropy` організовує регулярні зустрічі та конференції, де обговорюються нові функції, плани розвитку та обмінюються досвідом. Однією з головних цілей `Astropy` є створення єдиного стандарту для астрономічних обчислень у Python. Це досягається через узгодження інтерфейсів, стандартів та форматів даних між різними модулями та бібліотеками, що дозволяє користувачам легко переходити між різними інструментами без необхідності вчитися новим підходам.

Бібліотека `Astropy` є важливим інструментом для сучасних астрономів та дослідників, що дозволяє ефективно виконувати широкий спектр задач, пов'язаних з обробкою та аналізом астрономічних даних. Її інтеграція з іншими бібліотеками Python та можливість використання у поєднанні з середовищем IRAF робить її ще більш потужною та гнучкою. Завдяки активній спільноті та постійному розвитку, `Astropy` продовжує залишатися на передовій сучасних астрономічних досліджень, забезпечуючи користувачів новими інструментами та можливостями.

`Astropy` не є прямим замінником IRAF, але багато функцій, доступних у IRAF, також реалізовані в `Astropy`, що робить її зручним інструментом для

тих, хто вже знайомий з IRAF. Перехід від IRAF до Astropy дозволяє використовувати сучасні засоби програмування, більш гнучкі та потужні методи обробки даних, а також інтеграцію з іншими бібліотеками Python.

1.7 Формат FITS

Формат FITS є стандартом для зберігання, передачі та обробки астрономічних даних. Цей формат був розроблений Національною обсерваторією оптичної астрономії США (NOAO) у кінці 1970-х років і з тих пір став невід'ємною частиною інфраструктури астрономічних досліджень. FITS використовується для зберігання різноманітних типів даних, включаючи зображення, спектри, таблиці та багатовимірні масиви даних. Його гнучкість, надійність та здатність зберігати метадані разом з основними даними роблять його ідеальним форматом для астрономічних спостережень.

Формат складається з набору структурованих блоків даних, відомих як "записи" або "карти". Кожен запис складається з заголовка (header) та даних (data). Заголовок містить ключі та значення, що описують зміст та структуру даних, а також метадані, такі як дата спостереження, параметри телескопа, калібрувальна інформація тощо. Дані можуть бути зображеннями, спектральними даними або таблицями.

Спектральні дані зберігаються у FITS у вигляді одновимірних або багатовимірних масивів. Одновимірні спектри зазвичай представлені як масиви інтенсивностей у залежності від довжини хвилі або частоти. Ці масиви можуть бути збережені у первинному масиві даних (primary data array) або в розширеннях (extensions), таких як таблиці бінарних даних (binary table extensions). Багатовимірні спектри, отримані з допомогою спектрографів, можуть бути збережені у вигляді двовимірних або тривимірних масивів, де одна з вимірів відповідає довжині хвилі, а інші – просторовим координатам.

Заголовок FITS-файлу містить ключові слова, які описують структуру та зміст спектральних даних. Наприклад, ключові слова "NAXIS" та

"NAXISn" визначають кількість вимірів та розмірність масивів даних відповідно. Ключові слова "CRVALn", "CRPIXn" та "CDELTn" використовуються для визначення початкової довжини хвилі, центрального пікселя та кроку між пікселями для кожного виміру, що дозволяє точно відтворити спектральну шкалу та інші параметри, необхідні для аналізу даних.

FITS також підтримує зберігання метаданих, пов'язаних зі спектральними даними. Це може включати інформацію про калібрування, корекцію зсувів, шумові характеристики та інші параметри, що є важливими для правильної інтерпретації спектрів. Метадані можуть бути збережені у заголовках або в окремих розширеннях, що забезпечує гнучкість та зручність у роботі з великими обсягами даних.

Формат FITS забезпечує високу ступінь сумісності та стандартизації у зберіганні та обробці астрономічних даних. Завдяки його широкому прийняттю та підтримці багатьма програмними інструментами, астрономи можуть легко обмінюватися даними та співпрацювати у міжнародних проєктах. Програми, такі як IRAF та Astropy, мають вбудовану підтримку формату FITS, що дозволяє їм читати, записувати та обробляти спектральні дані без необхідності у додаткових перетвореннях.

Однією з основних переваг формату FITS є його гнучкість. Він підтримує різні типи даних та може бути адаптований до специфічних потреб користувачів. Наприклад, користувачі можуть створювати власні ключові слова для зберігання специфічної інформації про свої спостереження або експерименти. Це дозволяє розширювати функціональність формату та забезпечувати його відповідність новим вимогам та стандартам.

1.8 Недоліки використання IRAF та бібліотеки Astropy

Незважаючи на численні переваги, використання IRAF та бібліотеки Astropy має свої недоліки, які можуть обмежувати ефективність та зручність їх використання. Один з найбільших недоліків цих програмних інструментів

є відсутність можливості паралельної обробки даних та використання послідовних алгоритмів, що обмежує швидкість та продуктивність обробки великих обсягів астрономічних даних.

IRAF був розроблений в часи, коли обчислювальні ресурси були обмеженими та паралельна обробка не була широко розповсюдженою. Тому багато алгоритмів у IRAF базуються на послідовній обробці, що призводить до значних затримок при обробці великих наборів даних. Наприклад, спектральні дані, що отримуються з великих телескопів, можуть включати тисячі спектрів, кожен з яких потребує окремої обробки. Послідовна обробка таких обсягів даних може займати значний час, що обмежує ефективність досліджень та аналізу.

Astropy, хоча і є сучаснішою бібліотекою, також має обмеження щодо паралельної обробки. Багато алгоритмів, реалізованих у Astropy, працюють послідовно, що може бути неефективним при обробці великих масивів даних. Наприклад, обробка FITS-файлів з великими спектральними зображеннями може займати значний час, якщо вона виконується послідовно. Хоча Python підтримує багатопотоковість, використання паралельних обчислень у Astropy потребує додаткових зусиль від користувачів, що може бути складним для тих, хто не має відповідного досвіду.

Ще одним недоліком IRAF є його застарілий інтерфейс. Багато інструментів та модулів IRAF розроблялися десятиліттями тому і не враховують сучасні тенденції в розробці програмного забезпечення. Інтерфейс командного рядка, хоча і є потужним, може бути складним для недосвідчених користувачів. Відсутність графічного інтерфейсу ускладнює роботу з програмою, особливо для користувачів, які не мають досвіду роботи з командним рядком. Це може призвести до помилок та значних затримок при виконанні задач, що потребують точності та уваги до деталей.

Astropy, хоча і має більш сучасний інтерфейс, також може бути складною для розуміння. Величезна кількість модулів та функцій може бути заплутаною для тих, хто тільки починає працювати з нею. Документація є

досить обширною, але не завжди інтуїтивно зрозумілою. Користувачам без досвіду може бути складно знайти необхідну інформацію або приклади використання, що може уповільнити їх роботу та знизити ефективність.

Обидва інструменти мають обмеження щодо інтеграції з сучасними методами машинного навчання та аналізу даних. Хоча існують способи інтеграції Astropy з іншими бібліотеками Python, такими як TensorFlow або PyTorch, ці методи часто потребують значних зусиль та знань. Відсутність вбудованих засобів для використання сучасних технологій машинного навчання та аналізу даних обмежує можливості дослідників.

Крім того, обидва інструменти мають обмежені можливості для роботи з великими даними та високопродуктивними обчисленнями. Вони не оптимізовані для використання на суперкомп'ютерах або в хмарних обчислювальних середовищах, що обмежує їх продуктивність при обробці великих обсягів даних. Це особливо важливо для сучасних астрономічних проєктів, які генерують величезні масиви даних, що потребують високопродуктивних обчислювальних ресурсів.

