

Национальный исследовательский университет
«Высшая школа экономики»

Работа с тензорами с помощью GPGPU
Отчет по ИП ФКН

Студент: К.И. Сулейманов,
Руководитель: Р.А. Родригес Залепинос

Москва, 2025

Содержание

0 Публикация	3
1 Введение	8
2 Данные	8
3 Методы	12
3.1 Растворная алгебра	12
3.2 Wavelet	14
4 Технологии	15
4.1 CUDA C	15
4.2 QGIS на Python и PyCUDA	17
5 Текущие проблемы	18
6 Постановка задачи	19
6.1 Модель данных	19
6.2 Обучение работе с GPU	21
6.2.1 Чтение тензоров из формата GeoTIFF в GPU	21
6.2.2 Растворная алгебра на GPU	21
6.3 Сравнение скорости GPU и CPU	22
6.4 Ускорение вычислений с помощью вейвлетов	23
6.5 Реализация reshaping	23
7 Текущие результаты	23
8 Перспективы на будущее	24

0. Публикация

0.1 Опыт участия в конференции

В сентябре 2025 года принял участие в международной конференции «Суперкомпьютерные дни в России». Опубликована работа в трудах конференции. Представил устный доклад, который сопровождался постером (стендовым докладом), отвечал на вопросы участников, получил содержательную обратную связь от исследовательского сообщества. Посетил основные доклады и тематические сессии, пообщался с коллегами, что помогло уточнить дальнейшие шаги в проекте.

Выступление и общение на конференции показали высокую заинтересованность аудитории в ускорении растровой алгебры. Полученный опыт публичного представления работы и обсуждения результатов оказался крайне полезным и вдохновляющим, помог расширить профессиональные контакты и подтвердил актуальность выбранного направления. Этот обмен идеями влияет на дальнейшее планирование экспериментов и приоритетов реализации.

Отдельно получил книгу «Методы Монте-Карло для параллельных вычислений» (Зорин А.В., Федоткин М.А.) [42]. В ней изложены методы статистического моделирования для параллельных архитектур: генерация независимых потоков псевдослучайных чисел и случайных векторов, приближённое вычисление интегралов высокой размерности, численное решение дифференциальных уравнений и имитационное моделирование. Изданье ориентировано на студентов и исследователей, работающих с численным моделированием и параллельным программированием; использую её как справочник при подготовке экспериментов.

0.2 Публикация

Сулейманов К. И., Родригес Залепинос Р. А. **Обработка растровых данных на GPU в геоинформационных системах // Суперкомпьютерные дни в России: Труды международной конференции. 29–30 сентября 2025, Москва / Под ред. Вл. В. Воеводина.** Москва: МАКС Пресс, 2025. С. 244--246. DOI: 10.29003/m4750.978-5-317-07451-7.

0.3 Материалы стендового доклада

Ниже приведены публикация: аннотация и полный постер формата A1. Материалы доклада прошли два этапа рецензирования экспертами конференции, по результатам которых была оценена на 89 баллов, принята

на конференцию и получили комментарий "Хорошая работа, посвященная обработке изображений.".

В ходе подготовки публикации я познакомился с требованиями к обнародованию научной работы, оформлением и процессом рецензии работы. Подготовился к ответу на вопросы слушателей доклада. Подготовил стендовый доклад: оформил постер в соответствии с требованиями мероприятия, отработал краткую и расширенную подачу результатов, а также получил и зафиксировал обратную связь участников для дальнейшей работы.

Обработка растровых данных на GPU в геоинформационных системах

К.И. Сулейманов, Р.А. Родригес Залепинос

Национальный исследовательский университет «Высшая школа экономики»

Введение. Растворная модель данных – это широко распространенный метод хранения географических данных. Чаще всего эта модель представляет собой структуру, напоминающую сетку, в которой хранятся значения с регулярными интервалами по всей площади раstra. Раstry особенно хорошо подходят для хранения непрерывных данных, таких как температура и высота над уровнем моря, но также могут содержать дискретные и категориальные данные, например, данные о землепользовании. Разрешение раstra задается в линейных единицах (например, метрах) или угловых единицах (например, одной угловой секунде) и определяет протяженность вдоль одной стороны ячейки сетки. Раstry высокого (или низкого) разрешения имеют сравнительно меньшее расстояние между ячейками сетки и больше ячеек, чем раstry низкого (или высокого) разрешения, и требуют относительно большего объема памяти для хранения. Активные исследования в этой области направлены на улучшение схем сжатия и реализацию альтернативных форм ячеек (например, шестиугольников), а также на улучшение поддержки функций хранения и анализа раstrов с несколькими разрешениями [1].

Например, данные Sentinel — это спутниковые данные, предоставляемые программой Copernicus Европейского космического агентства [2]. Они включают многоспектральные снимки, радарные данные и данные о высотах, пригодные для мониторинга земной поверхности, сельского хозяйства, водных ресурсов и экологии [3].

Проблема обработки растровых данных заключается в их огромных объемах. При увеличении разрешения растрового изображения (т.е. уменьшении размера пикселя) объем хранимых данных резко растёт – например, при двухкратном уменьшении размера ячейки объем данных может увеличиться вчетверо, и скорость их обработки сильно снижается. С каждым годом появляются новые спутниковые миссии и БПЛА, генерирующие всё больше данных высокого разрешения, поэтому проблема быстрой обработки больших раstrов становится всё более актуальной и работа с растровыми данными выходит на первый план [3].

Для ускорения обработки растровых данных применяются графические процессоры (GPU) – специальные видеокарты, способные выполнять вычисления параллельно. GPU изначально предназначены для ускорения работы с графикой, но в последнее время их вычислительные мощности активно используется для общих вычислительных задач. В инструментах геопространственного анализа задача обработки раstrов выполняется GPU вместо центрального процессора: задача дробится на множество мелких подзадач, которые GPU выполняет параллельно с высокой скоростью, затем результаты собираются воедино [4, 5].

Цель работы – разработка универсальных GPU-ускоренных подходов для пакетной обработки геопривязанных растровых данных в ГИС, способные автоматически распараллеливать вычисления на видеокартах NVIDIA под CUDA и обеспечивать стабильное ускорение на любых объемах. Ожидается достичь 5–10 кратного сокращения времени анализа спутниковых данных и аэрофотоснимков (Sentinel-2, другие раstry) и гибкая интеграция с популярными ГИС-платформами с целью масштабирования обработки растровых данных.

В качестве основы предлагается использовать библиотеку GDAL на языке C++ [9]. Например, реализация алгебры карт (Map Algebra), используя библиотеку для чтения и записи данных, выполняя математические операции над пикселями раstrов, обрезки (clipping) с использованием GDALWarpOperation с параметрами -cutline и -crop_to_cutline для выделения области интереса из раstra по заданному векторному контуру или прямоугольной области и ресемплирования (resampling) с использованием функции GDALWarpOperation с настройкой параметров интерполяции и масштабирования для изменения пространственного разрешения раstra с использованием различных методов интерполяции (например, ближайшего соседа, билинейной, кубической).

Существующие GPU-библиотеки и утилиты для обработки растров: cuCIM от NVIDIA – это библиотека для GPU-ускоренной загрузки и обработки больших файлов TIFF [7]. OpenCV имеет модуль CUDA с GPU-версиями фильтров и преобразований [8], библиотека CuPy в Python предоставляет NumPy – подобные массивы для вычислений на GPU. Библиотека Rasterio (на основе GDAL) непосредственно CUDA не использует, но с помощью Dask/CuPy можно распараллеливать обработку больших растров. Несмотря на наличие всех этих инструментов, они рассчитаны на отдельные задачи и не приспособлены для массивно-параллельной пакетной обработки геопривязанных растров в ГИС-сценариях. Аналогов полностью предполагающих решение для массивно-параллельной обработки данных ГИС нет.

Исследование [4] реализовало параллельную обработку растров в ГИС посредством CUDA C, обеспечив 7-кратное ускорение по сравнению с ArcGIS. Время выполнения CUDA линейно зависело от размера данных (подтверждено тестами до 8×64 ГБ). Например, обработка растра размером 1.96 ГБ заняла 2000 секунд, сохраняя линейное ускорение. Основной узкий момент — загрузка данных в RAM, но оптимизация GPU сократила общее время. Результаты подтверждают эффективность GPU для масштабируемых вычислений.

В работе [5] представлен открытый плагин QGIS на Python/PyCUDA для ускорения анализа рельефа посредством GPU. PyCUDA показал 3-кратное ускорение по сравнению с QGIS (C++): обработка 1.5 ГБ заняла 3:35 минут вместо 11:00 минут, 12 ГБ – 28 минут вместо 45. Основное преимущество – в вычислениях (GPU обрабатывал 1.5 ГБ за 2 сек), но ограничения ввода-вывода снижали эффект для больших данных.

Исследование [6] показывает применение данных Sentinel: авторы оценили долю городской растительности в дельте реки Чжуцзян (Китай) с помощью данных Sentinel-2 (NDVI, разрешение 10 м) и валидации по высокодетальным изображениям Google. Результаты показали корреляцию 0.97 между оценками Sentinel-2 и эталонными данными, а также выявили значительные расхождения с методом WUDAPT level-0 (100 м): в плотных урбанизированных зонах WUDAPT завышал городскую фракцию, а в промзонах — занижал. Подход обеспечивает детализацию, критичную для климатического моделирования и анализа городской среды [6].

Тензорные СУБД являются популярным инструментом для обработки растровых данных, однако в них до сих пор не используются GPU для ускорения различный операций [10–13].

