Гибридный параллельный решатель в атмосферном динамическом ядре модели Земной системы ИВМ РАН

И. В. Цыбулин[1], М. А. Тарасевич[2,3], Е. М. Володин[2], А. С. Грицун[2]

[1]Яндекс.Технологии

[2]Институт вычислительной математики им. Г. И. Марчука РАН

[3]Гидрометеорологический научно-исследовательский центр РФ

tsybulinhome@gmail.com

29 сентября 2025 г.

Введение

- Модель Земной системы ИВМ РАН разрабатывается уже более 40 лет[1], включает в себя
  - модель атмосферы
  - аэрозольный блок
  - модель океана и морского льда

  Используется для расчетов на 100–1000 лет
- В текущей реализации представляет собой MPI-приложение для CPU кластера
- Текущая производительность для различных разрешений:
  - $2^\circ \times 1.5^\circ \times L21$ в атмосфере, $1^\circ \times 0.5^\circ \times L40$ в океане, около 35 лет/сутки на 160 ядрах
  - $1.25^\circ \times 1^\circ \times L73$ в атмосфере, $0.5^\circ \times 0.25^\circ \times L40$ в океане, около 10 лет/сутки на 900 ядрах
  - $0.67^\circ \times 0.5^\circ \times L73$ в атмосфере, $0.167^\circ \times 0.125^\circ \times L40$ в океане, около 2 лет/сутки на 1000 ядрах

[1] Volodin E.M. et al.: Development of the Institute of Numerical Mathematics, Russian Academy of Sciences, Earth Climate System Model. Izv. Atmos. Ocean. Phys. **61** 301–310 (2025)

Мотивация работы

- Можно ли получить ускорение от перехода на гибридный подход MPI+OpenMP
- Иметь возможность запустить модель на системах с GPU
- Понять, возможно ли текущие алгоритмы модели адаптировать для GPU
- Оценить, какую производительность можно ожидать в результате

Модель атмосферы

Типичное распределение времени работы:

- Динамическое ядро ($\sim 60\%$ времени)
  - Явные схемы в динамике ($\sim 12\%$ времени)
  - **Решатель уравнения типа Гельмгольца** ($\sim 13\%$ времени)
  - Обмены ($\sim 7\%$ времени)
  - Вертикальные процессы ($\sim 7\%$ времени)
  - Микрофизика ($\sim 20\%$ времени)
- Радиационный блок ($\sim 25\%$ времени)
- Физические параметризации ($\sim 15\%$ времени)

Решатель уравнения типа Гельмгольца — ключевой компонент реализации полунеявной по времени схемы.

Уравнение типа Гельмгольца

Возникает в результате полунеявной по времени дискретизации слагаемых в системе уравнений динамики. Представляет собой трехмерное уравнение эллиптического типа $u - \frac{\tau^2}{a^2}\mathcal{B}\Delta_{\varphi,\lambda}u = H$, аппроксимированное следующим образом:

$$u^k_{i,j}m_j - \frac{\tau^2}{a^2}\sum_{k'=1}^{n_{lev}} B^k_{k'}\left(\frac{u^{k'}_{i+1,j} - 2u^{k'}_{i,j} + u^{k'}_{i-1,j}}{\Delta\lambda^2 m_j} + \right.$$
$$\left. + \frac{m_{j+1/2}(u^{k'}_{i,j+1} - u^{k'}_{i,j}) - m_{j-1/2}(u^{k'}_{i,j} - u^{k'}_{i,j-1})}{\Delta\varphi^2}\right) = H^k_{i,j}m_j$$