У підсумку, хоча IRAF та Astropy є потужними інструментами для астрономічних досліджень, вони мають низку недоліків, які обмежують їх ефективність та зручність використання. Відсутність можливості паралельної обробки даних, застарілий інтерфейс та складнощі для новачків є основними проблемами, які потребують вирішення. Інтеграція з сучасними методами машинного навчання та аналізу даних, підтримка нових форматів даних та інструментів обробки, а також оптимізація для високопродуктивних обчислювальних середовищ можуть значно покращити можливості цих інструментів та зробити їх більш ефективними та зручними для користувачів.

2 РОЗРОБКА МОДЕЛІ АНАЛІЗУ СПЕКТРАЛЬНИХ ДАНИХ

2.1 Програмна архітектура CUDA. Інструментарій розробки

Складність обчислювань вимагає різкого збільшення ресурсів та швидкодії комп'ютерів. Найбільш перспективним напрямком збільшення швидкості розв'язку є широке впровадження ідей паралелізму у роботу обчислювальних систем. Сьогодні спроектовано та випробувано сотні різних комп'ютерів, що використовують у своїй архітектурі той чи інший вид паралельної обробки даних. Основна складність при проектуванні паралельних програм – забезпечення правильної послідовності взаємодій між різними обчислювальними процесами, а також координація ресурсів, що розділяються між ними.

Для розробки моделі було обрано найпопулярнішу програмну архітектуру від Nvidia CUDA [11, 12].

CUDA – це програмно-апаратна архітектура паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів NVIDIA. При використанні цієї технології слід орієнтуватися на наступну термінологію:

- пристрій (device) – сама відеокарта, графічний процесор (GPU), який виконує команди центрального процесора;
- хост (host) – центральний процесор (CPU), який запускає різні завдання на пристрої, виділяє пам'ять;
- ядро (kernel) – завдання, що буде виконуватися на GPU.

CUDA дозволяє програмістам реалізовувати на спеціальному спрощеному діалекті мови програмування C алгоритми, що виконуються на графічних процесорах Nvidia, та включати спеціальні функції в текст програми на C. Архітектура CUDA дає розробнику можливість на свій розсуд організовувати доступ до набору інструкцій GPU й управляти його пам'яттю. Дана технологія підтримує декілька мов програмування. Серед них є Java,

Python, C/C++.

Для розподіленого алгоритму обробки спектральних даних слід використовувати наступні інструменти:

- бібліотеку OpenCV [13] відкритого коду, яка призначена для надання спільної інфраструктури для застосунків комп'ютерного зору та прискорення використання перцепції машин у комерційних продуктах;

- середовище Microsoft Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft, що використовується для розробки комп'ютерних програм, веб-сайтів, веб-додатків, веб-сервісів та мобільних додатків;

- мову програмування C++, яка являє собою високорівневу мову з можливостями низькорівневого програмування.

Бібліотека OpenCV містить в собі алгоритми обробки даних з використанням розподілених обчислень, які можна використовувати за допомогою середовища CUDA. Також бібліотека містить математичні утиліти, аналогічні Astropy.

Набір інструментів CUDA містить бібліотеки, засоби налагодження та оптимізації, компілятор, документацію та бібліотеку часу виконання для розгортання програм. Він містить компоненти, які підтримують глибоке навчання, лінійну алгебру, обробку сигналів та паралельні алгоритми.

Загалом бібліотеки CUDA підтримують усі сімейства графічних процесорів NVIDIA, але найкраще працює на останньому поколінні, наприклад V100, який може бути втричі швидшим, ніж P100, для навчальних робочих навантажень глибокого навчання, як показано нижче; A100 може додати ще 2-кратне прискорення. Використання однієї чи кількох бібліотек – це найпростіший спосіб скористатися перевагами графічних процесорів, за умови, що потрібні алгоритми реалізовано у відповідній бібліотеці.

Швидке перетворення Фур'є (ШПФ) є одним із основних алгоритмів обробки сигналів, він перетворює сигнал (наприклад, звукову форму) на спектр частот. cuFFT – це БПФ із прискоренням GPU.

Основною мовою програмування на CUDA є мова C++, що являє собою розширення однієї з версій мови C. Вона набула широкої відомості завдяки своїй функціональності та легкості в написанні коду, так як сама мова перш за все призначена для об'єктно-орієнтованого програмування, що дозволяє за необхідності розписати програму на класи і об'єкти та декларувати взаємодію між ними.

Великою перевагою мови C++ серед інших мов є легкість завантаження та використання програмних бібліотек. Основну філософію дизайну C та C++ можна описати як "довіряти програмісту". Мову C++ розроблено, щоб надати програмісту високий ступінь свободи робити те, що він хоче.

C++ відмінно підходить в ситуаціях, коли потрібна висока продуктивність і точний контроль над пам'яттю та іншими ресурсами.

C++ широко використовується для:

- комп'ютерних розважальних аплікацій;
- систем реального часу;
- високоефективних фінансових програм;
- графічних додатків та моделювання;
- офісних програм;
- вбудованого програмного забезпечення;
- обробки аудіо та відео;
- штучного інтелекту та нейронних мереж.

2.2 Аналоги CUDA від AMD

Завдяки своїй ефективності, широкому набору інструментів та бібліотек, а також активній підтримці та розширеній документації, CUDA стала домінуючим стандартом у галузі високоефективних обчислень та машинного навчання. Однак існують альтернативи від компанії AMD, зокрема ROCm (Radeon Open Compute) та OpenCL (Open Computing Language), які надають схожі можливості для роботи з графічними процесорами AMD.

ROCm або Radeon Open Compute [14] є відкритою платформою для високопродуктивних обчислень на графічних процесорах AMD. ROCm надає набір інструментів та бібліотек для паралельних обчислень, аналогічно до CUDA. Вона включає компілятор HCC (Heterogeneous Compute Compiler), який підтримує мовні розширення для C++ та дозволяє писати програми для GPU AMD. Крім того, ROCm підтримує HIP (Heterogeneous-Compute Interface for Portability), який дозволяє легко переносити код, написаний для CUDA, на GPU AMD. Це значно спрощує процес адаптації програмного забезпечення для різних платформ.

OpenCL є ще однією альтернативою CUDA, розробленою компанією Khronos Group. Це відкритий стандарт для паралельних обчислень на різних платформах, включаючи GPU, CPU та інші процесори. OpenCL надає гнучкий та потужний інструмент для розробки високопродуктивних обчислювальних додатків, що можуть працювати на різних апаратних архітектурах. Важливою перевагою OpenCL є його сумісність з багатьма виробниками апаратного забезпечення, що дозволяє розробникам створювати кросплатформні рішення.

Незважаючи на існування альтернатив, CUDA залишається більш популярною з кількох причин. По-перше, CUDA була однією з перших платформ для паралельних обчислень на GPU та завдяки цьому здобула значну перевагу в часі. NVIDIA активно підтримує та розвиває CUDA, забезпечуючи стабільність, ефективність та регулярні оновлення платформи.

По-друге, екосистема інструментів та бібліотек CUDA є надзвичайно розвиненою. NVIDIA надає широкий спектр бібліотек, таких як cuDNN для глибокого навчання, cuBLAS для лінійної алгебри та багато інших, що значно спрощують розробку високопродуктивних додатків. Ці бібліотеки оптимізовані для роботи з GPU NVIDIA та забезпечують високу продуктивність.

По-третє, документація та підтримка спільноти грають важливу роль у популярності CUDA. NVIDIA забезпечує детальну документацію, навчальні

матеріали, приклади коду та інструменти для розробників. Крім того, існує велика спільнота користувачів та розробників CUDA, що дозволяє легко знайти відповіді на питання, обмінюватися досвідом та отримувати підтримку.

