

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.

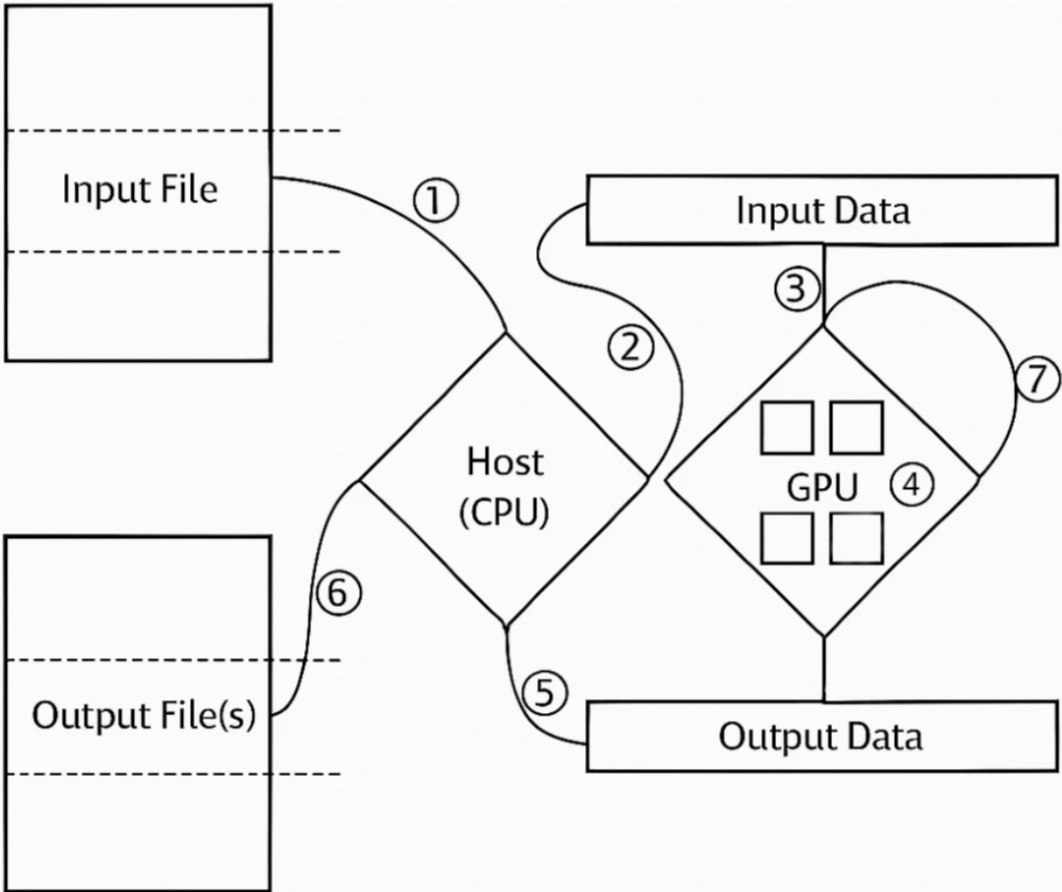
Цель работы

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 параллелить 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 остаётся 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