Литература

1. Rodriges Zalipynis R.A., Array DBMS: Past, Present, and (Near) Future. Proc. VLDB Endow. 2021, 14, 3186–3189
2. Sentinel Data. Copernicus satellite missions. URL: <https://sentivista.copernicus.eu>
3. The ArcGIS Imagery Book: New View. New Vision. Redlands, CA: ESRI Press, 2016.
4. Kirby S. P., Kostan W. B., Lembo A. J., Parallelizing Raster-Based Functions in GIS with CUDA C. Salisbury University. 2013. URL: <https://faculty.salisbury.edu/~ajlemb0/cudaposter.pdf>
5. Fuerst A., et al, GIS based terrain analysis with GPU and CPU strategies. Salisbury University. 2016. URL: https://faculty.salisbury.edu/~ealu/reu/Projects_File/2016/Lembo.pdf
6. Wong, M. M. F., Fung, J. C. H., & Yeung, P. P. S., High-resolution calculation of the urban vegetation fraction in the Pearl River Delta from the Sentinel-2 NDVI for urban climate model parameterization. *Geoscience Letters*, 6, 1-10, 2019.
7. NVIDIA RAPIDS cuCIM. URL: <https://github.com/rapidsai/cucim>
8. OpenCV (модуль CUDA). URL: <https://opencv.org/platforms/cuda>
9. GDAL – Geospatial Data Abstraction Library. URL: <https://gdal.org>
10. Rodriges Zalipynis R.A., BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. Proc. VLDB Endow. 2020, 13, 2909–2912.
11. Rodriges Zalipynis R.A., ChronosDB: Distributed, File Based, Geospatial Array DBMS. Proc. VLDB Endow. 2018, 11, 1247–1261.
12. Rodriges Zalipynis R.A., ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 1985–1988.
13. Baumann P., Misev D., Merticariu V., Huu B.P., Array databases: Concepts, standards, implementations. J. Big Data 2021, 8, 1–61.

Обработка растровых данных на GPU в геоинформационных системах

К.И. Сулейманов, Р.А. Родригес Залепинос

Национальный исследовательский университет «Высшая школа экономики»

Введение

Растровая модель данных – это широко распространенный метод хранения географических данных. Чаще всего эта модель представляет собой структуру, напоминающую сетку, в которой хранятся значения с регулярными интервалами по всей площади растра.

Активные исследования в этой области направлены на улучшение схем сжатия и реализацию альтернативных форм ячеек (например, шестиугольников), а также на улучшение поддержки функций хранения и анализа растров с несколькими разрешениями.

Проблема

Проблема обработки растровых данных заключается в их огромных объемах.

Для ускорения обработки растровых данных применяются графические процессоры (GPU) – специальные видеокарты, способные выполнять вычисления параллельно.

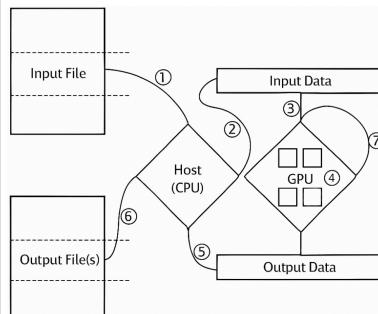
Цель работы

Цель работы – разработка универсальных GPU-ускоренных подходов для пакетной обработки геопривязанных растровых данных в ГИС, способные автоматически распараллеливать вычисления на видеокартах NVIDIA под CUDA и обеспечивать стабильное ускорение на любых объемах. Ожидается достичь 5–10 кратного сокращения времени анализа спутниковых данных и аэрофотоснимков (Sentinel-2, другие растры) и гибкая интеграция с популярными ГИС-платформами с целью масштабирования обработки растровых данных.

Существующие GPU-библиотеки и утилиты

- **cuCIM (NVIDIA)** — GPU-ускоренная загрузка и обработка больших TIFF-файлов [7].
- **OpenCV (CUDA-модуль)** — GPU-версии фильтров и преобразований [8].
- **CuPy (Python)** — NumPy-подобные массивы для GPU.
- **Rasterio (GDAL)** — напрямую CUDA не использует, но в связке с Dask/CuPy позволяет параллелизовать обработку больших растров.

Все они решают узкие задачи, но нет единого инструмента для массово-параллельной пакетной обработки георастров в ГИС.



Изображение 1. Модель программы обработки данных на GPU

1. Разделить данные по строкам и загрузить их в массив в памяти.
2. Скопировать массив данных на GPU.
3. Запустить GPU-ядро для обработки растра.
4. Выполнить параллельную обработку данных на GPU с помощью комбинации блоков и потоков.
5. По завершении каждого ядра скопировать данные с GPU обратно на CPU.
6. Записать результаты в соответствующий файл (по одному файлу на каждую растровую функцию).
7. Запустить дополнительные ядра для данных, которые уже находятся на GPU.

CUDA C: реализация параллельной обработки растров, **ускорение в 7** по сравнению с ArcGIS.

- Обработка **1.96 ГБ** заняла **2000 сек**, ускорение линейное при росте данных до 8×64 ГБ.
- Узкое место — загрузка в RAM.

QGIS + PyCUDA (Python плагин): ускорение анализа рельефа, **в 3 быстрее** стандартного QGIS (C++).

- **1.5 ГБ:** 3:35 мин (GPU) vs 11:00 мин (CPU).
- **12 ГБ:** 28 мин (GPU) vs 45 мин (CPU).
- GPU-обработка **1.5 ГБ за 2 сек**, но ввод-вывод остается ограничением.

Function	CUDA	QGIS
Input	1:55	2:00
Computation	1:00	7:00
Output	2:20	2:00
Total Time	3:35	11:00

Таблица 1. Сравнение скорости CUDA и QGIS

1. Введение

Современные вычислительные задачи — от обучения больших языковых моделей до постобработки спутниковых снимков всё чаще опираются на операции с многомерными тензорами [4, 41]. Согласно исследованиям, рост размерности и объёма данных приводит к экспоненциальному увеличению вычислительной сложности и объёма передаваемой информации [17], что делает использование GPGPU-ускорителей фактически обязательным для достижения необходимой производительности и энергоэффективности [20, 7].

Бывший научный сотрудник Стэнфордской лаборатории компьютерной графики и нынешний главный инженер по мобильным и облачным вычислениям в компании Nvidia говорит о том, что неспециализированные вычисления на графических процессорах идеально подходят для обработки данных и показывают многократное преимущество перед вычислениями на центральных процессорах, за счет возможности распараллеливания вычислений на большом объеме памяти. GPU выигрывает, когда все элементы обрабатываются одним и тем же ядром (kernel) без зависимостей [16].

Программирование на GPGPU и использование их для массивно-параллельных вычислений становится чрезвычайно популярным. Во многих самых производительных суперкомпьютерах мира установлены GPGPU: <https://top500.org/>. Графические ускорители также установлены в суперкомпьютере "сHARISMa" НИУ ВШЭ [9].

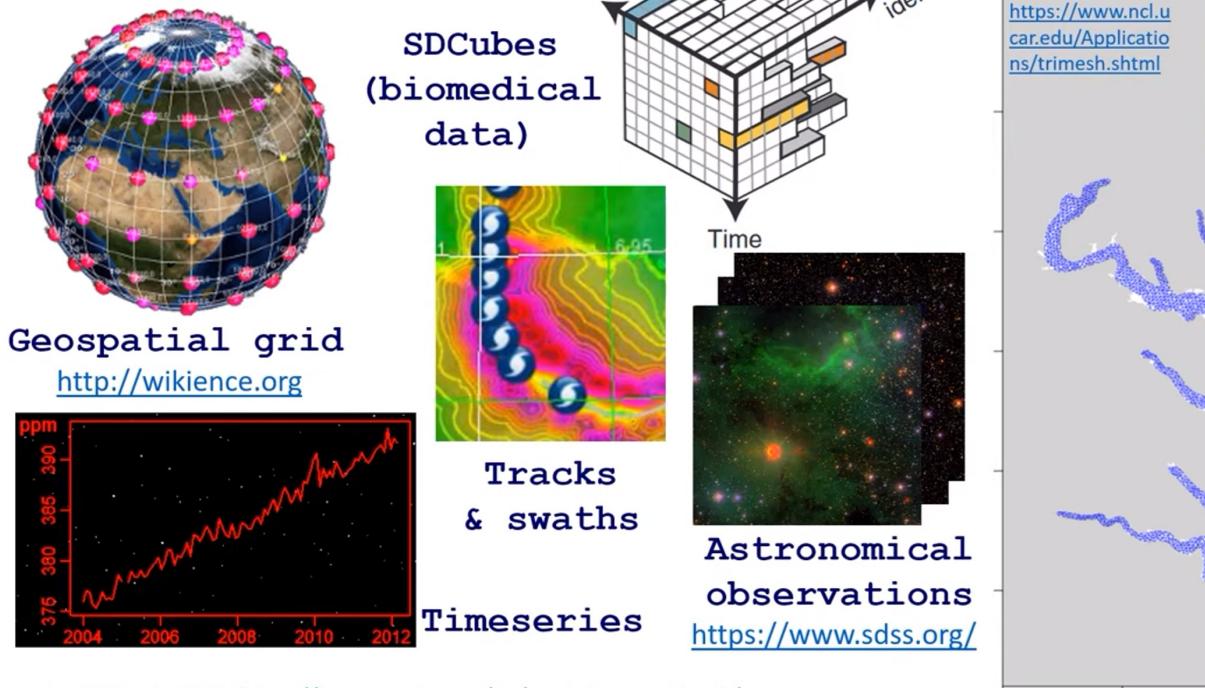
Тензор - является мощной абстракцией, многие типы данных являются тензорами либо тензор является естественным представлением для этих данных, рис. 1. Под тензором в работе понимается многомерный массив [26].