Іншою важливою причиною є стратегічне партнерство NVIDIA з багатьма компаніями та дослідницькими установами. NVIDIA активно співпрацює з провідними університетами, дослідницькими інститутами та компаніями в галузі технологій, що сприяє поширенню та впровадженню CUDA у різних проєктах та дослідженнях. Це створює ефект сніжної кулі, коли більше користувачів обирають CUDA через її популярність і підтримку.

Продуктивність та стабільність CUDA є ключовими факторами, що роблять її привабливою для розробників. Продуктивність GPU NVIDIA, оптимізованих для роботи з CUDA, є високою, що дозволяє досягати значних результатів у високопродуктивних обчисленнях та машинному навчанні. Стабільність та надійність платформи також відіграють важливу роль у забезпеченні безперебійної роботи додатків.

Таким чином, хоча існують потужні альтернативи від AMD, такі як ROCm та OpenCL, CUDA залишається більш популярною завдяки своїй розвиненій екосистемі, високій продуктивності, стабільності та активній підтримці з боку NVIDIA та спільноти користувачів. Це робить її першочерговим вибором для багатьох розробників та дослідників у галузі високопродуктивних обчислень.

2.3 Машинне навчання

Машинне навчання – це підгалузь штучного інтелекту, яка широко визначається як здатність машини імітувати інтелектуальну поведінку людини. Системи штучного інтелекту використовуються для виконання складних завдань у спосіб, подібний до того, як люди вирішують проблеми.

Мета штучного інтелекту полягає в тому, щоб створити комп'ютерні моделі, які демонструють "розумну поведінку", як люди, за словами Бориса

Каца, головного наукового співробітника та керівника InfoLab Group в CSAIL. Це означає машини, які можуть розпізнавати візуальну сцену, розуміти текст, написаний природною мовою або виконувати дію у фізичному світі.

Машинне навчання починається з даних – чисел, фотографій або тексту, наприклад, банківських транзакцій, фотографій людей або навіть хлібобулочних виробів, записів про ремонт, даних часових рядів із датчиків або звітів про продажі. Дані збираються та готуються для використання як навчальні дані або інформація, на основі якої навчатиметься модель машинного навчання. Чим більше даних, тим краще програма.

Деякі дані беруться з навчальних даних, які використовуються як дані оцінювання, які перевіряють, наскільки точна модель машинного навчання, коли їй показуються нові дані. Результатом є модель, яку можна використовувати в майбутньому з різними наборами даних.

Успішні алгоритми машинного навчання можуть робити різні речі, функція системи машинного навчання може бути описовою, тобто система використовує дані, щоб пояснити, що сталося; прогнозною, тобто система використовує дані, щоб передбачити, що станеться; або наказовою, що означає, що система використовуватиме дані, щоб робити пропозиції щодо того, які дії вжити.

Існує три підкатегорії машинного навчання:

- контрольовані моделі машинного навчання навчаються за допомогою позначених наборів даних, що дозволяє моделям навчатися та ставати точнішими з часом. Наприклад, алгоритм буде навчатися за зображеннями собак та інших речей, які позначені людьми і після навчання машина навчиться способом самостійно ідентифікувати зображення собак. Кероване машинне навчання є найпоширенішим типом, який використовується сьогодні;

- у неконтрольованому машинному навчанні програма шукає шаблони в немаркованих даних. Машинне навчання без нагляду може знаходити

шаблони або тенденції, які люди явно не шукають. Наприклад, програма машинного навчання без контролю може переглядати дані про онлайн-продажі та ідентифікувати різні типи клієнтів, які здійснюють покупки;

- машинне навчання підкріплення навчає машини методом проб та помилок виконувати найкращі дії, встановлюючи систему винагороди. Навчання з підкріпленням може навчити моделі грати в ігри або навчити автономні транспортні засоби керувати, повідомляючи машині, коли вона прийняла правильні рішення, що допомагає їй з часом навчитися, які дії їй слід виконати.

2.4 Бібліотека OpenCV

OpenCV – бібліотека комп'ютерного зору та машинного навчання з відкритим кодом. OpenCV було створено, щоб забезпечити загальну інфраструктуру для програм комп'ютерного зору та прискорити використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом Apache 2, OpenCV дозволяє компаніям легко використовувати та змінювати код.

Бібліотека містить понад 2500 оптимізованих алгоритмів, що включає повний набір як класичних, так і найсучасніших алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання обличч, ідентифікації об'єктів, класифікації дій людей у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок із стереокамер, з'єднання зображень для отримання високої роздільної здатності зображення цілої сцени, знаходити подібні зображення в базі даних зображень, видаляти червоні очі із зображень, зроблених за допомогою спалаху, стежити за рухами очей, розпізнавати пейзаж та встановлювати маркери для накладання на нього доповненої реальності тощо. OpenCV має понад 47 тисяч користувачів спільноти та оціночна кількість її завантажень перевищує 18 мільйонів. Бібліотека широко використовується компаніями,

дослідницькими групами та державними органами.

Поряд із такими відомими компаніями, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато стартапів, таких як Applied Minds, VideoSurf та Zeitera, які широко використовують OpenCV. Застосування OpenCV охоплює широкий діапазон: від з'єднання зображень вуличного перегляду, виявлення вторгнень у відеоспостереження в Ізраїлі, моніторингу обладнання в Китаї, допомоги роботам у навігації та підбиранні об'єктів у Willow Garage, виявлення нещасних випадків утоплення в басейні в Європі, використання інтерактивного мистецтва в Іспанія та Нью-Йорк, перевірка злітно-посадкових смуг на наявність сміття в Туреччині, перевірка етикеток на продуктах на фабриках по всьому світу та швидке виявлення обличч у Японії.

Бібліотека має інтерфейси C++, Python, Java та MATLAB і підтримує Windows, Linux, Android та Mac OS. OpenCV здебільшого орієнтується на програми бачення в реальному часі та використовує переваги інструкцій MMX та SSE, якщо вони доступні. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які створюють або підтримують ці алгоритми. OpenCV написано на мові C++ на основі шаблонного інтерфейсу, який бездоганно працює з контейнерами STL.

2.5 Бібліотека CFITSIO

CFITSIO [15] є важливою бібліотекою для роботи з файлами формату FITS, які використовуються для зберігання астрономічних даних. Розроблена та підтримувана NASA, ця бібліотека забезпечує зручний та ефективний доступ до даних у форматі FITS, що є критично важливим для астрономічних досліджень.

Бібліотека дозволяє читати, записувати та обробляти FITS-файли з високою продуктивністю, вона написана на мові програмування C, що робить її швидкою та сумісною з багатьма іншими мовами через обгортки.

Вона підтримує різні типи даних, такі як зображення, спектри та таблиці, що робить її універсальним інструментом для роботи з астрономічними даними.

Однією з ключових переваг CFITSIO є її здатність обробляти великі обсяги даних. Вона використовує ефективне управління пам'яттю та буферизацію для зменшення кількості операцій введення-виведення, що дозволяє швидко обробляти дані навіть у великих обсягах. Крім того, бібліотека підтримує компресію даних, що зменшує розмір файлів та прискорює передачу даних.

CFITSIO має детально документований API, який полегшує інтеграцію бібліотеки в різні проєкти. Документація включає опис функцій, приклади коду та рекомендації щодо використання, що допомагає розробникам швидко освоїти бібліотеку та використовувати її можливості.

Бібліотека широко використовується у багатьох астрономічних проєктах. Вона є основним інструментом у таких програмних пакетах, як HEASoft та FTOOLS, які використовуються для обробки даних високої енергії в астрофізиці. CFITSIO також застосовується в обсерваторіях для обробки даних з телескопів.