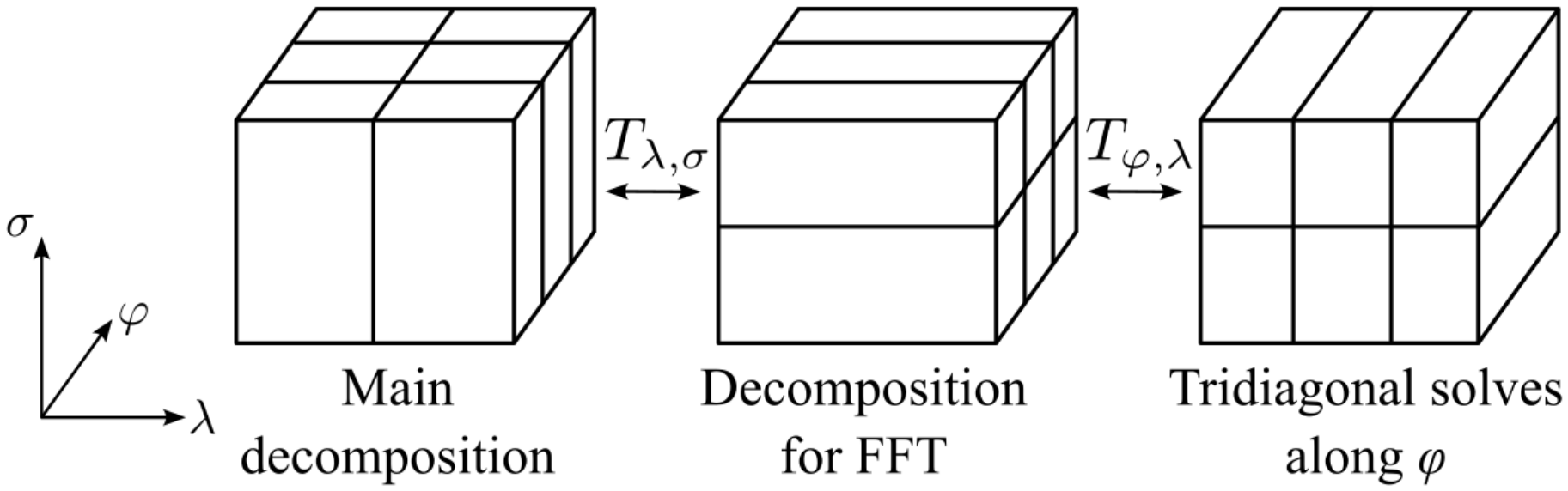
Здесь $B^k_{k'}$ — постоянный вертикальный оператор, $m_j = \cos\varphi_j$.

Метод решения уравнения типа Гельмгольца

Применяется метод расщепления по переменным:

1. Переход в базис собственных функций вертикального оператора B: исходное трехмерное уравнение расщепляется на набор независимых двумерных задач для каждой вертикальной моды
2. Применяется дискретное преобразование Фурье вдоль каждого круга широты, что приводит к независимым задачам для каждой гармоники решения
3. Для каждой вертикальной моды и каждой гармоники задача сводится к системе с трехдиагональной матрицей и решается независимо вдоль меридианов
4. Выполняется обратное дискретное преобразование Фурье, восстанавливающее решение вдоль долгот из гармоник
5. Вертикальное распределение восстанавливается путем суммирования отдельных вертикальных мод