Эти факты свидетельствуют о чрезвычайной важности научно-исследовательской темы "Работа с тензорами с помощью GPGPU".

2. Данные

Растровая модель данных – это широко распространенный метод хранения географических данных. Чаще всего эта модель представляет собой структуру, напоминающую сетку, в которой хранятся значения с регулярными интервалами по всей площади раstra. Раstry особенно хорошо подходят для хранения непрерывных данных, таких как температура и высота над уровнем моря, но также могут содержать дискретные и категориальные данные, например, данные о землепользовании.

Datasets



- ArcGIS book. 2021. <https://learn.arcgis.com/en/arcgis-imagery-book/>
- M. Bjorn et al. "Adaptive informatics for multifactorial and high-content **biological** data." *Nature methods* 8.6 (2011)
- <https://www.hdfgroup.org/portfolio-item/astronomy/>

Рис. 1: Типы данных, которые естественным образом представляются в виде тензоров [40, 39]

Разрешение растра задается в линейных единицах (например, метрах) или угловых единицах (например, одной угловой секунде) и определяет протяженность вдоль одной стороны ячейки сетки. Раstry высокого (или низкого) разрешения имеют сравнительно меньшее расстояние между ячейками сетки и больше ячеек, чем раstry низкого (или высокого) разрешения, и требуют относительно большего объема памяти для хранения.

Активные исследования в этой области направлены на улучшение схем сжатия и реализацию альтернативных форм ячеек (например, шестиугольников), а также на улучшение поддержки функций хранения и анализа растром с несколькими разрешениями [39].

Например, Sentinel — это спутниковые данные, предоставляемые программой Copernicus Европейского космического агентства [30]. Они включают многоспектральные снимки, радарные данные и данные о высотах, пригодные для мониторинга земной поверхности, сельского хозяйства, водных ресурсов и экологии [2].

Исследование [36] показывает применение данных Sentinel: авторы оценили долю городской растительности в дельте реки Чжуцзян (Китай) с помощью данных Sentinel-2 (NDVI, разрешение 10 м) и валидации по высокодетальным изображениям Google. Результаты показали корреляцию 0.97

между оценками Sentinel-2 и эталонными данными, а также выявили значительные расхождения с методом WUDAPT level-0 (100 м): в плотных урбанизированных зонах WUDAPT завышал городскую фракцию, а в промзонах — занижал. Подход обеспечивает детализацию, критичную для климатического моделирования и анализа городской среды.

Другими популярными данными являются данные программы Landsat 2. Программа Landsat - самая длительная программа по наблюдению за Землей из космоса [21].

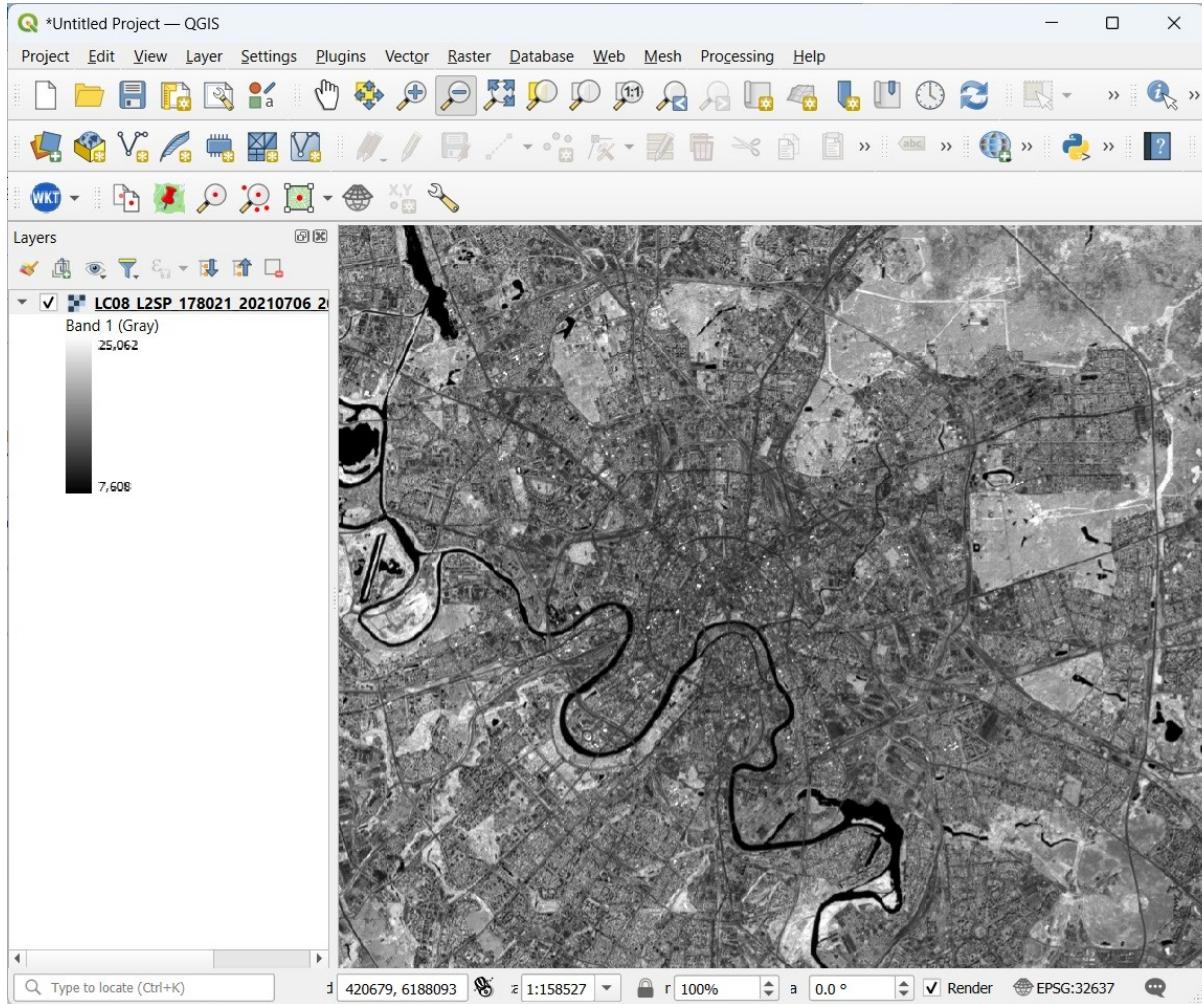


Рис. 2: Экранный снимок QGIS: Москва, ближний инфракрасный канал спутниковой сцены Landsat

Отдельно стоит архив NOAA Physical Sciences Laboratory (PSL) — исследовательской лаборатории NOAA, фокус которой связан с улучшением прогнозирования водной доступности и экстремумов (гидрологические/климатические риски) [25]. PSL ведёт публичный архив “Data and Imagery”, публикую исходные наборы и производные продукты/визуализации, сгруппированные по типам. В каталоге ncer.reanalysis2/pressure/ размещены годовые файлы NetCDF по схеме «переменная × год» (например, `air.1979.nc`),

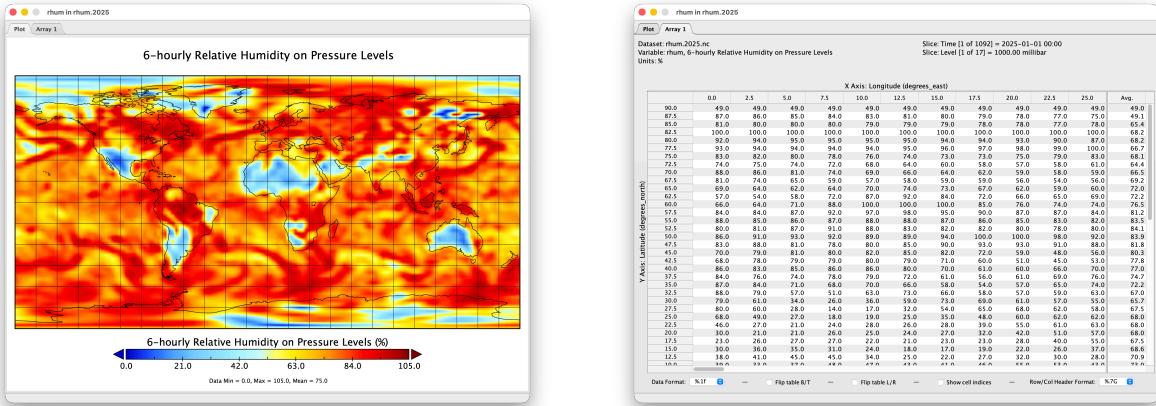


Рис. 3: Примеры просмотра NetCDF-данных в Panoply (слои давления и ветра)

`air.2025.nc` за 1979–2025 годы с указанием даты последнего изменения и размера [24]. Доступны группы переменных `air`, `hgt`, `omega`, `rhum`, `uwnd`, `vwnd`; размеры заметно различаются: `air.*` порядка 300 МБ в год, `rhum.*` 165–172 МБ, `omega.*` 480–605 МБ, `uwnd.*` 520–542 МБ.