Важливою особливістю CFITSIO є її сумісність з іншими мовами програмування через обгортки. Існують обгортки для Python (Astropy), Fortran, Perl та інших мов, що дозволяє використовувати бібліотеку в різних середовищах, що забезпечує гнучкість для розробників. Вони можуть обирати мову програмування, яка найкраще відповідає їхнім потребам.

Бібліотека також підтримує роботу з хмарними обчисленнями та розподіленими системами. Це дозволяє використовувати CFITSIO для обробки даних на сучасних високопродуктивних системах, що значно підвищує ефективність досліджень. Можливість паралельного доступу до даних та підтримка багатопоточності забезпечують високу продуктивність при роботі з великими обсягами даних.

CFITSIO є незамінним інструментом для астрономів, які працюють з даними у форматі FITS. Завдяки добре документованому API та сумісності з

іншими мовами програмування, CFITSIO забезпечує надійний доступ до астрономічних даних та дозволяє розробникам швидко інтегрувати її у свої проекти.

2.6 Аналіз складових моделі спектральних даних

На даний час існує модель обробки спектральних даних, яка може бути описана за допомогою кортежу (вираз 2.1). Наведена модель має низку недоліків і головними із них є використання застарілого програмного забезпечення та бібліотек, котрі не містять в собі змогу обробляти дані розподілено.

$$MP_{SD} = \{D, S [IRAF, Astropy], SA\}, \quad (2.1)$$

де D – датасет спектральних даних;

S – програмне забезпечення, що обробляє датасет даних;

SA – послідовний алгоритм обробки датасету.

Щоб уникнути недоліків, які було зазначено раніше, запропонуємо нову модель (вираз 2.2). Наведена модель відрізняється від моделі (2.1) тим, що в ній присутні паралельні методи обробки та середовище CUDA, які дозволяють значно покращити обробку спектральних даних астрономічних об'єктів за рахунок використання паралельних обчислень [16].

$$MP_{SD} = \{D, S [CUDA, OpenCV], DA\}, \quad (2.2)$$

де DA – розподілені алгоритми обробки датасету.

У порівнянні з послідовним методом, де дані обробляються послідовно один за одним, паралельний метод дозволяє реалізувати одночасну обробку, знижуючи загальний час обчислень та підвищуючи продуктивність. Таким чином, нова модель забезпечує більш ефективне використання обчислювальних ресурсів, відкриваючи шлях до більш глибокого та швидкого аналізу астрономічних даних.

Алгоритм обробки складається з наступних частин (рисунок 2.1):

- підключення бібліотеки OpenCV: в проєкт реалізації необхідно додати OpenCV бібліотеку;
- отримання даних у форматі FITS: на даному етапі модель отримує дані для обробки;
- проходження циклом по всім даним у форматі FITS: використовуючи паралельні та розподілені обчислення здійснюється обробка спектральних даних об'єкта (бібліотека OpenCV містить методи для роботи з форматом FITS);
- виведення результатів обробки у візуальному та текстовому форматах: частину результатів доцільно наводити у вигляді графіків (частотний спектр) іншу слід подати у текстовому форматі (характеристика хвиль);
- збереження результатів: запис даних у будь-який текстовий документ чи створення власного формату збереження для відтворення у програмі.



Рисунок 2.1 – Алгоритм роботи моделі, яка орієнтована на паралельні обчислення

Алгоритм моделі використовує наступні методи бібліотеки OpenCV:

- методи перетворення Фур'є (FFT), які дозволяють аналізувати спектральні компоненти даних. Перетворення Фур'є трансформує сигнал з часового домену в частотний, дозволяючи виявляти періодичні структури в даних та аналізувати спектральні складові (використання FFT в CUDA може значно прискорити обчислення за рахунок паралельної обробки даних);

- фільтрація Гауса (Gaussian Blurring) використовується для згладжування зображення шляхом застосування Гаусового ядра. Цей метод можна застосувати для попередньої обробки спектральних даних, щоб зменшити шум та підготувати дані для подальшого аналізу (реалізація Гаусового згладжування в CUDA може значно покращити швидкість обробки великих наборів даних);

- краєве виявлення (Edge Detection), це методи (наприклад, оператор Собеля), які можуть використовуватися для виявлення країв астрономічних об'єктів в спектральних даних. Краєве виявлення корисне для виділення меж об'єктів та може бути застосоване до спектральних зображень для подальшого аналізу структур (паралельна обробка даних за допомогою CUDA може забезпечити швидке виявлення країв в великих наборах даних);

- морфологічні операції включають дилатацію та ерозію, які можуть використовуватися для корекції форми астрономічних об'єктів у спектральних даних (ці методи можуть допомогти покращити виділення та аналіз структур, роблячи їх більш виразними для подальшого дослідження);

- процес нормалізації коригує інтенсивність пікселів спектральних зображень, щоб їх було легше аналізувати (нормалізація допомагає згрупувати дані з подібним діапазоном значень, що є критично важливим для порівняльного аналізу різних спектральних зображень);

- паралельна обробка даних – це розбиття даних на менші блоки для паралельної обробки, що дозволяє максимально використовувати обчислювальні ресурси GPU. Важливо розробити алгоритм таким чином, щоб він міг ефективно масштабуватися залежно від кількості доступних

потоків.

Під час реалізації алгоритмів обробки даних на CUDA важливо ефективно використовувати пам'ять GPU, щоб мінімізувати затримки доступу до даних. Використання різних рівнів кешування та розумне управління пам'яттю може значно підвищити продуктивність.

Припустимо, що маємо дві моделі, одна з яких робить послідовну обробку, інша – розподілену та має 10 ядер, доступних для обробки. При таких потужностях, якщо на обробку одного файлу уходить одна хвилина, то послідовна модель буде обробляти один файл за хвилину, коли паралельна – 10 файлів за хвилину.

3 РЕАЛІЗАЦІЯ АЛГОРИТМУ АНАЛІЗУ СПЕКТРАЛЬНИХ ДАНИХ

3.1 Доступ до спектральних даних

Архіви NASA [17] є одними з найбільш доступних та вичерпних джерел спектральних даних. Вони забезпечують відкритий доступ до даних з численних місій та проєктів, дозволяючи дослідникам з усього світу використовувати ці ресурси для своїх досліджень. Серед найбільш важливих архівів NASA можна виділити Архів космічних телескопів (MAST), Архів рентгенівських та гамма-даних (HEASARC) і Архів інфрачервоних даних (IRSA).

MAST (Mikulski Archive for Space Telescopes) є одним з основних архівів NASA, що містить дані з таких місій, як Hubble Space Telescope (HST), Kepler, TESS (Transiting Exoplanet Survey Satellite) та інших космічних телескопів. MAST надає доступ до спектральних даних у форматі FITS, які можна завантажити через інтерфейс архіву. Для цього користувачі можуть скористатися веб-порталом MAST, де доступні різні інструменти для пошуку та завантаження даних. MAST дозволяє здійснювати пошук за різними критеріями, такими як об'єкт спостереження, дата спостереження, тип даних тощо.

HEASARC (High Energy Astrophysics Science Archive Research Center) є ще одним важливим архівом NASA, що зосереджується на даних з рентгенівських та гамма-телескопів. Цей архів містить дані з місій, таких як Chandra X-ray Observatory, XMM-Newton, Swift, Fermi Gamma-ray Space Telescope та інших. HEASARC також надає доступ до спектральних даних у форматі FITS. Щоб отримати доступ до даних користувачі можуть використати веб-портал HEASARC, де доступні інструменти для пошуку та завантаження даних. Архів підтримує широкий спектр критеріїв пошуку, що дозволяє легко знаходити необхідні дані.