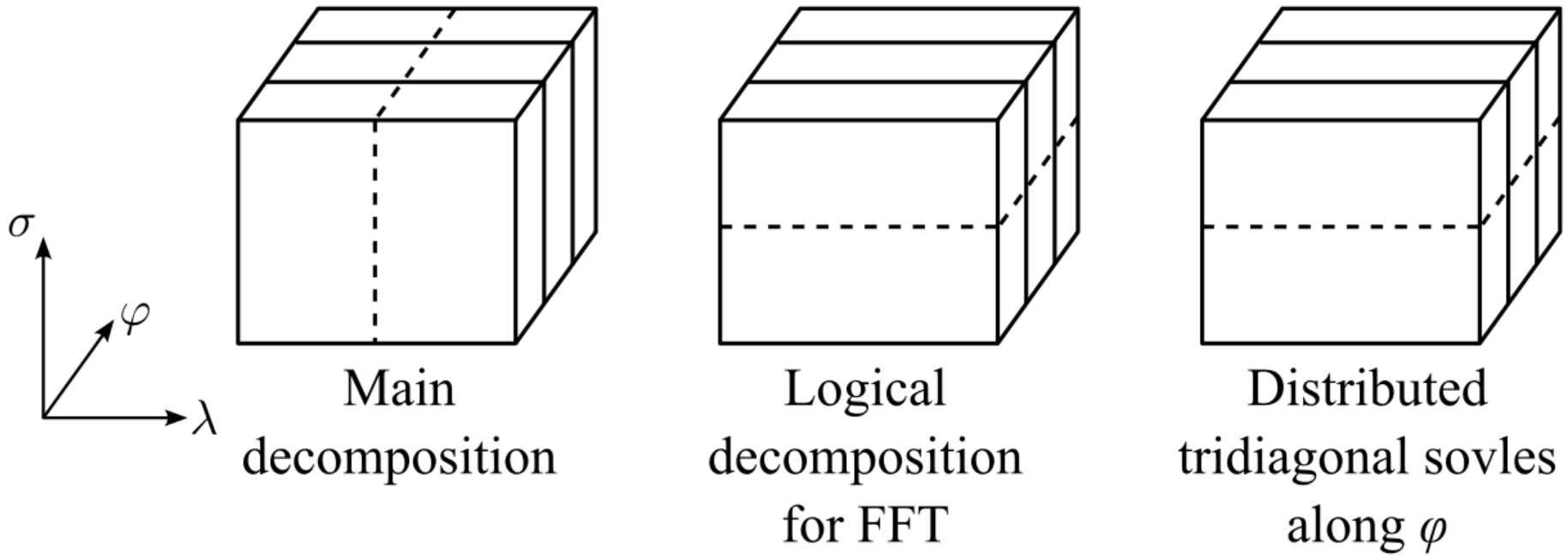
Существующий MPI решатель



Используется два типа глобальных транспонирований данных

- Данные собираются целыми кругами широты перед выполнением БПФ
- Данные собираются целыми меридианами перед решением трехдиагональной системы
- Те же транспонирования проводятся в обратном порядке для заключительных шагов

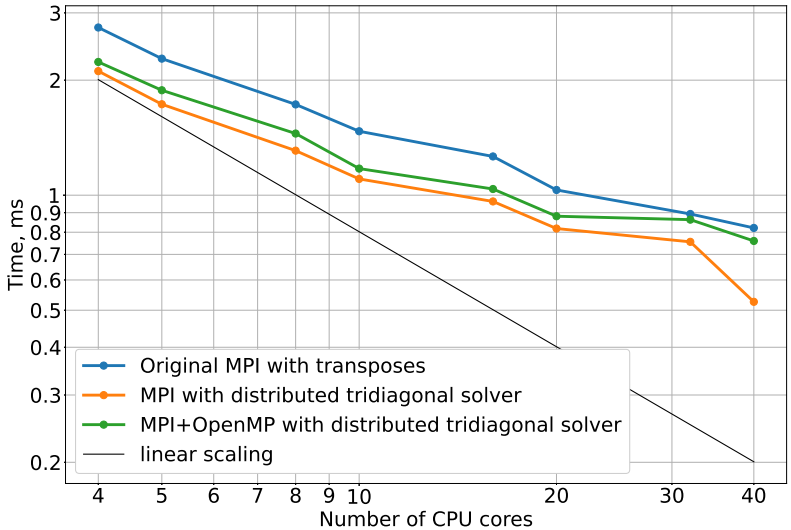
Решатель без транспонирования данных



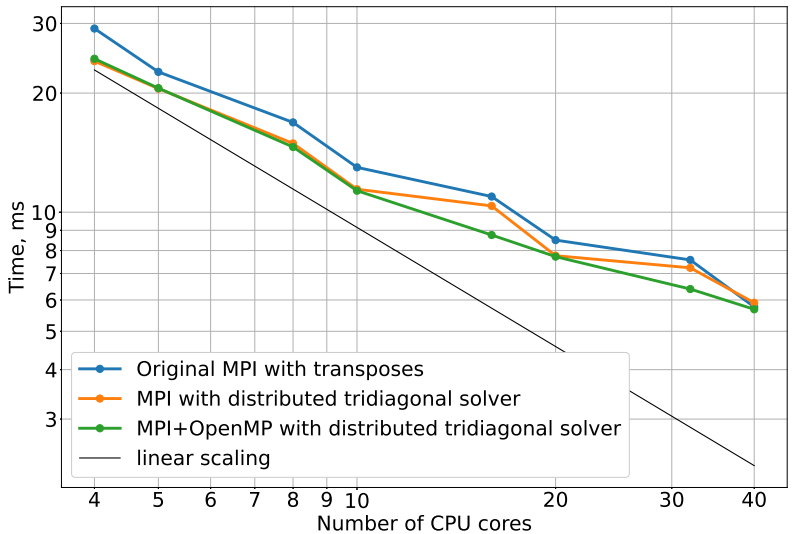
- Транспонирование $\sigma \leftrightarrow \lambda$ можно убрать, если использовать MPI разбиение вдоль широт и **OpenMP разбиение** вдоль долгот/высот
- Транспонирование $\lambda \leftrightarrow \varphi$ можно убрать, если делать **распределенную**[2] прогонку