Операция решейпинга для этих данных нетривиальна, потому что размеры файлов заметно отличаются как между годами, так и между переменными (например, файлы за 2025 год меньше, чем за 2024, а omega/uwnd значительно крупнее air/rhum). Это означает, что количество значений и/или число временных шагов и итоговые размеры многомерных массивов могут различаться, поэтому форму массива нельзя безопасно задавать вручную; её нужно извлекать из метаданных NetCDF (dimensions/variables) и преобразовывать данные, сохраняя смысл осей (а не просто совпадение произведения размеров).

Для просмотра и быстрой проверки таких NetCDF-полей удобно использовать приложение Panoply (NASA GISS) [23]: оно позволяет открывать многомерные массивы, выбирать срезы и визуализировать поля, что ускоряет первичную проверку данных перед загрузкой в вычислительный конвейер. На рис. 3 приведены примеры просмотра слоёв.

Проблема обработки растровых данных заключается в их огромных объёмах. При увеличении разрешения растрового изображения (т.е. уменьшении размера пикселя) объём хранимых данных резко растёт – например, при двукратном уменьшении размера ячейки объём данных может увеличиться вчетверо, и скорость их обработки сильно снижается.

С каждым годом появляются новые спутниковые миссии и БПЛА, генерирующие всё больше данных высокого разрешения, поэтому проблема быстрой обработки больших растров становится всё более актуальной и работа с растровыми данными выходит на первый план [2].

Для ускорения обработки растровых данных применяются графические процессоры (GPU) – специальные видеокарты, способные выполнять вычисления параллельно. GPU изначально предназначены для ускорения работы с графикой, но в последнее время их вычислительные мощности активно используется для общих вычислительных задач.

В инструментах геопространственного анализа задача обработки растров выполняется GPU вместо центрального процессора: задача дробится на множество мелких подзадач, которые GPU выполняет параллельно с высокой скоростью, затем результаты собираются воедино [18, 11].

В дальнейшем мы концентрируемся именно на спутниковых данных, поскольку это одни из самых объемных тензорных данных в открытом доступе с большим перечнем практических задач, в которых они активно применяются.

3. Методы

3.1. Растровая алгебра

Растровая алгебра представляет собой набор методов и операций, применяемых для выполнения пространственных и математических расчётов с растровыми данными. Основная идея заключается в том, что каждое вычисление применяется одновременно ко всем ячейкам раstra или группе растром, образуя новые растровые слои, содержащие результат этих операций. Наиболее распространённые операции включают арифметические, логические, статистические и тригонометрические вычисления, а также специализированные операции, такие как расчёт индексов вегетации (например, NDVI), уклона и экспозиции.

Формальные основы растровой алгебры были заложены в работах Даны Томлина, предложившего концепцию операций над растровыми слоями (алгебра карт) [33]. Данные подходы позволяют представить сложные пространственные задачи в виде комбинаций простых элементарных операций [34]. В последующие годы идеи Томлина были расширены и дополнены, например, в работе Камары и соавторов [5], предложивших включить в растровую алгебру пространственные предикаты и отношения, а также в трудах Джереми Менниса, расширившего применение растровой алгебры на многомерные и временные данные [22].

В рамках настоящей работы основное внимание уделяется применению растровой алгебры для анализа спутниковых изображений. Одной из ключевых операций, которые планируется реализовать и ускорить с помощью GPU, является вычисление нормализованного разностного индекса вегетации (NDVI). NDVI рассчитывается по формуле:

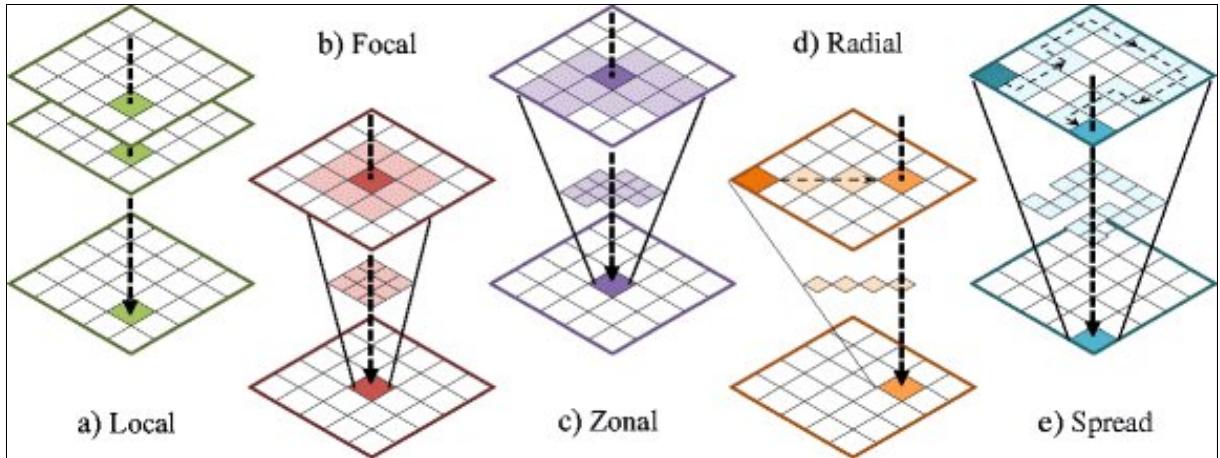


Рис. 4: Типы операций в растровой алгебре [6]

$$NDVI = \frac{NIR - RED}{NIR + RED + 1}, \quad (1)$$

где NIR — значение пикселя в ближнем инфракрасном канале, а RED — в красном канале изображения. Этот индекс широко используется для мониторинга состояния растительности и сельскохозяйственных угодий [15].

Другим важным примером применения растровой алгебры является операция ресемплинга, предполагающая изменение пространственного разрешения изображения. Ресемплинг становится критически важным при интеграции данных с разных спутников, имеющих различные пространственные разрешения [15, 38].

Для реализации операций растровой алгебры на GPU планируется использовать технологию CUDA, которая позволит распределить обработку растровых данных по тысячам параллельных потоков. Согласно классификации из работы [6], операции растровой алгебры хорошо подходят для GPU-ускорения благодаря их массово-параллельной природе и отсутствию зависимостей между вычислениями отдельных пикселей (см. рис. 4).

Существуют примеры успешного GPU-ускорения растровых операций, такие как реализация пространственных функций в GRASS GIS, показавшие ускорение расчётов на порядок [31], а также ускорение интерполяции и анализа видимости на GPU, продемонстрировавшее значительный прирост производительности по сравнению с традиционными CPU-реализациями [37].

Таким образом, успешная реализация растровой алгебры на GPU позволит значительно сократить время обработки спутниковых данных, повысив эффективность анализа и обеспечив возможность работы с крупномасштабными геопространственными проектами.

3.2. Wavelet

Современные задачи обработки изображений требуют эффективных вычислений, обладающих как высокой скоростью, так и масштабируемостью. Одним из перспективных направлений является использование вейвлет-преобразования, позволяющего работать с изображениями в частотно-пространственном представлении.

Данный подход обладает рядом преимуществ: он обеспечивает компактное (разреженное) представление данных, позволяет выполнять операции с прогрессивным уточнением (от грубого к детальному уровню), а также локализует действия как в пространственной, так и в частотной области. Ввиду широкой применимости и совместимости с современными стандартами (например, JPEG-2000), вейвлет-домен представляет интерес как в академических исследованиях, так и в практических ГИС-задачах.

В работе Iddo Drori и Dani Lischinski [10] предложен единый подход к выполнению ряда операций — таких как свёртка, 3D-вейвлет-войпинг (wavelet warping), и смешивание изображений — непосредственно в вейвлет-домене. Основной идеей является представление изображения и сопутствующих операторов в виде линейных комбинаций матриц, над которыми затем выполняется вейвлет-преобразование. Итоговая операция осуществляется над малым числом ненулевых коэффициентов, а восстановление результата производится посредством обратного преобразования.

Среди ключевых преимуществ подхода — возможность выполнения операций на различных разрешениях (multi-resolution), поддержка прогрессивной реконструкции и значительное ускорение расчётов за счёт разреженности представления. Например, при применении к задаче 3D-вейвлет-войпинга авторы показали, что в сферических проекциях алгоритм вейвлет-войпинга работает до 1.8 раза быстрее, чем классические методы. При этом сохраняется возможность интерактивной визуализации изображений в режиме реального времени благодаря поэтапной реконструкции.

Кроме того, вейвлет-преобразование позволяет реализовать как lossless, так и lossy-сжатие изображений. Как показано в учебных материалах по сжатию на базе преобразования Хаара [35], детализация изображения (так называемые *detail coefficients*) зачастую близка к нулю. Это делает возможным удаление малых коэффициентов без существенной потери качества, что, в свою очередь, позволяет достичь высокой степени сжатия.

Для реализации вейвлет-преобразований были рассмотрены несколько базисов, включая S-преобразование (целочисленная версия Хаара) и I(2,2)-базис (интерполяционное биортогональное преобразование). Первый обеспечивает максимальную скорость, второй — наилучшее качество визуального результата при потере части информации. Выбор подходящего вейвлет-

базиса определяется компромиссом между разреженностью, временем реконструкции и устойчивостью к визуальным искажениям.

Таким образом, подход к обработке изображений в вейвлет-домене представляет собой универсальное и эффективное решение, применимое как к операциям над отдельными изображениями, так и к последовательностям кадров, особенно в условиях ограниченных вычислительных ресурсов.

