МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М.В.ЛОМОНОСОВА

Факультет «Вычислительной математики и кибернетики»

Отчет

по курсу «Суперкомпьютерное моделирование и технологии

Выполнил: *Новикова Анастасия Вячеславовна*

Группа: *620*

Москва, *2025*

Последовательная задача

Задача

$$sin((2 * pi * x)/L\_x) * sin((pi * y)/L\_y) * sin((pi * z)/L\_z) * cos(a * t + t + 2 * pi),$$

$$at = 1/2 * sqrt(4/L\_x\^2 + 1/L\_y\^2 + 1/L\_z\^2),$$

$$a\^2 = 1/(4 * pi\^2)$$

Время

Сетка: 256, шаг времени: 0.00142259, шагов: 20

Общее время работы программы: 19.5647 сек

Максимальная погрешность: 5.91572e-05

Время вычислений: 17.7798 сек

Распараллеливание с помощью OpenMP

Основные измерения

Число потоков OpenMP (Np)	Число точек сетки (N^3)	Время решения (T)	Ускорение (S)	Погрешность (δ)	Относительная δ
1	128^3	5.83		0.00438	0.02%
2	128^3	3.125	1.87	0.00438	0.02%
4	128^3	1.632	3.57	0.00438	0.02%
8	128^3	0.922	6.32	0.00438	0.02%
16	128^3	0.57	10.23	0.00438	0.02%
2	256^3	17.46		0,0039	0.01%
4	256^3	12.28	1.42	0,0039	0.01%
8	256^3	6.682	2.61	0,0039	0.01%
16	256^3	4.31	4.05	0,0039	0.01%
32	256^3	3.24	5.39	0,0039	0.01%

Оценка эффективности

За эталон была принята **исходная последовательная программа без MPI и OpenMP**.
Её времена выполнения:

- сетка **128^3**:

  T =9 с

- сетка 256^3:

  T=19 с

Время вычислялось только по вычислительной части.

Ускорение вычислялось по отношению каждого времени к последовательной программе.

Также проверялась абсолютная погрешность, ожидания были, что при увеличении сетки она будет уменьшаться, а при изменения нитей – нет. По таблице видно, что погрешность вела себя ожидаемым образом.

Пояснение результатов

Сетка 128:

Ускорение почти линейное — чем больше потоков, тем быстрее работает.

При очень большом количестве потоков наблюдается эффект, когда ускорение даже выше ожидаемого. Это потому что:

- кэш используется
- меньше данных приходится на каждый поток

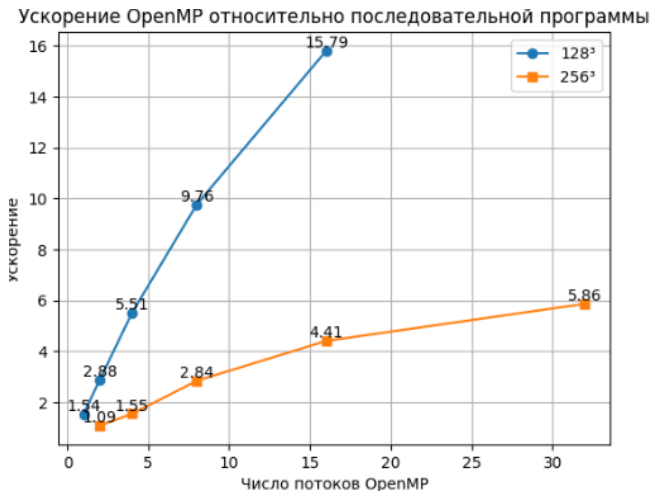
Сетка 256:

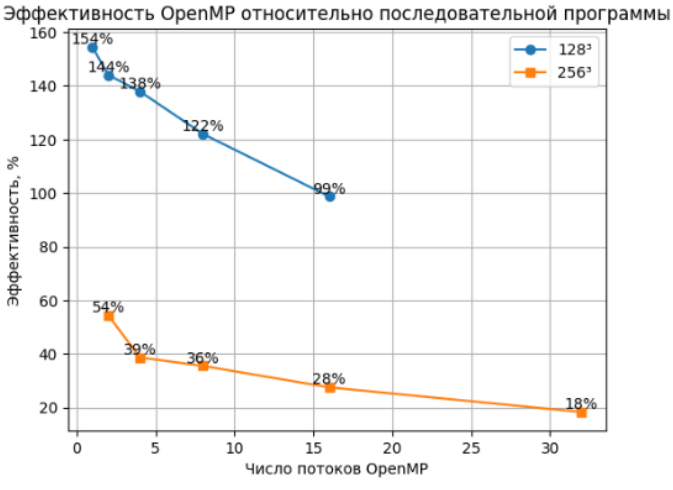
Здесь ускорение растёт медленнее. После какого-то количества потока рост почти останавливается.