[2] Mattor M., Williams T. J., Hewett D. W.: Algorithm for solving tridiagonal matrix problems in parallel. Parallel Computing **21** 1769 (1995)

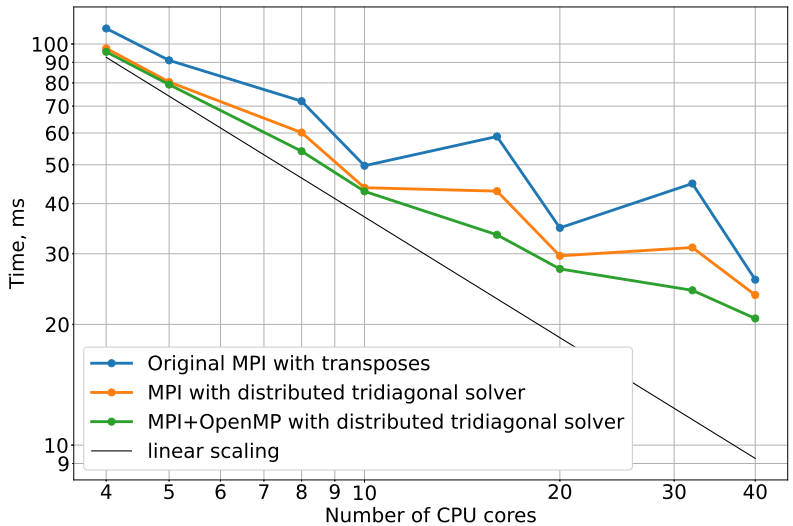
Решатель без транспонирования данных: MPI vs OpenMP



a) 6P ($120 \times 180 \times L21$)



b) 6M ($180 \times 288 \times L73$)



c) 6H ($360 \times 540 \times L73$)

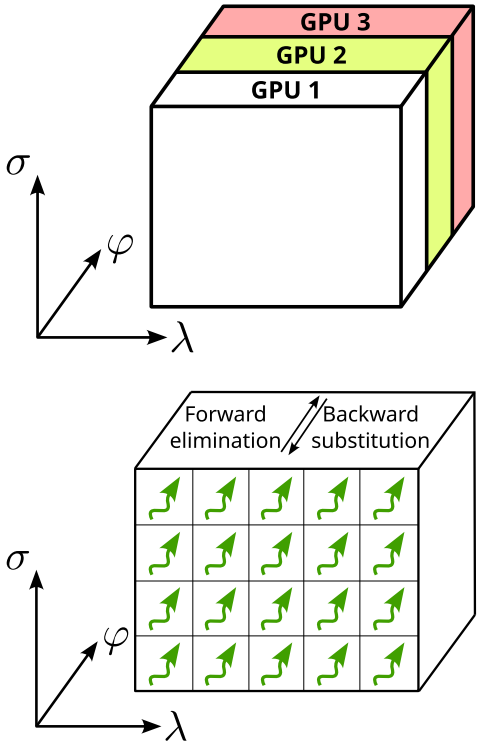
Выводы по гибридному MPI+OpenMP подходу

- Ускорение сильно зависит от разрешения задачи, на грубом разрешении MPI вариант оказывается немного быстрее, на подробном — гибридный MPI+OpenMP вариант
- Даже без использования OpenMP можно ускорить код за счет распределенной прогонки
- Код эффективной OpenMP реализации получается сложным, требует работы в одном параллельном регионе
- Ни MPI, ни гибридный MPI+OpenMP подход не показывают хорошего масштабирования внутри узла, в обоих случаях эффективность достигает лишь 40% при полной утилизации