4. Технологии

Существующие GPU-библиотеки и утилиты для обработки растров: cuCIM от NVIDIA – это библиотека для GPU-ускоренной загрузки и обработки больших файлов TIFF [7]. OpenCV имеет модуль CUDA с GPU-версиями фильтров и преобразований [8], библиотека CuPy в Python предоставляет NumPy – подобные массивы для вычислений на GPU.

Библиотека Rasterio (на основе GDAL) непосредственно CUDA не использует, но с помощью Dask/CuPy можно распараллеливать обработку больших растров. Несмотря на наличие всех этих инструментов, они рассчитаны на отдельные задачи и не приспособлены для массивно-параллельной пакетной обработки геопривязанных растров в ГИС-сценариях. Аналогов полностью предполагающих решение для массивно-параллельной обработки данных ГИС нет.

4.1. CUDA C

CUDA C — это расширение языка программирования C/C++, разработанное компанией NVIDIA для реализации параллельных вычислений на графических процессорах (GPU). Основное преимущество данной технологии заключается в возможности эффективного распределения вычислений между тысячами потоков, что делает её особенно привлекательной для задач, требующих обработки больших объёмов данных [8].

В контексте геоинформационных систем (ГИС), где распространены ресурсоёмкие операции с растровыми изображениями, использование CUDA C позволяет существенно повысить производительность по сравнению с традиционными подходами на центральном процессоре (CPU). Ввиду актуальности ускорения пространственных расчётов в рамках анализа изображений и моделирования, данная технология была выбрана для подробного рассмотрения.

В работе [19] была реализована параллельная обработка растровых данных в геоинформационных системах (ГИС) с использованием техно-

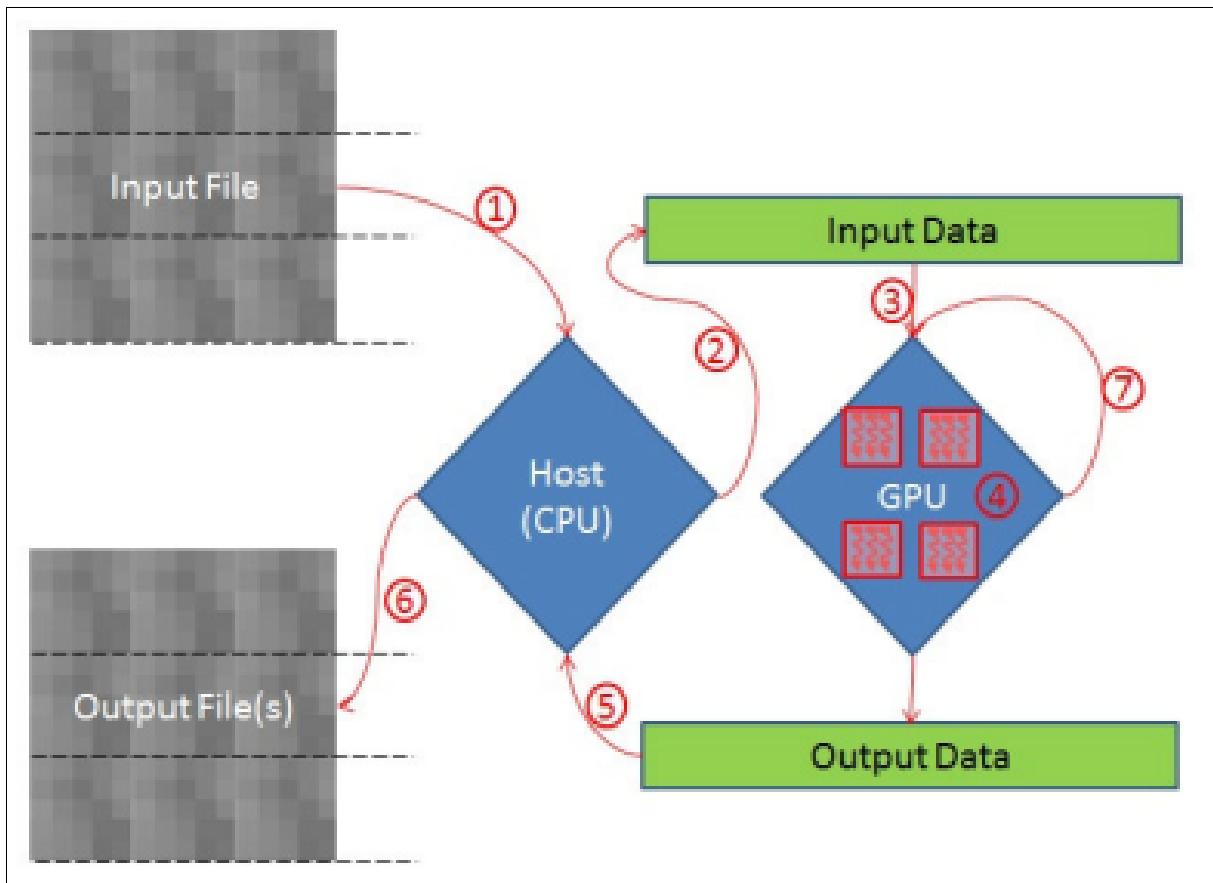


Рис. 5: Модель программы из исследования [19]

логии CUDA C. Полученные результаты демонстрируют, что применение графических процессоров (GPU) позволяет достичь существенного ускорения — в среднем в 7 раз быстрее по сравнению с традиционной реализацией на платформе ArcGIS.

Проведённые тесты показали линейную зависимость времени выполнения CUDA-функций от объёма входных данных. В частности, обработка растрового файла объёмом 1,96 ГБ заняла порядка 2000 секунд, в то время как обработка 93 ГБ (25 миллиардов пикселей) потребовала менее 3000 секунд при предварительно спрогнозированном времени в 3100 секунд, что подтверждает стабильность и масштабируемость подхода.

Наибольшее ускорение наблюдалось при выполнении нескольких пространственных функций на одном и том же наборе данных — прирост производительности по сравнению с ArcGIS варьировался от 3 до 22 раз. Среди исследованных функций — уклон, экспозиция, среднее значение, минимум и максимум, каждая из которых является «параллелизуемой» и хорошо масштабируется на GPU. Обработка данных осуществлялась поэтапно: чтение и разбиение на блоки, копирование в память GPU, выполнение вычислений в потоках и блоках, возвращение результатов в память CPU и последующая запись в выходной файл.

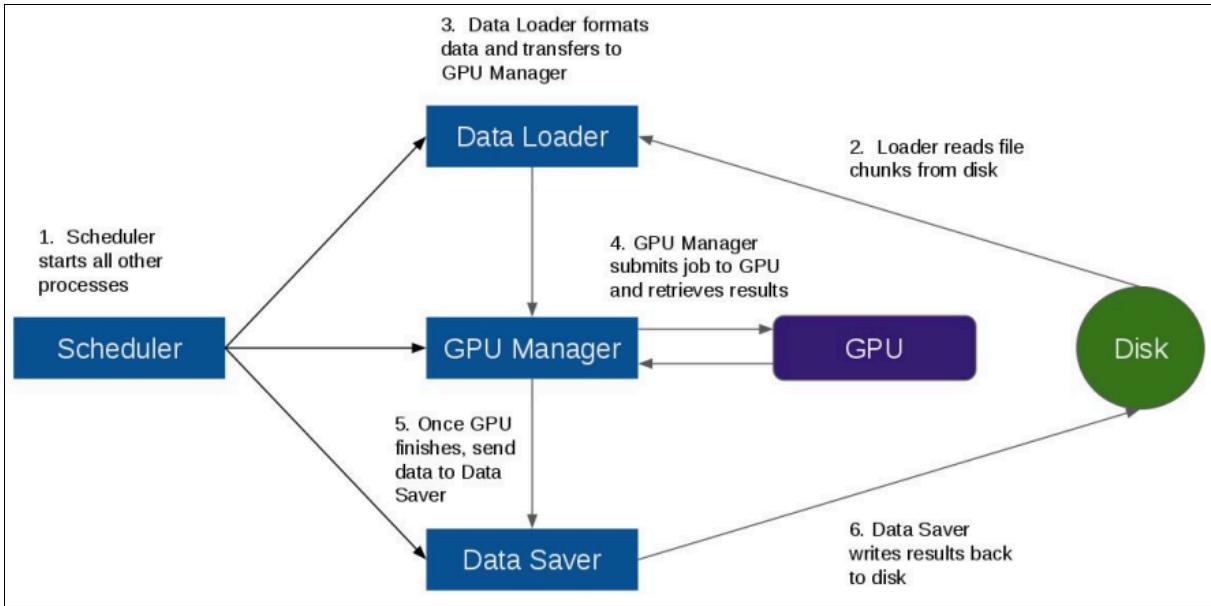


Рис. 6: Модель программы из исследования [12]

Было выявлено, что основным узким местом остаётся фаза чтения данных с диска и загрузки их в оперативную память. Однако авторы отмечают, что даже без полной оптимизации этого этапа, внедрение CUDA обеспечивает значительное общее сокращение времени выполнения. Рекомендуемое направление дальнейших исследований — параллельная загрузка данных в оперативную память посредством многопоточности на CPU во время выполнения вычислений на GPU. Этот подход может обеспечить дополнительный прирост производительности. Кроме того, авторы предлагают применять аналогичные GPU-ориентированные методы в смежных областях, таких как дистанционное зондирование, экологический мониторинг и моделирование природных ресурсов, где также широко применяются повторяющиеся пространственные расчёты.