IRSA (Infrared Science Archive) є архівом NASA, що містить дані з

інфрачервоних місій, таких як Spitzer Space Telescope, WISE (Wide-field Infrared Survey Explorer), Herschel та інших. IRSA надає доступ до спектральних даних у форматі FITS, які можна завантажити через веб-портал архіву. Користувачі можуть здійснювати пошук за різними критеріями, включаючи об'єкт спостереження, дату спостереження, довжину хвилі тощо. IRSA також надає інструменти для візуалізації та аналізу даних, що робить роботу з архівом ще зручнішою.

Окрім архівів NASA, існують і інші важливі джерела спектральних даних у форматі FITS. Наприклад, Європейська космічна агенція (ESA) також надає доступ до даних зі своїх місій через архіви, такі як ESA Science Data Archive (ESAC) [18]. Користувачі можуть здійснювати пошук та завантаження спектральних даних у форматі FITS через веб-портал архіву.

На додаток до великих архівів, багато обсерваторій та наукових установ також надають доступ до своїх даних. Наприклад, Sloan Digital Sky Survey (SDSS) [19] є одним з найбільш значущих астрономічних проєктів, що надає доступ до великої кількості спектральних даних. Дані SDSS доступні у форматі FITS і можуть бути завантажені через веб-портал проєкту.

Щоб отримати спектральні дані у форматі FITS важливо також розуміти основні принципи роботи з ними. FITS є гнучким форматом, що дозволяє зберігати різні типи даних, включаючи одновимірні спектри, багатовимірні масиви, зображення та таблиці. Спектральні дані у форматі FITS зазвичай включають метадані, які описують структуру даних, параметри спостереження, калібрувальну інформацію тощо. Це дозволяє точно відтворити умови спостереження та забезпечити їх коректний аналіз.

Для реалізації алгоритмів було обрано спектральні дані формату FITS галактики Андромеда з архівів NASA, які забезпечують високу ступінь структурованості даних, що робить їх зручними для використання. Для первинного аналізу даних архіву надають інструменти для візуалізації, що дозволяють попередньо переглянути та обробити дані без необхідності завантаження на локальний комп'ютер.

3.2 Підготовка робочого середовища

3.2.1 Встановлення CUDA

Для розробки високопродуктивних додатків з використанням GPU від NVIDIA було використано середовище розробки Visual Studio 2022. Спершу необхідно завантажити та встановити останню версію драйверів для графічного процесора NVIDIA з офіційного сайту, що забезпечує коректну роботу обладнання та сумісність з CUDA.

Наступним кроком стало завантаження CUDA Toolkit з офіційного сайту NVIDIA [20]. Обрано версію для відповідної операційної системи та завантажено інсталяційний файл. Запуск інсталятора та вибір стандартної установки включає всі необхідні компоненти, такі як компілятор nvcc, бібліотеки CUDA та зразки програм.

Після завершення установки CUDA Toolkit здійснено налаштування середовища Visual Studio 2022. Далі створено новий проєкт, обравши шаблон "CUDA Runtime" з доступних опцій. Цей шаблон автоматично налаштовує проєкт для використання CUDA та включає приклад коду, що демонструє базову інтеграцію з нею.

Якщо відкрити проєкт, який було створено на попередньому кроці, та переглянути його структуру файлів, то можливо побачити, що основний файл проєкту містив простий приклад коду на CUDA, який демонструє основні принципи паралельних обчислень на графічному процесорі. Далі в цій код можливо ввести деякі зміни для подальшої перевірки його роботи та виконати компіляцію проєкту, використовуючи вбудований компілятор nvcc. Результатом компіляції буде запуск програми. Виконання програми показало, що обчислення на GPU відбувалися коректно, і результати відповідали очікуваням. Це підтвердило, що установка CUDA та інтеграція з Visual Studio 2022 були виконані успішно.

Протягом всього процесу також перевірялася наявність необхідних змінних середовища, які додаються автоматично під час встановлення CUDA

Toolkit. Зокрема, змінна `CUDA_PATH` вказувала на каталог встановлення CUDA, що дозволяло компілятору `nvcc` та іншим інструментам правильно знаходити необхідні бібліотеки та заголовки.

3.2.2 Встановлення OpenCV

Інтеграція OpenCV у проєкт, який створено у Visual Studio 2022, проходила в кілька етапів та потребувала належної уваги до деталей, щоб забезпечити коректну роботу всіх компонентів.

Спочатку було завантажено останню версію OpenCV з офіційного сайту (завантаження включало попередньо скомпільовані бібліотеки для Windows, що значно спрощує процес інтеграції). Після завантаження архіву з бібліотекою його було розпаковано у зручне місце на комп'ютері, щоб мати легкий доступ до всіх необхідних файлів (це включало директорії з заголовковими файлами, бібліотеками та динамічними бібліотеками, які потрібні для роботи з OpenCV).

У налаштуваннях проєкту було вказано шляхи до заголовкових файлів OpenCV. Для цього було здійснено перехід у розділ "Properties" проєкту, потім у вкладку "VC++ Directories" та додано шлях до директорії `include`, яка знаходиться в розпакованій папці OpenCV, до списку "Include Directories". Це дозволило компілятору знаходити всі необхідні заголовкові файли під час побудови проєкту.

Наступним кроком є вказання шляхів до бібліотек OpenCV. У тій самій вкладці "VC++ Directories" додано шлях до директорії `x64/vc15/lib`, де знаходяться бібліотеки OpenCV, до списку "Library Directories". Це дозволило компілятору та лінкеру знаходити необхідні файли під час компіляції та лінкування проєкту. Без цього кроку компілятор не зміг би знайти бібліотеки, і проєкт не компілювався б успішно.

Потім було здійснено перехід до розділу "Linker" у налаштуваннях проєкту та додано необхідні бібліотеки OpenCV до списку "Additional Dependencies". У полі "Additional Dependencies" вказано назви бібліотек,

таких як `opencv_world455.lib`, які відповідають завантаженій версії OpenCV. Це забезпечило правильне зв'язування з необхідними функціями та методами бібліотеки під час компіляції проєкту.

Останнім етапом стало налаштування змінних середовища для правильного виконання програми. Додано шлях до директорії `x64/vc15/bin`, де знаходяться динамічні бібліотеки OpenCV (.dll файли), до змінної середовища `PATH` в системі Windows. Це дозволило файлу, що виконується, знаходити необхідні бібліотеки під час запуску програми. Без цього кроку програма не змогла б знайти бібліотеки під час виконання, що призвело б до помилок завантаження.

3.2.3 Встановлення бібліотеки CFITSIO

Для розробки проєкту з використанням бібліотеки CFITSIO було вирішено скористатися менеджером пакетів `vsrkg` [21], який значно спрощує процес установки та інтеграції бібліотек у проєкти Visual Studio 2022. Використання `vsrkg` дозволяє автоматично завантажувати, компілювати та налаштовувати необхідні бібліотеки, що робить процес інтеграції швидким і зручним.

Було завантажено та встановлено `vsrkg`, слідуючи інструкціям на офіційному сайті проєкту. Після завантаження архів було розпаковано у зручну директорію на комп'ютері. Далі, використовуючи командний рядок, було здійснено перехід до цієї директорії та виконано команду для побудови `vsrkg`. Ця команда запустила процес компіляції `vsrkg`, що зайняло кілька хвилин. Після успішної побудови менеджера пакетів його було додано до системного шляху, щоб мати можливість викликати `vsrkg` з будь-якого місця в командному рядку.