Реализация на GPU

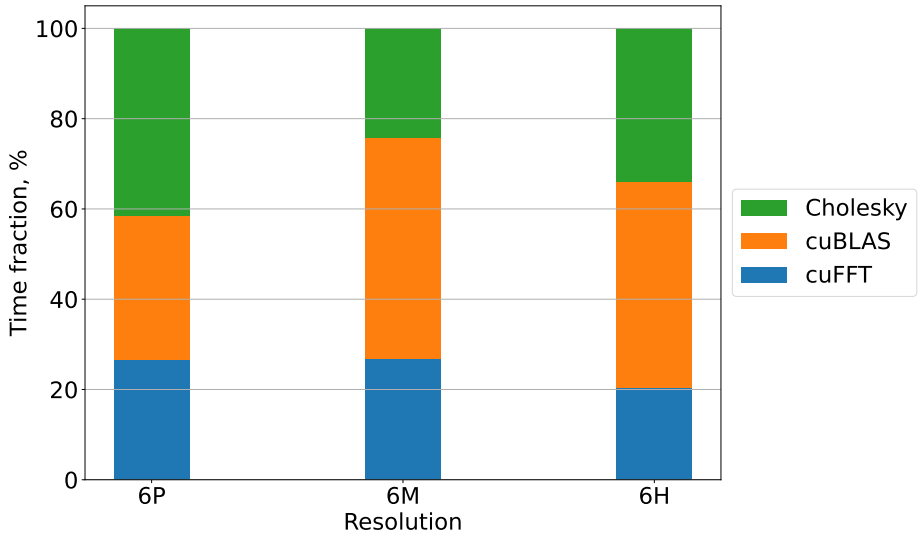
Идейно следует тем же шагам, что и MPI+OpenMP реализация:

1. Переход от вертикальных уровней к модам вертикального оператора — умножение на плотную матрицу (cuBLAS)
2. Преобразование Фурье вдоль кругов широты — одномерные БПФ (cuFFT)
3. Решение трехдиагональной системы методом Холецкого
4. Обратное преобразование Фурье вдоль кругов широты — одномерные БПФ (cuFFT)
5. Переход от мод к вертикальным уровням — умножение на плотную матрицу (cuBLAS)

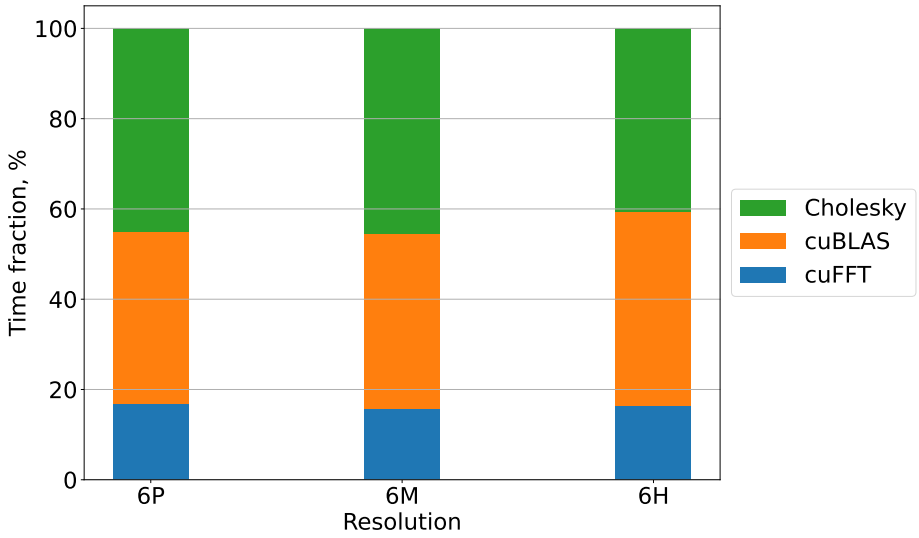


Шаг 3 на данный момент реализован для случая **одного GPU**.