4.2. QGIS на Python и PyCUDA

Одним из актуальных направлений в области ускорения ГИС-вычислений является использование открытых решений, интегрированных в популярные ГИС-платформы. В рамках данного обзора было решено рассмотреть реализацию анализа рельефа в среде QGIS с использованием языка Python и библиотеки PyCUDA. Выбор обусловлен тем, что PyCUDA позволяет комбинировать удобство разработки на Python с высокой производительностью вычислений на GPU, а также предоставляет доступ к параллельным вычислениям пользователям, не обладающим глубокими знаниями в области низкоуровневого программирования.

В работе [12] представлен открытый плагин для QGIS, реализую-

щий ускорение анализа рельефа с помощью графического процессора. Плагин был написан на языке Python с использованием библиотеки PyCUDA и предназначен для выполнения вычислений уклона, экспозиции и затенения рельефа на растровых данных. Алгоритмы плагина рассчитаны на работу с окрестностью 3×3 вокруг каждого пикселя и позволяют легко модифицировать код под другие функции анализа.

Архитектура решения включает три независимых процесса: загрузку данных, их обработку на GPU и последующую запись на диск. Такая модульная структура обеспечивает параллелизм как на уровне CPU (ввод-вывод), так и на уровне GPU (вычисления), что приводит к сокращению общего времени выполнения.

Результаты экспериментов показали, что реализация на PyCUDA обеспечивает 3-кратное ускорение по сравнению со встроенной реализацией QGIS на C++. Например, для растрового файла объёмом 1,5 ГБ общее время выполнения составило 3 минуты 35 секунд против 11 минут в QGIS. При этом само время вычислений на GPU заняло менее 2 секунд, а остальные затраты связаны в основном с операциями ввода-вывода.

Для большего файла объёмом 12 ГБ ускорение оказалось менее выраженным (28 минут против 45 минут в QGIS), что обусловлено ограничениями по скорости чтения и записи данных с диска. Авторы отмечают, что использование SSD и чтение крупных блоков данных снижает влияние узкого места, но полностью не устраняет его.

В целом, работа демонстрирует потенциал применения GPU вычислений в составе открытых ГИС-платформ, подчёркивая значимость дальнейших исследований в области оптимизации ввода-вывода и расширения поддерживаемых алгоритмов анализа рельефа.

5. Текущие проблемы

Тензорные СУБД являются популярным инструментом для обработки растровых данных, однако в них до сих пор не используются GPU для ускорения различных операций [3, 27, 28, 29].

Существующие GPU-библиотеки (cuCIM, модуль CUDA в OpenCV, CuPy) ориентированы на отдельные операции и не образуют единой экосистемы для пакетной обработки геопривязанных растров; интеграция с GDAL/Rasterio или QGIS требует значительных усилий.

Выбор базиса (S-преобразование vs. I(2,2)) и порога отсечения коэффициентов влияет на качество реконструкции и время обратного преобразования; отсутствуют автоматизированные рекомендации для ГИС-практиков [10].

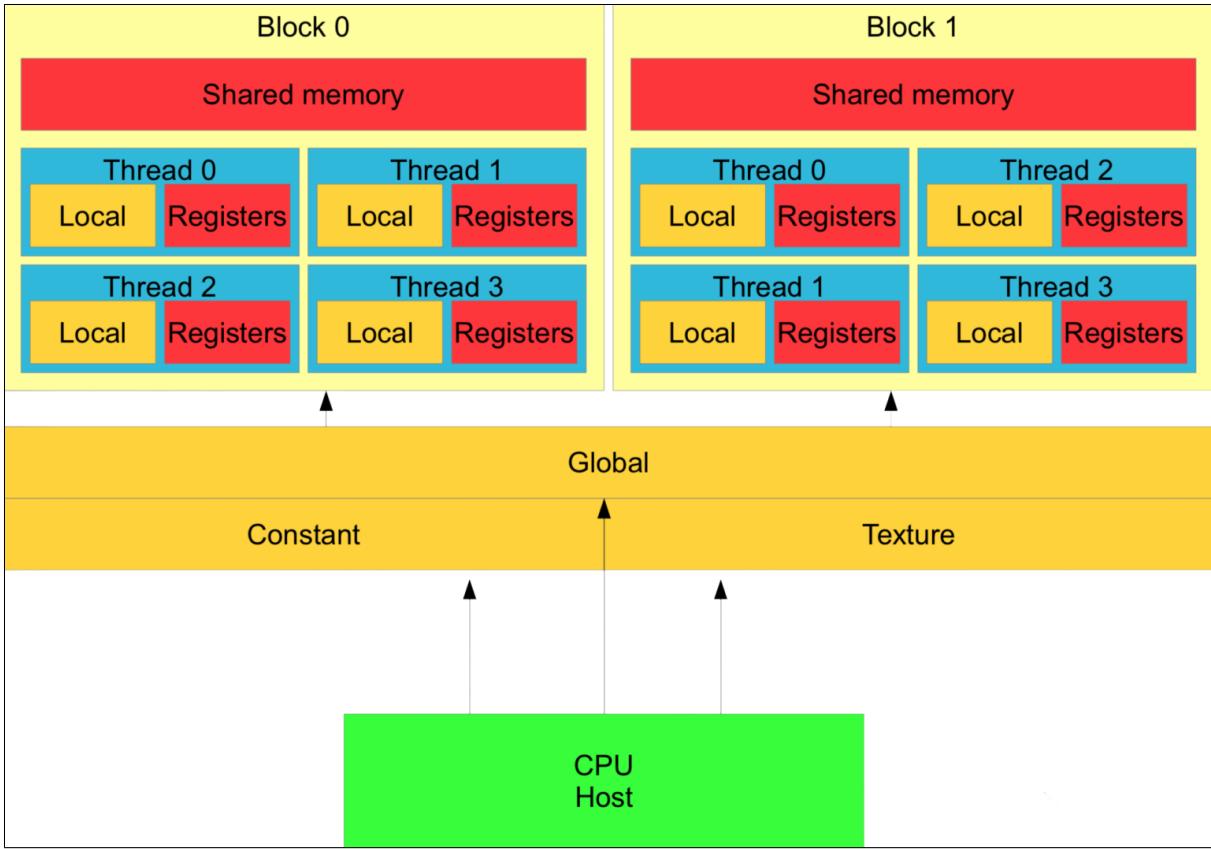


Рис. 7: Иерархия памяти в CUDA [32]

И как уже было отмечено ранее, одной из самых больших проблем является то, что в современных решениях процессы чтения, вычисления и записи редко организованы как полноценный конвейер, из-за чего GPU простояивает во время операций ввода-вывода. Необходимо грамотно реализовывать работу с памятью в CUDA с пониманием её иерархии (рис. 7).

6. Постановка задачи

Обзор литературы

В рамках проекта необходимо изучить литературу по теме и разобраться и качественно её проанализировать. Систематизация и интеллектуальная проработка статей, их структурирование и указание связи между ними и проектом в том числе.

6.1. Модель данных

Для формальной постановки задачи, нам необходимо определить формальную модель данных. Будем использовать часть модели данных

из [29].

Определение 6.1 (*N*-мерный массив). Пусть $N \in \mathbb{N}$, $N > 0$. *N*-мерным массивом (тензором) называется отображение

$$A : D_1 \times D_2 \times \cdots \times D_N \longrightarrow T,$$

где для каждого $i \in \{1, \dots, N\}$

$$D_i = \{0, 1, \dots, l_i - 1\} \subset \mathbb{Z}, \quad l_i \in \mathbb{N},$$

а T — числовой тип данных (например, `int`, `float`).

Величина l_i называется *длиной* (или *размером*) i -й оси, а кортеж

$$\text{shape}(A) := (l_1, l_2, \dots, l_N)$$

— *формой* (*shape*) массива (тензора) A .

Определение 6.2 (Мощность массива (тензора)). *Мощностью* (или *размером*) массива A называется общее число его элементов

$$|A| := \prod_{i=1}^N l_i.$$

Элемент с координатами $(x_1, \dots, x_N) \in D_1 \times \cdots \times D_N$ обозначается $A[x_1, x_2, \dots, x_N]$ и имеет тип T . Значение `NA` зарезервировано для обозначения *пропущенного* элемента.

Определение 6.3 (Переформатирование (reshaping) массива). Пусть $A : D_1 \times \cdots \times D_N \rightarrow T$ — массив формы $\text{shape}(A) = (l_1, \dots, l_N)$. Пусть заданы новые размеры m_1, \dots, m_M такие, что $\prod_{i=1}^N l_i = \prod_{j=1}^M m_j$. Линейный индекс элемента исходного массива

$$\text{lin}(x_1, \dots, x_N) = \sum_{k=1}^N x_k \left(\prod_{j=k+1}^N l_j \right)$$

переводится в координаты выходного массива B по правилу

$$y_j = \left\lfloor \frac{\text{lin}(x_1, \dots, x_N)}{\prod_{r=j+1}^M m_r} \right\rfloor \bmod m_j, \quad j = 1, \dots, M,$$

и значение присваивается как $B[y_1, \dots, y_M] = A[x_1, \dots, x_N]$. Операция обозначается $B = \text{reshape}_{(m_1, \dots, m_M)}(A)$ и сохраняет порядок элементов в линейном представлении [29].

6.2. Обучение работе с GPU

Любые научно-технические работы на GPU имеют высокий порог вхождения. Поэтому для начала необходимо освоить устройство и работу GPU, научиться выполнять ввод/вывод данных и программировать GPU.