Після налаштування `vsrkg` було здійснено установку бібліотеки CFITSIO. Менеджер пакетів `vsrkg` дозволяє легко встановлювати необхідні бібліотеки, викликаючи відповідні команди у командному рядку. Було використано команду для завантаження та компіляції CFITSIO, що

автоматично здійснило всі необхідні кроки, включаючи завантаження вихідних кодів бібліотеки з офіційного репозиторію, їх компіляцію та інтеграцію з системою.

Після завершення процесу установки бібліотеки CFITSIO через `vsrpg` було здійснено перехід до налаштування середовища розробки Visual Studio 2022 для роботи з цією бібліотекою. Відкривши Visual Studio, створено новий проєкт на мові C++. Обрано шаблон "Консольний додаток", що створило базову структуру проєкту, яка надалі буде використовувати бібліотеку CFITSIO.

Далі було здійснено перехід до налаштувань проєкту у Visual Studio, щоб вказати необхідні шляхи до файлів заголовків та бібліотек CFITSIO. В налаштуваннях проєкту було додано шляхи до директорій, де `vsrpg` зберігає встановлені бібліотеки. Це включало додавання шляхів до файлів заголовків у розділі "Include Directories" та шляхів до бібліотек у розділі "Library Directories". Також, в розділі "Linker" було додано необхідні бібліотеки CFITSIO до списку додаткових залежностей, щоб забезпечити коректне зв'язування проєкту з цією бібліотекою.

Використання `vsrpg` для встановлення бібліотеки CFITSIO значно спростило процес інтеграції. Менеджер пакетів автоматизував багато рутинних задач, таких як завантаження вихідних кодів, компіляція та налаштування шляхів до файлів. Це дозволило зосередитися на розробці функціональності проєкту, а не на технічних деталях налаштування середовища. Крім того, використання `vsrpg` забезпечило сумісність всіх компонентів та мінімізувало можливі помилки, пов'язані з некоректною конфігурацією.

3.3 Реалізація алгоритму спектральних даних

Критерієм для аналізу було обрано визначення максимальної інтенсивності спектра та порівняння ефективності обчислень на різних апаратних платформах. Для досягнення цієї мети були розроблені алгоритми

обробки даних на CPU та GPU, а також створено засоби візуалізації результатів, що дозволяє наочно представити отримані дані та зробити відповідні висновки щодо ефективності використання тих чи інших апаратних засобів.

3.3.1 Метод послідовного аналізу даних на CPU

Для аналізу спектральних даних та визначення максимальної інтенсивності за допомогою центрального процесора (CPU) була розроблена функція `FindMaxIntensityByProcessor` (лістинг 3.1). Ця функція приймає масив спектральних даних та визначає максимальне значення інтенсивності.

Функція виконує простий ітеративний алгоритм, що проходить через всі елементи масиву спектру. Початкове значення максимальної інтенсивності встановлюється як 0. На кожному кроці ітерації функція порівнює поточне значення інтенсивності з поточним максимальним значенням i , якщо поточне значення більше, оновлює максимальну інтенсивність. Такий підхід дозволяє швидко та ефективно знайти максимальне значення інтенсивності у спектральних даних, використовуючи лише обчислювальні ресурси центрального процесора.

Лістинг 3.1 – Реалізація методу послідовного аналізу даних на CPU

```
void FindMaxIntensityByProcessor(const float* spectrum, float& maxIntensity,
int spectrumSize)
{
    maxIntensity = 0.0f;
    for (int i = 0; i < spectrumSize; ++i)
        if (spectrum[i] > maxIntensity)
            maxIntensity = spectrum[i];
}
```

3.3.2 Метод паралельного аналізу даних на GPU

Функція `FindMaxIntensityByGraphicsCard` (лістинг 3.2) відповідає за аналіз спектральних даних та визначення максимальної інтенсивності за допомогою графічного процесора (GPU). Ця функція виконується паралельно потоками CUDA.

Метод використовує механізм спільної пам'яті для збереження локальних значень інтенсивності, обчислюваних кожним потоком. На початку кожен потік зчитує значення з масиву спектру та зберігає його у спільній пам'яті. Якщо індекс потоку виходить за межі масиву, встановлюється мінімальне значення.

Після синхронізації потоків значення інтенсивності зливаються за допомогою циклу з бітовим зсувом. На кожній ітерації порівнюються пари значень та зберігається більше значення.

Коли залишається тільки одне значення на кожному блоці потоків, воно оновлює глобальну максимальну інтенсивність за допомогою атомарної операції. Цей підхід дозволяє ефективно розподілити обчислення та зменшити час обробки великих обсягів спектральних даних, використовуючи потужності GPU.

Лістинг 3.2 – Реалізація методу паралельного аналізу даних на GPU

```
__global__ void FindMaxIntensityByGraphicsCard(float* spectrum, float*
maxIntensity, int spectrumSize)
{
    extern __shared__ float shared_data[];
    int tid = threadIdx.x;
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < spectrumSize)
        shared_data[tid] = spectrum[idx];
    else
        shared_data[tid] = -FLT_MAX;
    __syncthreads();
    for (int stride = blockDim.x / 2; stride > 0; stride >>= 1)
    {
        if (tid < stride)
            shared_data[tid] = fmaxf(shared_data[tid], shared_data[tid +
stride]);
        __syncthreads();
    }
    if (tid == 0)
        atomicMax((int*)maxIntensity, __float_as_int(shared_data[0]));
}
```

3.3.3 Методи візуалізації результатів

Для візуалізації спектральних даних та результатів обчислень були розроблені функції DisplaySpectrumData (лістинг Б.1) та DisplayResults (лістинг Б.2). Вони використовують бібліотеку OpenCV для створення

графіків та виводу текстової інформації на екран. Ці функції забезпечують наочне представлення даних, що полегшує їх аналіз та інтерпретацію.

Функція `DisplaySpectrumData` приймає масив спектральних даних і його розмір як вхідні параметри. Вона створює зображення розміром 800x600 пікселів з полями по 50 пікселів з кожного боку. Спочатку визначається максимальна інтенсивність у спектральних даних, що необхідно для масштабування значень по вертикальній осі. Потім функція малює лінії, що з'єднують сусідні точки спектру, щоб створити графік. Додатково, на графіку малюються осі з підписами та позначками, які показують значення довжини хвилі та інтенсивності. Це дозволяє легко читати дані з графіку та робити висновки про спектральний розподіл.

Функція `DisplayResults` використовується для відображення текстових результатів, таких як значення інтенсивностей, знайдених максимальних значень або інших обчислень. Вона приймає рядок результатів як вхідний параметр та виводить його на зображення, яке потім відображається на екрані. Результати можуть включати, наприклад, порівняння обчислень, виконаних на CPU та GPU. Це дозволяє користувачам швидко оцінити ефективність обчислень та зробити відповідні висновки.

Обидва методи використовують основні можливості OpenCV, такі як малювання ліній, тексту та інших графічних примітивів. Вони також використовують різні параметри для налаштування вигляду графіків, такі як колір, товщина ліній та розмір шрифту. Це забезпечує гнучкість у візуалізації та дозволяє адаптувати графіки до конкретних вимог дослідження.

Застосування бібліотеки OpenCV для візуалізації даних має кілька переваг. По-перше, це потужний інструмент з великою кількістю функцій для обробки зображень, що дозволяє легко створювати складні візуалізації. По-друге, OpenCV підтримує різні формати зображень і має хорошу інтеграцію з іншими бібліотеками, що робить його універсальним рішенням для наукових та інженерних застосувань.

Використання таких візуалізаційних функцій у рамках аналізу

спектральних даних дозволяє отримати більш глибоке розуміння процесів, що досліджуються. Візуалізація допомагає виявляти закономірності, аномалії та інші важливі аспекти даних, які можуть бути неочевидними при простому перегляді числових значень. Це особливо важливо у випадках, коли дані містять велику кількість точок або мають складну структуру.