Распределение времени работы GPU решателя

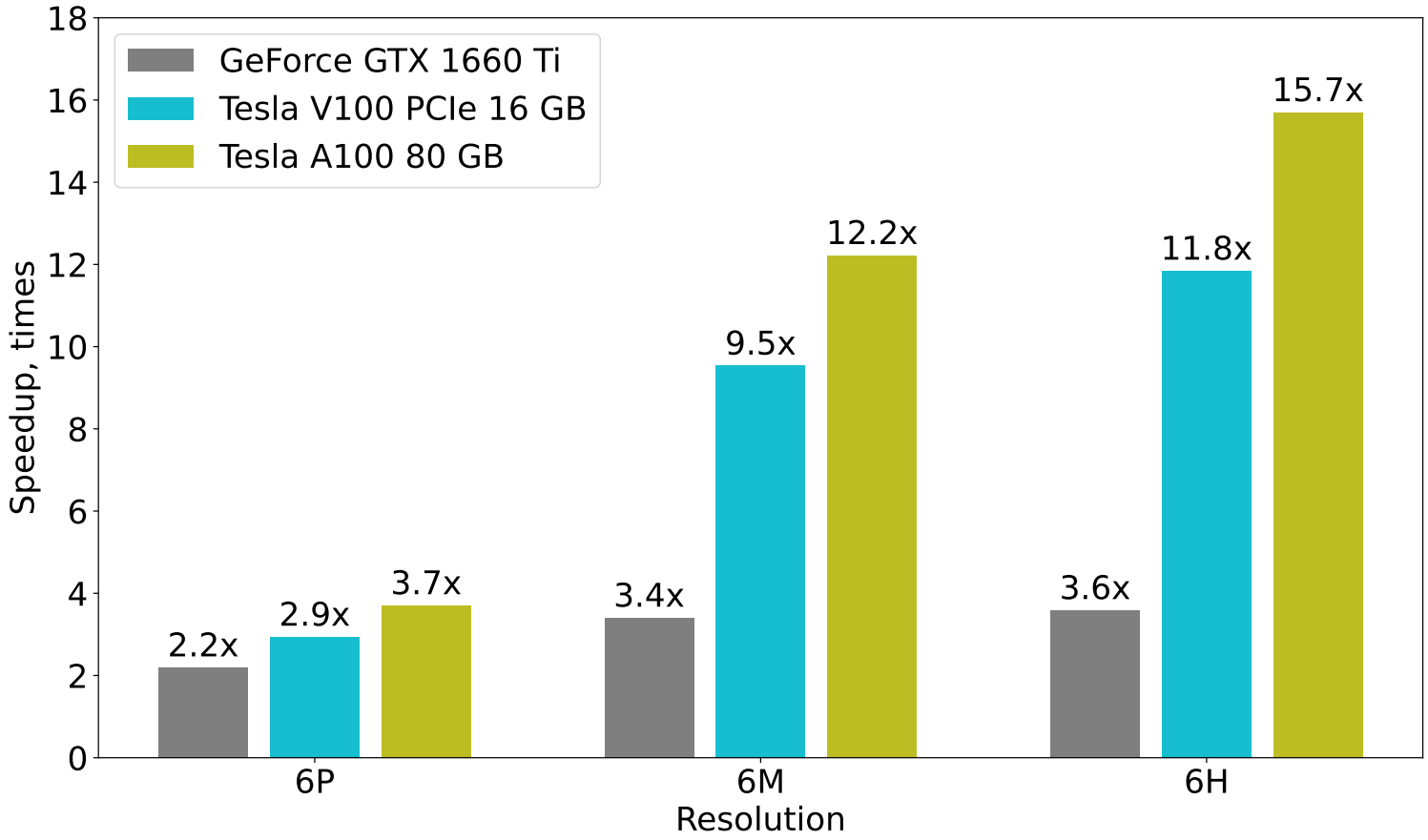


a) GeForce GTX 1660 Ti



b) Tesla A100

Ускорение GPU реализации



- Сравнивается время работы с использованием одного GPU и время самой быстрой параллельной CPU реализации на 40 ядрах (1 узел CPU кластера)
- Время работы **не учитывает** копирование данных на GPU и обратно

Выводы

- GPU потребительского класса на этой задаче показывает производительность 2–3 узлов CPU кластера, а специализированные ускорители — более, чем 10 узлов
- Если учитывать время передачи данных между CPU RAM и GPU vRAM, это ускорение теряется
- Чтобы сохранить производительность в реальных задачах, необходимо вынести на GPU все динамическое ядро модели атмосферы

Спасибо за внимание!

Вопросы?

tsybulinhome@gmail.com