6.2.1. Чтение тензоров из формата GeoTIFF в GPU

GeoTIFF сочетает привычный контейнер TIFF с полной геопривязкой, поэтому спутниковые сцены можно сразу открывать в инструментах GDAL/Rasterio. Типовой поток обработки включает чтение плитки или окна с помощью `RasterIO` (GDAL) либо cuCIM в закреплённую (pinned) память хоста и асинхронное копирование этого блока в видеопамять через `cudaMemcpyAsync`; (3) выполнение CUDA-kernel над загруженным массивом. Такой подход позволяет параллельно перекрывать ввод-вывод и вычисления [13].

6.2.2. Растворная алгебра на GPU

Ранее уже было сказано про растворную алгебру. В этом разделе я хочу формально определить операции NDVI и ресемплинга описанные выше. Другие операции так же рассматриваются в проекте, эти две операции выбраны лишь для демонстрации в отчете. Определения также возьмем отсюда [29].

Определение 6.4 (Нормализованный относительный индекс растительности, NDVI). Пусть $A^{\text{nir}}, A^{\text{red}} : D_1 \times D_2 \rightarrow \mathbb{R}_{\geq 0} \cup \{\text{NA}\}$ — взаимно гео-референцированные массивы, представляющие значения ближнего инфракрасного (NIR) и красного (Red) диапазонов спутникового снимка соответственно. Для каждой ячейки $(x, y) \in D_1 \times D_2$ $NDVI$ определяется как

$$NDVI(x, y) = \frac{A^{\text{nir}}[x, y] - A^{\text{red}}[x, y]}{A^{\text{nir}}[x, y] + A^{\text{red}}[x, y] + 1}$$

Значение $NDVI(x, y) \in [-1, 1]$ служит прокси-оценкой фотосинтетической активности растительного покрова.

Определение 6.5 (k -кратное ресемплирование усреднением). Пусть $A : D_1 \times D_2 \rightarrow T$ — двумерный массив, где $D_i = \{0, 1, \dots, l_i - 1\}$ и $k \in \mathbb{N}$, $k \geq 2$ — коэффициент декрета (обычно $k = 2$). Определим уплотнённый массив

$$R_k(A) : D'_1 \times D'_2 \rightarrow T, \quad D'_i = \left\{0, 1, \dots, \left\lfloor \frac{l_i}{k} \right\rfloor - 1\right\},$$

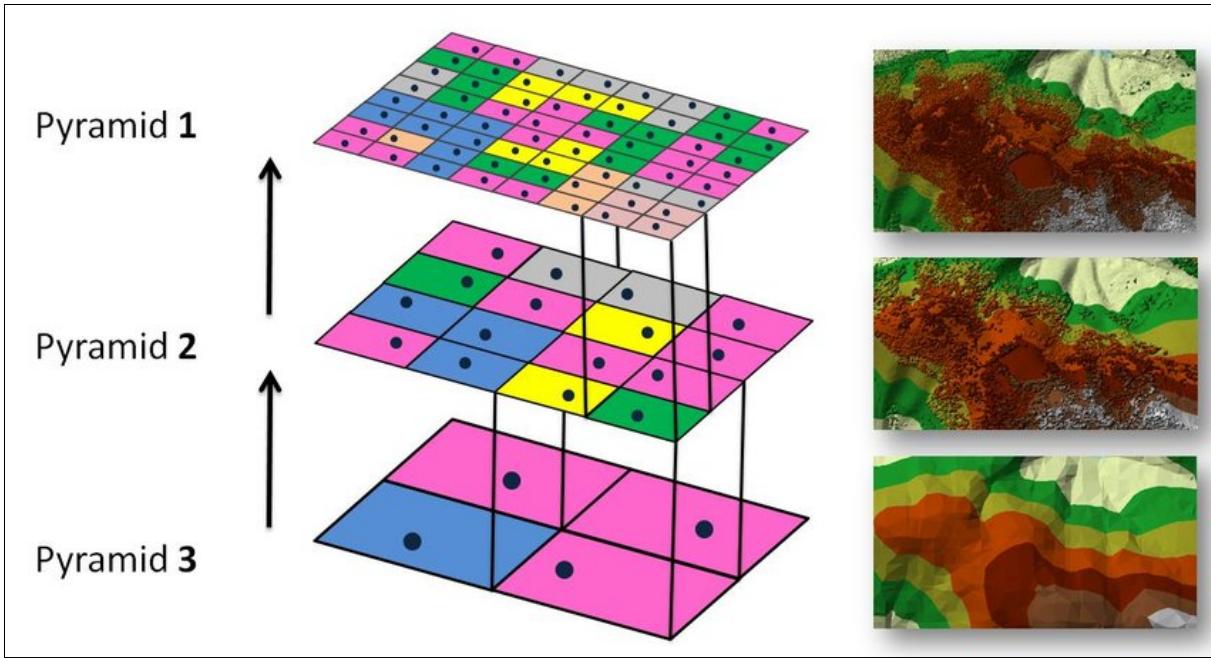


Рис. 8: Визуализация ресемплирования [1]

по правилу

$$R_k(A)[u, v] = \text{avg}\left(\{A[x, y] \mid uk \leq x < (u+1)k, vk \leq y < (v+1)k, A[x, y] \neq \text{NA}\}\right),$$

$$(u, v) \in D'_1 \times D'_2.$$

где $\text{avg}(\emptyset) = \text{NA}$. Таким образом, каждая ячейка выходного массива есть среднее ненулевых значений соответствующего блока $k \times k$ входного массива. При $k = 2$ $R_k(A)$ соответствует одному уровню *многоуровневой пирамиды разрешений* (рис. 8).

6.3. Сравнение скорости GPU и CPU

Важной частью является сравнение реализованной технологии с `gdal_calc` — это консольная утилита пакета GDAL, которая выполняет арифметические и логические операции над растровыми наборами данных, интерпретируя выражения в синтаксисе NumPy. Она позволяет автоматизировать комплексную обработку растров (например, вычисление индексов, маскирование, объединение каналов) с сохранением пространственной привязки и метаданных исходных файлов [14].

Это необходимо для того, чтобы подвести итоги проделанной работы и проанализировать полученные результаты ускорения.

6.4. Ускорение вычислений с помощью вейвлетов

Про данный метод также было написано ранее. Отмечу лишь, что этот метод достаточно продвинутым и обладает масштабируемостью. Так что её изучение и реализация считаю крайне важным пунктом работы.

6.5. Реализация *resizing*

На основе формализации операции *resizing* (см. определение 6.3 и [29]) запланированы два этапа реализации:

- многопоточная CPU-версия с сохранением порядка элементов в линейной нумерации и поддержкой форматов NetCDF/GeoTIFF (базовые данные: NOAA NCEP Reanalysis, примеры из ArcGIS и Panoply);
- порт на GPU (CUDA) с сопоставлением потоков линейным индексам и сравнением скорости с CPU, включая профилирование копирования и вычислений.

7. Текущие результаты

Выполнено

- Проведено изучение и качественный обзор литературы, написан отчет по результатам всей проделанной работы, чем является данный документ.
- Посещена международная конференция Суперкомпьютерные дни в России, на которую была подготовлена публикация и выступление с ним. Посещено множество других выступлений и тематических секций.
- Получен доступ к Яндекс.Облаку с графическими картами NVIDIA Tesla V100 содержащими 5120 ядер CUDA, позволяющих выполнять высокопроизводительные вычисления (High Performance Computing, HPC).
- Подготовленный и загружены на сервер данные NCEP/DOE Reanalysis 2 в формате NetCDF (.nc). Реализована продвинутая операция *resizing* на уровне многопоточного CPU и базовая версия на GPU на сервере.

8. Перспективы на будущее

Дальнейшая работа будет направлена на доведение выбранной модели данных до практического вычислительного конвейера для массовой обработки геопривязанных тензоров. В первую очередь планируется унифицировать представление входных форматов (GeoTIFF и NetCDF) на уровне единого абстрактного тензора с явными осями, метаданными и поддержкой NA/NoData, чтобы операции растровой алгебры и reshaping могли применяться к данным без ручной подгонки размеров и порядка осей. Параллельно будет усилен конвейер ввода-вывода: переход к плиточной обработке (chunking), закреплённой памяти хоста и асинхронным копированиям с перекрытием чтения, вычислений и записи, чтобы минимизировать простоя GPU.

Следующий этап — реализация и оптимизация ключевых операций растровой алгебры на GPU (NDVI, ресемплинг усреднением, а также базовые поэлементные и окрестностные операции), включая профилирование, оценку пропускной способности памяти и выбор оптимальных параметров сетки потоков. Для подтверждения практической ценности планируется систематическое сравнение с CPU-реализациями и утилитами экосистемы GDAL/QGIS по времени, масштабируемости и затратам на ввод-вывод на наборах различного разрешения и объёма.

