

# 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 раstra. Растры 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. Растры высокого (или низкого) разрешения имеют сравнительно меньшее расстояние между ячейками сетки и больше ячеек, чем растры 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 растров 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 пиксела) объем хранимых данных резко растёт – например, при двукратном 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 растров 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 используются 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 [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 растры) 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 растров, 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. Основным узким моментом — 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 высокдетальным 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 различных операций [10–13].

## Литература

1. Rodrigues 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/~ajlembo/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](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. Rodrigues Zalipynis R.A., BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. Proc. VLDB Endow. 2020, 13, 2909–2912.
11. Rodrigues Zalipynis R.A., ChronosDB: Distributed, File Based, Geospatial Array DBMS. Proc. VLDB Endow. 2018, 11, 1247–1261.
12. Rodrigues 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.