Причины:

- данных становится слишком много
- растёт количество обращений к памяти или обменов между процессами
- шина памяти не справляется — упираемся в свою пропускную способность.

  Графики





Распараллеливание с помощью MPI

Основные измерения

Число MPI процессов	Число точек сетки (в кубе)	Время решения	Ускорение	Погрешность
1	128	8.4		0.004
4	128	2.31	3.6	0.004
8	128	1.6	5.3	0.004
16	128	0.9	9.3	0.004
32	128	0.4	21.0	0.004
1	256	32.14		0.00388605
4	256	9.3	3.5	0.00388605
8	256	6.1	5.3	0.00388605
16	256	2.76	11.6	0.00388605
32	256	1.32	24.3	0.00388605
20	512	36.3	-	0.0019112

Оценка эффективности

Эффективность параллельной MPI-программы оценивалась на основе сравнения времени её работы с временем **обычной последовательной реализации**

За эталон была принята **исходная последовательная программа без MPI и OpenMP**.
Её времена выполнения:

- сетка $\mathbf{128^3}$:

  T =9 с

- сетка $\mathbf{256^3}$:

  T=19 с

Время выполнения MPI-программы измерялось с помощью функции `MPI_Wtime()` и включало только вычислительную часть алгоритма.

Ускорение вычислялось по отношению каждого времени к последовательной программе.

Таким образом, ускорение характеризует выигрыш во времени, достигнутый за счёт распараллеливания

Пояснение результатов

Сетка 128

- Наблюдается **быстрый рост ускорения** при увеличении числа MPI-процессов.
- При большом числе процессов ускорение существенно превышает линейное.

Эффективность:

- превышает 100% на малом числе процессов,
- постепенно снижается при увеличении процессов.

Это связано с:

- уменьшением объёма данных на один процесс
- снижением накладных расходов по сравнению с последовательной версией.

Сетка 256

- Ускорение также растёт с увеличением числа процессов,
- однако рост менее равномерный.

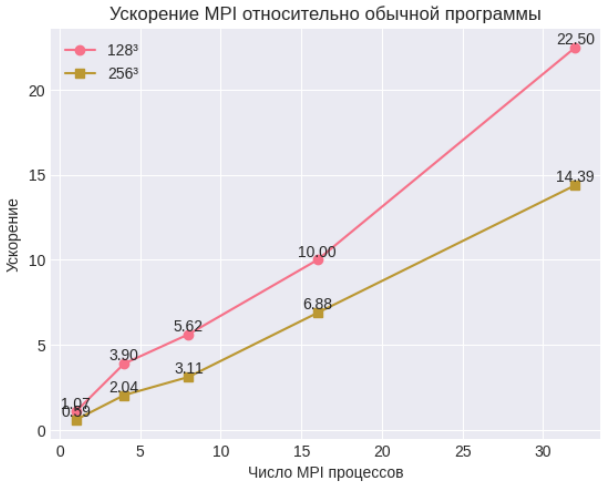
Эффективность:

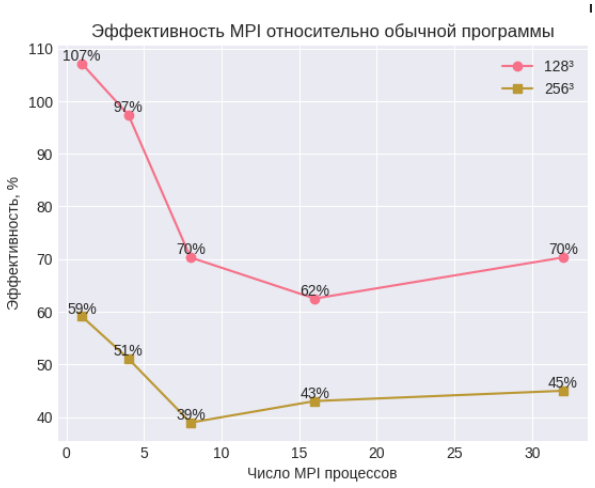
- ниже, чем для сетки 128^3,
- убывает быстрее при росте числа процессов.