Список литературы

- [1] Ahmad Sheikh Abdallah. *Terrain Forest Dataset Showing a Surface Level of Detail for Different Pyramid Levels*. Figure from the doctoral dissertation “The Use of Geographical Information Systems for 3D Urban Models Reconstruction from Aerial Lidar Data”. 2010. URL: https://www.researchgate.net/figure/Terrain-Forest-Dataset-Showing-a-Surface-Level-of-Detail-for-Different-Pyramid-Levels_fig5_48546734 (дата обр. 04.07.2025).
- [2] ArcGIS. *The ArcGIS Imagery Book: New View. New Vision*. Esri Press, 2016.
- [3] Peter Baumann и др. ``Array Databases: Concepts, Standards, Implementations''. B: *Journal of Big Data* 8 (2021), c. 28. DOI: 10.1186/s40537-020-00399-2. URL: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00399-2>.
- [4] Tom B. Brown и др. ``Language Models are Few-Shot Learners''. B: *Advances in Neural Information Processing Systems* 33 (2020), c. 1877—1901.
- [5] Gilberto Câmara и др. ``Towards a generalized map algebra: principles and data types''. B: *Proceedings of the Brazilian Symposium on GeoInformatics (GeoInfo)*. 2005, c. 66—81.
- [6] Jesús Carabaño, Jan Westerholm и Tapani Sarjakoski. ``A compiler approach to map algebra: automatic parallelization, locality optimization, and GPU acceleration of raster spatial analysis''. B: *GeoInformatica* 22 (2018), c. 211—235.
- [7] Sharan Chetlur и др. ``cuDNN: Efficient Primitives for Deep Learning''. B: *arXiv preprint arXiv:1410.0759* (2014).
- [8] NVIDIA Corporation. *CUDA C++ Programming Guide*. Developer documentation. 2025. URL: <https://docs.nvidia.com/cuda/c-programming-guide/> (дата обр. 04.07.2025).
- [9] Суперкомпьютер "cHARISMa" НИУ ВШЭ. 2024. URL: <https://hpc.hse.ru/hardware/hpc-cluster>.
- [10] Iddo Drori и Dani Lischinski. ``Fast Multi-Resolution Image Operations in the Wavelet Domain''. B: *IEEE Transactions on Visualization and Computer Graphics* 9.3 (2003), c. 264—273.

- [11] A. Fuerst и др. *GIS Based Terrain Analysis with GPU and CPU Strategies*. Poster presentation. 2016. URL: https://faculty.salisbury.edu/~ealu/reu/Projects_File/2016/Lembo.pdf (дата обр. 03.07.2025).
- [12] Alex Fuerst, Charles Kazer и William Hoffman. *GIS based terrain analysis with GPU and CPU strategies*. 2016. URL: https://faculty.salisbury.edu/~ealu/reu/Projects_File/2016/Lembo.pdf.
- [13] GDAL Project Steering Committee. *GTiff--GeoTIFF File Format (GDAL Raster Driver Documentation)*. Online Documentation. Accessed 4 July 2025. 2025. URL: <https://gdal.org/en/stable/drivers/raster/gtiff.html>.
- [14] GDAL/OGR contributors. *gdal_calc — Command-line raster calculator with NumPy syntax*. GDAL documentation. Open Source Geospatial Foundation. 2025. URL: https://gdal.org/en/stable/programs/gdal_calc.html (дата обр. 04.07.2025).
- [15] Soeren Gebbert, Edzer Pebesma и Noel Gorelick. ``Processing Remote Sensing Data with GRASS GIS on Google Earth Engine''. В: *ISPRS International Journal of Geo-Information* 8.9 (2019), с. 393. DOI: 10.3390/ijgi8090393.
- [16] Mike Houston. *General Purpose Computation on Graphics Processors (GPGPU)*. Presentation. 2007. URL: https://graphics.stanford.edu/~mhouston/public_talks/R520-mhouston.pdf (дата обр. 03.07.2025).
- [17] Norman P. Jouppi и др. ``A Domain-Specific Supercomputer for Training Deep Neural Networks''. В: *Proceedings of the 47th International Symposium on Computer Architecture*. 2020.
- [18] S. P. Kirby, W. B. Kostan и A. J. Lembo. *Parallelizing Raster-Based Functions in GIS with CUDA C*. Poster presentation. 2013. URL: <https://faculty.salisbury.edu/~ajlembo/cudaposter.pdf> (дата обр. 03.07.2025).
- [19] Sean P. Kirby, William B. Kostan и Arthur J. Lembo. *Parallelizing Raster-Based Functions in GIS with CUDA C*. 2013. URL: <https://faculty.salisbury.edu/~ajlembo/cudaposter.pdf>.
- [20] Markus Kurth и др. ``Harnessing Hopper Tensor Cores for Exascale-Class HPC''. В: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'24)*. 2024.
- [21] *Landsat Science*. <https://landsat.gsfc.nasa.gov/>.

- [22] Jeremy Mennis. ``Multidimensional map algebra: design and implementation of a spatio-temporal GIS processing language''. B: *Transactions in GIS* 14.1 (2010), c. 1—21. DOI: 10 . 1111 / j . 1467 - 9671 . 2009 . 01179 . x.
- [23] NASA GISS. *Panoply NetCDF/GRIB/HDF Data Viewer*. Visualization tool for multidimensional scientific data. 2025. URL: <https://www.giss.nasa.gov/tools/panoply/download/> (дата обр. 04.07.2025).
- [24] NOAA Physical Sciences Laboratory. *NCEP Reanalysis 2: Pressure-Level Fields (NetCDF, variable × year)*. Annual NetCDF files for variables air, hgt, omega, rhum, uwnd, vwnd (1979–2025). 2025. URL: <https://downloads.psl.noaa.gov/Datasets/ncep.reanalysis2/pressure/> (дата обр. 04.07.2025).
- [25] NOAA Physical Sciences Laboratory. *NOAA PSL Home*. Data and Imagery archive for climate and hydrologic applications. 2025. URL: <https://psl.noaa.gov/> (дата обр. 04.07.2025).
- [26] Ivan V Oseledets. ``Tensor-train decomposition''. B: *SIAM Journal on Scientific Computing* 33.5 (2011), c. 2295—2317.
- [27] Ramon Antonio Rodriges Zalipynis. ``BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS''. B: *Proceedings of the VLDB Endowment* 13.12 (2020), c. 2909—2912. DOI: 10.14778/3415478.3415506. URL: <https://www.vldb.org/pvldb/vol13/p2909-zalipynis.pdf>.
- [28] Ramon Antonio Rodriges Zalipynis. ``ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud''. B: *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. Amsterdam, The Netherlands: ACM, 2019, c. 1985—1988. DOI: 10.1145/3299869.3320242. URL: <https://doi.org/10.1145/3299869.3320242>.
- [29] Ramon Antonio Rodriges Zalipynis. ``ChronosDB: Distributed, File Based, Geospatial Array DBMS''. B: *Proceedings of the VLDB Endowment* 11.10 (2018), c. 1247—1261. DOI: 10.14778/3231751.3231754. URL: <https://www.vldb.org/pvldb/vol11/p1247-zalipynis.pdf>.
- [30] Sentinel. *Sentinel Data. Copernicus satellite missions*. <https://sentivista.copernicus.eu>.
- [31] Matthias Steinbach и Reinhard Hemmerling. ``Accelerating batch processing of spatial data in GRASS GIS by utilizing GPUs''. B: *Proceedings of FOSS4G Conference*. Nottingham, UK, 2012, c. 1—10.

- [32] The Beard Sage. *CUDA – Memory Hierarchy*. Blog post. 2020. URL: <http://thebeardsage.com/cuda-memory-hierarchy/> (дата обр. 04.07.2025).
- [33] Dana C. Tomlin. *GIS and Cartographic Modeling*. 2nd. Redlands, California: ESRI Press, 2012.
- [34] Dana C. Tomlin. "Map algebra: one perspective". B: *Landscape and Urban Planning* 30.1-2 (1994), c. 3—12. DOI: 10.1016/0169-2046(94)90063-9.
- [35] Unknown. *Haar Wavelet Image Compression*. Lecture Notes, Math 572. n.d. URL: <https://mathweb.ucsd.edu/~jbrualdi/math572/HaarWaveletNotes.pdf>.
- [36] Michael Mau Fung Wong, Jimmy Chi Hung Fung и Peter Pak Shing Yeung. "High-resolution calculation of the urban vegetation fraction in the Pearl River Delta from the Sentinel-2 NDVI for urban climate model parameterization". B: *Geoscience Letters* 6.1 (2019), c. 1—10. DOI: 10.1186/s40562-019-0132-4. URL: <https://geoscienceletters.springeropen.com/articles/10.1186/s40562-019-0132-4> (дата обр. 04.07.2025).
- [37] Yubin Xia и др. "GPU acceleration of geospatial analysis methods". B: *Computers & Geosciences* 37.3 (2011), c. 296—305. DOI: 10.1016/j.cageo.2010.05.017.
- [38] Yuhua Yang и др. "Updating urban maps using high-resolution remote sensing imagery by combining multiscale and multidirectional features". B: *ISPRS Journal of Photogrammetry and Remote Sensing* 140 (2018), c. 104—118. DOI: 10.1016/j.isprsjprs.2017.05.002.
- [39] Ramon Antonio Rodriges Zalipynis. "Array DBMS: Past, Present, and (Near) Future". B: *PVLDB* 14.12 (2021), c. 3186—3189.
- [40] Ramon Antonio Rodriges Zalipynis. *Part 2. Array Data Models and Query Languages*. 2021. URL: <https://youtu.be/T8n8gQhWHN0>.
- [41] Xiaoqiang Zhu и др. "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources". B: *IEEE Geoscience and Remote Sensing Magazine* 8.4 (2020), c. 8—36.
- [42] А. В. Зорин и М. А. Федоткин. *Методы Монте-Карло для параллельных вычислений*. Москва: Издательский дом МГУ, 2013, с. 192. ISBN: 978-5-211-06530-7. URL: <https://msupress.com/catalogue/books/book/metody-monte-karlo-dlya-parallelnykh-vychisleniy/> (дата обр. 04.07.2025).