3.3.4 Головний метод програми

Основний метод програми відповідає за виконання всіх етапів обробки спектральних даних, включаючи читання даних із файлу, обчислення максимальної інтенсивності за допомогою CPU та GPU, а також візуалізацію результатів. У лістингу Б.3 наведено детальне пояснення коду основного методу, де основними кроками є:

- відкриття FITS-файлу та читання спектральних даних;
- обробка даних на CPU;
- обробка даних на GPU;
- візуалізація результатів.

Першим етапом є відкриття FITS-файлу, що містить спектральні дані. Для цього використовується функція `fits_open_file`, яка приймає шлях до файлу та режим доступу (в даному випадку – тільки для читання). Якщо файл успішно відкрито, наступним кроком є отримання параметрів зображення, таких як кількість осей та розміри даних, за допомогою функції `fits_get_img_param`. Ці параметри використовуються для визначення загального розміру даних, які потрібно прочитати.

Після визначення розміру даних, виділяється пам'ять для зберігання спектральних даних на хості (CPU). У разі успішного виділення пам'яті, виконується читання даних з FITS-файлу за допомогою функції `fits_read_img`. Якщо використовується збільшений розмір даних, масив даних доповнюється шляхом циклічного повторення існуючих значень до досягнення необхідного розміру.

Для обробки даних на CPU використовується функція `FindMaxIntensityByProcessor`, яка знаходить максимальну інтенсивність у спектральних даних. Процесорний час обчислень вимірюється за допомогою стандартних засобів для роботи з часовими інтервалами в C++. Після завершення обчислень результати, включаючи максимальну інтенсивність, час виконання та продуктивність (кількість операцій в секунду), зберігаються для подальшого відображення.

Для обробки даних на графічному процесорі (GPU) використовуються функції CUDA. Спочатку виділяється пам'ять на GPU для зберігання спектральних даних та змінної, що зберігатиме максимальну інтенсивність. Дані копіюються з хоста на пристрій за допомогою `cudaMemcpy`. Потім викликається ядро CUDA, яке виконує паралельне обчислення максимальної інтенсивності. Після завершення обчислень результати копіюються назад на хост. Час виконання GPU-обчислень також вимірюється, а результати зберігаються для подальшого відображення.

Після завершення обчислень на CPU та GPU результати візуалізуються за допомогою функцій `DisplaySpectrumData` та `DisplayResults`. Функція `DisplaySpectrumData` створює графік спектральних даних, який показує розподіл інтенсивності по довжині хвилі. Це дозволяє візуально оцінити спектральні характеристики.

3.4 Результати роботи алгоритму

Після успішної компіляції проєкту, програма відкриває два вікна з результатами (рисунки 3.1, 3.2). Для порівняння ефективності методів було використано такі характеристики, як час виконання методу та кількість операцій в секунду.

Згідно результатів, можна зробити висновок, що CPU при обробці одного FITS-файлу працює набагато швидше та ефективніше, ніж GPU. Для подальшої оцінки використаємо більший об'єм даних та повторимо аналіз спектральних даних.