Основные причины:

- увеличение объёма передаваемых граничных данных,
- рост накладных расходов на коммуникации (`MPI_Sendrecv`),
- ограничение пропускной способности памяти

Графики





Распараллеливание с помощью OpenMP+MPI

Основные измерения

Число MPI процессов	Число нитей	Число точек сетки (в кубе)	Время решения	Ускорение	Погрешность
4	1	128	2.42	-	0.008
4	2	128	1.86	1.30x	0.008
4	4	128	1.43	1.69x	0.008
4	8	128	1.32	1.83x	0.008
8	1	256	6.85	-	0.0042
8	2	256	5.12	1.34x	0.0042
8	4	256	3.94	1.74x	0.0042
8	8	256	3.45	1.99x	0.0042
20	8	512	16.23	-	0.00115

Оценка эффективности

Так как используются два уровня параллелизма, эффективность считалась по общему числу ядер**:**

$N_{cores} = N_{MPI} \times N_{OMP}$

$E = S / N_{cores} \cdot 100\%$

Сетка 128 (4 MPI процесса)

- Ускорение растёт при увеличении числа OpenMP-потоков
- Однако рост замедляется после 4–8 потоков

Эффективность:

- высокая при малом числе потоков
- заметно падает при увеличении общего числа ядер

Сетка 256 (8 MPI процесса)

- Ускорение растёт более равномерно
- Использование OpenMP даёт выигрыш поверх MPI

Эффективность:

- выше, чем для 128
- снижается медленнее при увеличении числа потоков

Пояснение результатов

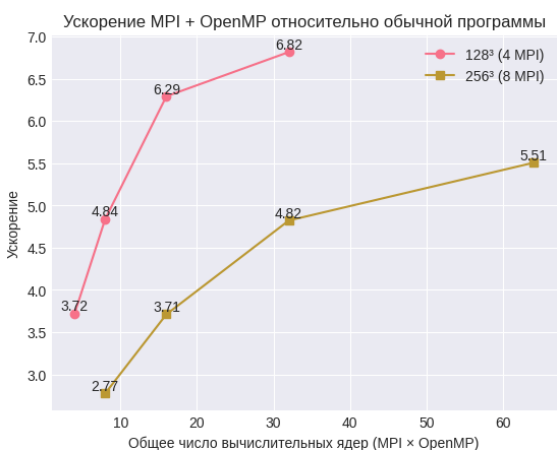
Причины для замедления роста для 128 сетки:

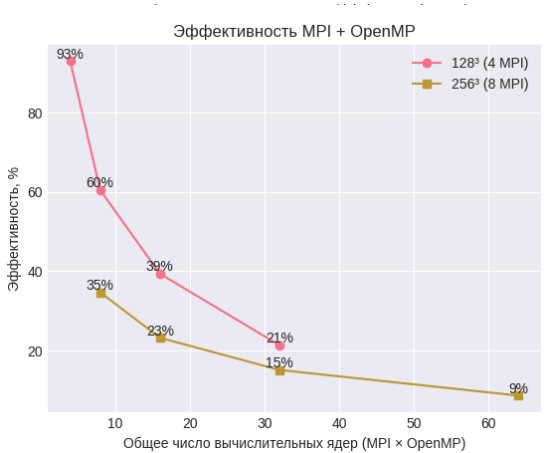
- сравнительно небольшой размер задачи
- рост накладных расходов OpenMP
- конкуренция потоков за память внутри одного MPI-процесса

Объяснение для сетки 256:

- большим объёмом вычислений на процесс
- лучшим соотношением вычислений и коммуникаций
- более эффективной загрузкой потоков OpenMP

Графики





Распараллеливание с помощью MPI+GPU

Основные измерения

Параметр	1 MPI процесс, 1 GPU	2 MPI процесса, 2 GPU
MPI процессов	1	2
GPU	1	2
Сетка	512	512
Общее время (сек)	3.0246	1.76172
Инициализация (сек)	0.0130395	0.00960617
Основной цикл (сек)	1.65274	1.21283
CUDA ядра (сек)	1.467	0.507026
MPI коммуникации (сек)	0.0420241	0.00163077
Копирование GPU→CPU (сек)	0.06409	0.0341015
Вычисления в цикле (сек)	1.4067	0.507026
Финализация (сек)	0.0532821	0.0531821
Физическое время (сек)	0.177475	0.177475
Погрешность	0.000470837	0.000470837
Скорость (Mpoints/sec)	8134.93	4439.35
Эффективность GPU	1.16%	2.19%
Ускорение	1.0	1.57

Оценка эффективности:

MPI процессов	OpenMP нитей	GPU	Сетка	Время
20	1	-	512	36.3
20	8	-	512	16.23
1	1	1	512	3.0246
2	1	2	512	1.76172

- Ускорение GPU в 9.4 раз быстрее гибридной программы для 2 GPU

Пояснение результата:

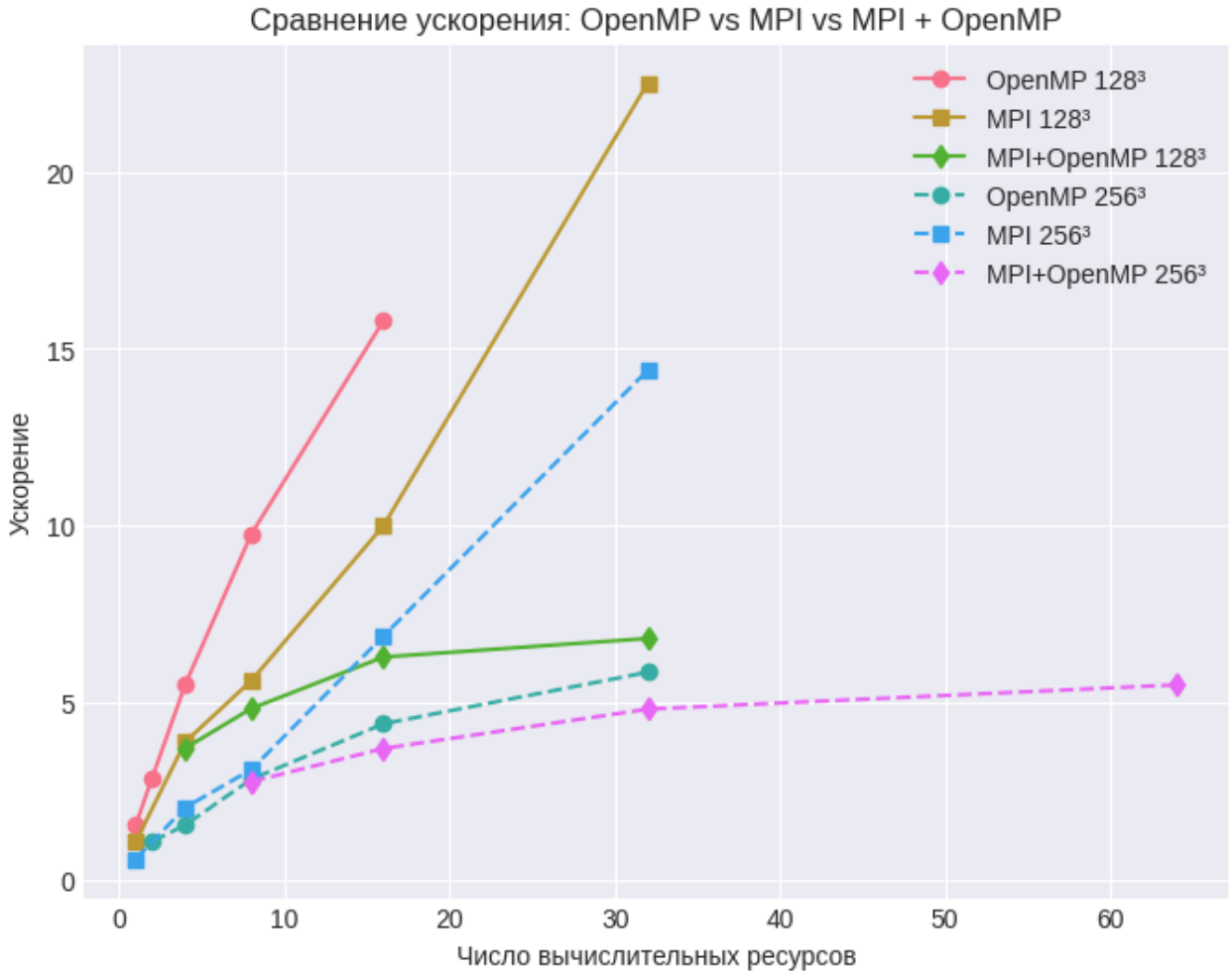
Анализ эффективности показывает, что только 1.16% времени на 1 GPU и 2.19% на 2 GPU приходится непосредственно на вычисления в ядрах CUDA. Это связано с тем, основное время тратится на:

- Копирование данных между GPU и CPU
- Подготовку данных перед вычислениями

При увеличении числа MPI процессов с 1 до 2 наблюдается ускорение 1.57×, что меньше идеального линейного масштабирования. Это объясняется ростом накладных расходов на:

- MPI коммуникации между процессами, которые увеличились
- Синхронизацию и обмен граничными данными

Графики параллельных реализаций



MPI лучше масштабируется с ростом числа ресурсов, чем OpenMP и гибридный метод.

OpenMP эффективен на малом числе ресурсов, но не выдерживает роста числа вычислителей.

Гибридный подход не показывает преимуществ в данной конфигурации и для этих размеров задач.

Заключение:
Для малых и средних задач предпочтительнее использовать OpenMP на ограниченном числе ресурсов. Для больших вычислительных ресурсов и задач небольшого объёма лучше применять MPI. Гибридный подход в рассматриваемой конфигурации не обеспечивает прироста производительности.

Однако для большой сетки гибридный подход показывает прирост. Для сетки 512. По сравнению просто с MPI программой выигрыш по времени в почти 3 раза, а GPU в почти 10 раз.