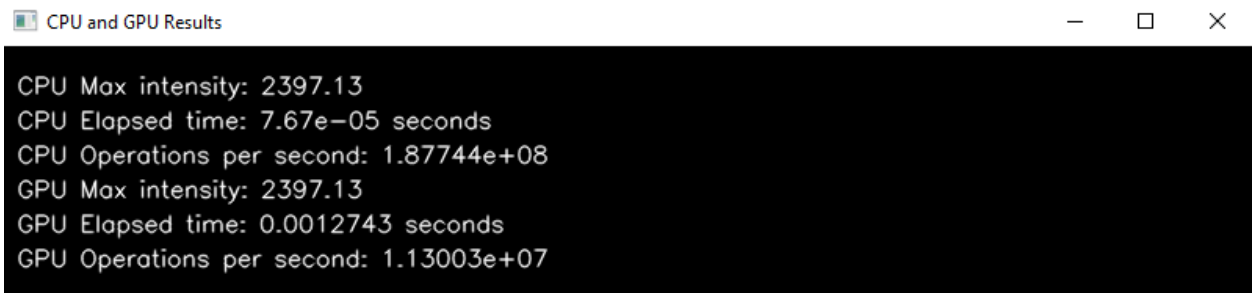


Рисунок 3.1 – Результати виконання методів на CPU та GPU

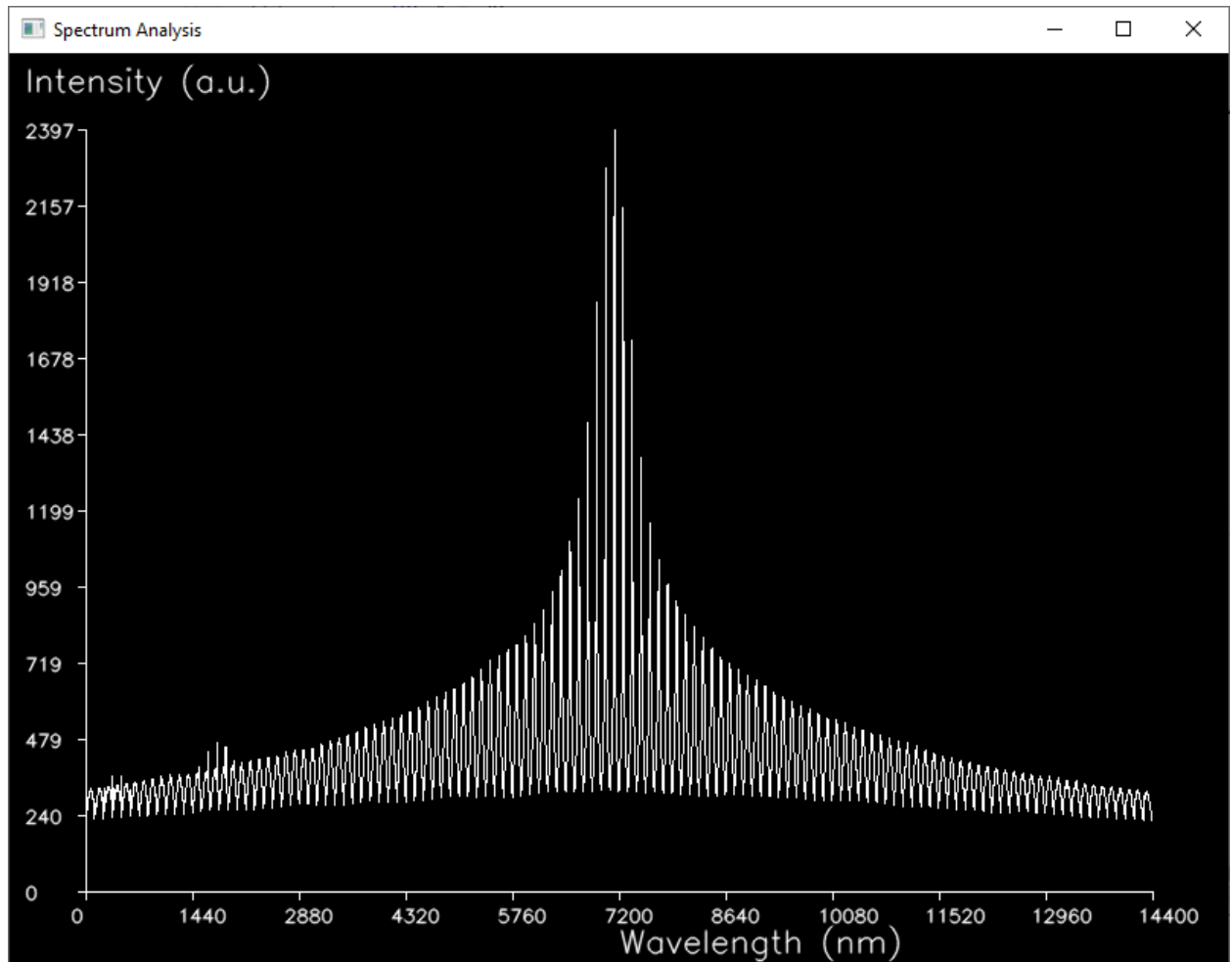


Рисунок 3.2 – Функція залежності інтенсивності сигналу від довжини хвилі

Для збільшення кількості даних використаємо коефіцієнт `SpectrumSizeMultiplier`, який вказує, у скільки разів необхідно збільшити об'єм для аналізу. Дані повторних аналізів наведено у таблиці 3.1, за результатами яких можна зробити висновок, що при великих об'ємах даних GPU працює набагато швидке та ефективніше, ніж CPU. Це особливо помітно при обробці великих масивів спектральних даних, де GPU демонструє значну перевагу.

Таблиця 3.1 – Результати роботи алгоритму в залежності від коефіцієнту збільшення об'єму даних

SpectrumSizeMultiplier	Час виконання на CPU в секундах	Час виконання на GPU в секундах	Кількість операцій в секунду на CPU	Кількість операцій в секунду на GPU
1	0.0000767	0.0012743	187744000	11303000
1024	0.0203643	0.0126303	724091000	1167480000
2046	0.0306026	0.0271126	962742000	1086670000
4092	0.0784251	0.0475633	751351000	1238870000
8096	0.126231	0.091471	923562000	1274530000

ВИСНОВКИ

В результаті проведених досліджень було розглянуто актуальні методи та інструменти для аналізу спектральних даних астрономічних об'єктів, приділено увагу використанню середовища IRAF та бібліотеки Astropy, які залишаються важливими інструментами в для ефективного аналізу даних. Також було наголошено про актуальність паралельних та розподілених обчислень на платформі CUDA, що відіграють критичну роль в оптимізації обробки великих обсягів даних, отриманих з радіотелескопів.

На основі аналізу існуючих підходів та інструментів було обґрунтовано необхідність використання інструментів, які реалізують можливості паралельних обчислень. Запропонована модель обробки спектральних даних відкриває нові горизонти для підвищення ефективності обробки спектральних даних, які отримано з радіотелескопів та збережено у форматі FITS. Використання паралельних обчислень дозволяє значно зменшити час, необхідний для обробки великих даних, що є ключовим для прискорення наукових досліджень.

Рекомендована модель, що ґрунтується на паралельних обчисленнях, пропонує ефективний спосіб вирішення цієї задачі. Вона не тільки сприяє підвищенню швидкості обробки даних, але й відкриває нові можливості для глибшого розуміння структури Всесвіту та динаміки небесних явищ.

Під час проведення експериментів було виявлено, що при обробці малих обсягів даних, час виконання на CPU менший, ніж на GPU. Це пояснюється тим, що накладні витрати на ініціалізацію та передачу даних на GPU стають значущими у порівнянні з загальним часом виконання завдання. Тобто, коли обсяги даних незначні, час на передачу даних та запуск ядра CUDA на GPU перевищує виграш від паралельної обробки.

З іншого боку, при збільшенні обсягу даних ситуація змінюється. Для великих обсягів даних GPU починає демонструвати значну перевагу у швидкості обробки. Це пов'язано з високою паралельною обчислювальною

потужністю GPU, яка дозволяє одночасно обробляти великі масиви даних. У таких випадках накладні витрати на передачу даних і запуск ядра стають незначущими у порівнянні з часом, який витрачається на обробку даних.

Результати експериментів підтверджують необхідність використання паралельних обчислень на GPU для ефективної обробки великих обсягів спектральних даних в астрономії. Використання GPU дозволяє значно скоротити час обробки даних, що є критично важливим для прискорення наукових досліджень та відкриття нових астрономічних явищ.

Майбутні дослідження в цій галузі можуть включати подальше вдосконалення алгоритмів паралельної обробки, розширення функціоналу інструментів, що використовуються, а також адаптацію моделі для нових областей застосування в астрономії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Подорожняк А.О., Гриб Р.М., Домнін С.В. (2013), "Морфологічна обробка цифрових зображень з телескопів", Сучасна спеціальна техніка, №1(32), С. 34-39.
2. Dmytrenko A.M., Akhmetov V.S. (2017), "Astronomical image processing for high-accurate astrometry data", Odessa Astronomical Publications, Volume 30, p. 163-165.
3. Savanevych V.E., Khlamov S.V., Akhmetov V.S., Briukhovetskyi A.B., Vlasenko V.P., Dikov E.N. et. al. (2022), "CoLiTecVS software for the automated reduction of photometric observations in CCD-frames", Astronomy and Computing, 40, 100605. doi: <https://doi.org/10.1016/j.ascom.2022.100605>
4. Вертегел С., Вишняков В., Гуреля В., Сластін С., Піскун О., Харченко С., Мороз В. (2022), "Розробка методики створення і оновлення картографічної основи з використанням космічних знімків від супутників "SUPER VIEW-1", Екологічна безпека та природокористування, 41(1), С. 89-101. <https://doi.org/10.32347/2411-4049.2022.1.89-101>
5. Бандурка О.І., Свинчук О.В. (2022), "Метод ідентифікації космічних знімків для прогнозування лісових пожеж", Системи управління, навігації та зв'язку. Збірник наукових праць, Т. 1 (67). С. 13-18. doi:<https://doi.org/10.26906/SUNZ.2022.1.013>.
6. Манойлов В.П., Омельчук В.В., Опанюк В.В. (2008), "Дистанційне зондування Землі із космосу", Науковотехнічні основи формування й обробки видової інформації: Монографія, 384 с.
7. Анисенко О.В. (2017), "Розвиток дистанційного зондування землі в Україні", Агросвіт, № 7, С. 52-59.
8. Сайт IRAF. URL: <https://iraf-community.github.io>.
9. Сайт Astropy. URL: <https://www.astropy.org>.
10. Сайт SAS. URL:https://www.sas.com/en_us/insights/big-data/what-is-big-data.html.

11. Сайт CUDA. URL: <https://docs.nvidia.com/cuda/doc/index.html>.
12. Сисуєв В. "Використання розподілених обчислювальних потужностей для аналізу даних астрономічних спостережень", Астрономія і сьогодення: матеріали IX Всеукраїнської наукової конференції, №13, 2020. С. 71-74.
13. Сайт OpenCV. URL: <https://opencv.org>.
14. Сайт AMD. URL: <https://www.amd.com/en/products/software>.
15. Сайт CFITSIO. URL: <https://heasarc.gsfc.nasa.gov/fitsio/fitsio.html>.
16. Ніколаєнко Д.С., Філімончук Т.В., Майстренко Г.В. Модель обробки спектральних даних астро-номічних об'єктів // Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: ПНТУ, 2024. Випуск 2 (76). С. 135-137. doi: 10.26906/SUNZ.2024.2.135.
17. Сайт NASA. URL: <https://www.nasa.gov/archives>.
18. Сайт IRAM. URL: <https://iram-institute.org/science-portal/data-archive>.
19. Сайт SDSS. URL: <https://www.sdss.org>.
20. Доступ до CUDA Toolkit. URL: <https://developer.nvidia.com/cuda-toolkit>.
21. Доступ до утиліти vcpkg. URL: <https://vcpkg.io/en>.