ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА»

Отчет по лабораторной работе №3

Модификация программы из л/р №1 для параллельной работы по технологии MPI.

Выполнили:

Мантров И.А.

гр. 6313-10.05.03D

Принял:

Минаев Е.Ю.

Самара 2025

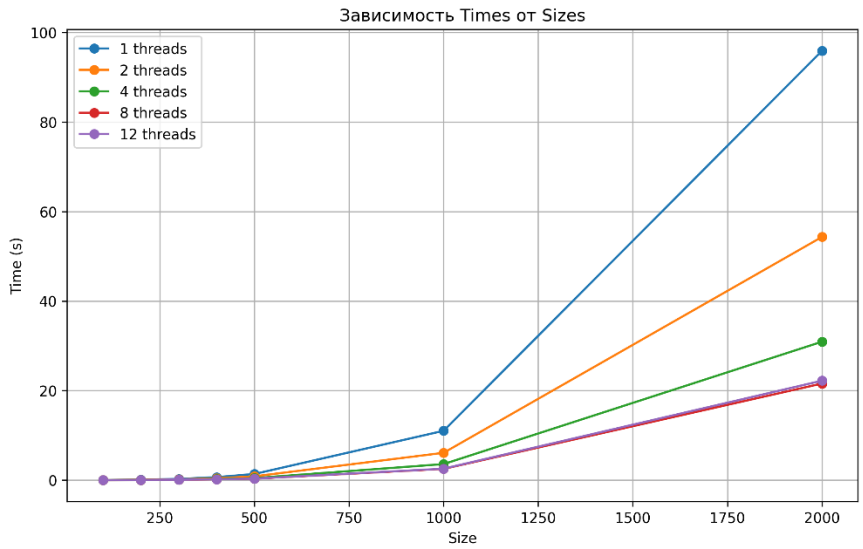
Структура лабораторной работы (1 часть):

- `matrix.h, matrix.cc` – класс матриц `Matrix`, в котором реализован метод перемножения матриц (перегрузка оператора *)
- `random_generator.h, random_generator.cc` – класс `RandomGenerator`, в котором реализована генерация матриц фиксированного размера с случайными элементами
- `stat.h, stat.cc` – класс ExecutionTimer для замера времени выполнения.
- `mpi.cc` - основная программа для выполнения параллельного матричного умножения с использованием MPI. Здесь реализованы функции для перемножения матриц `matrix_multiply_mpi()`, вспомогательные функции для работы с директорией и создания csv файла: `create_directory()`, `chdir()`, `write_csv_results()`
- `statistic.py` – модуль для верификации перемножения матриц с помощью вспомогательного средства numpy и для построения статистического графика.

В файле result расположены исходные матрицы и результаты их перемножения, а также в файле statistic.txt находится статистика времени перемножения матриц.

Результаты верификации расположены в файле compare_result.txt

Статистический график расположен в файле stat.png



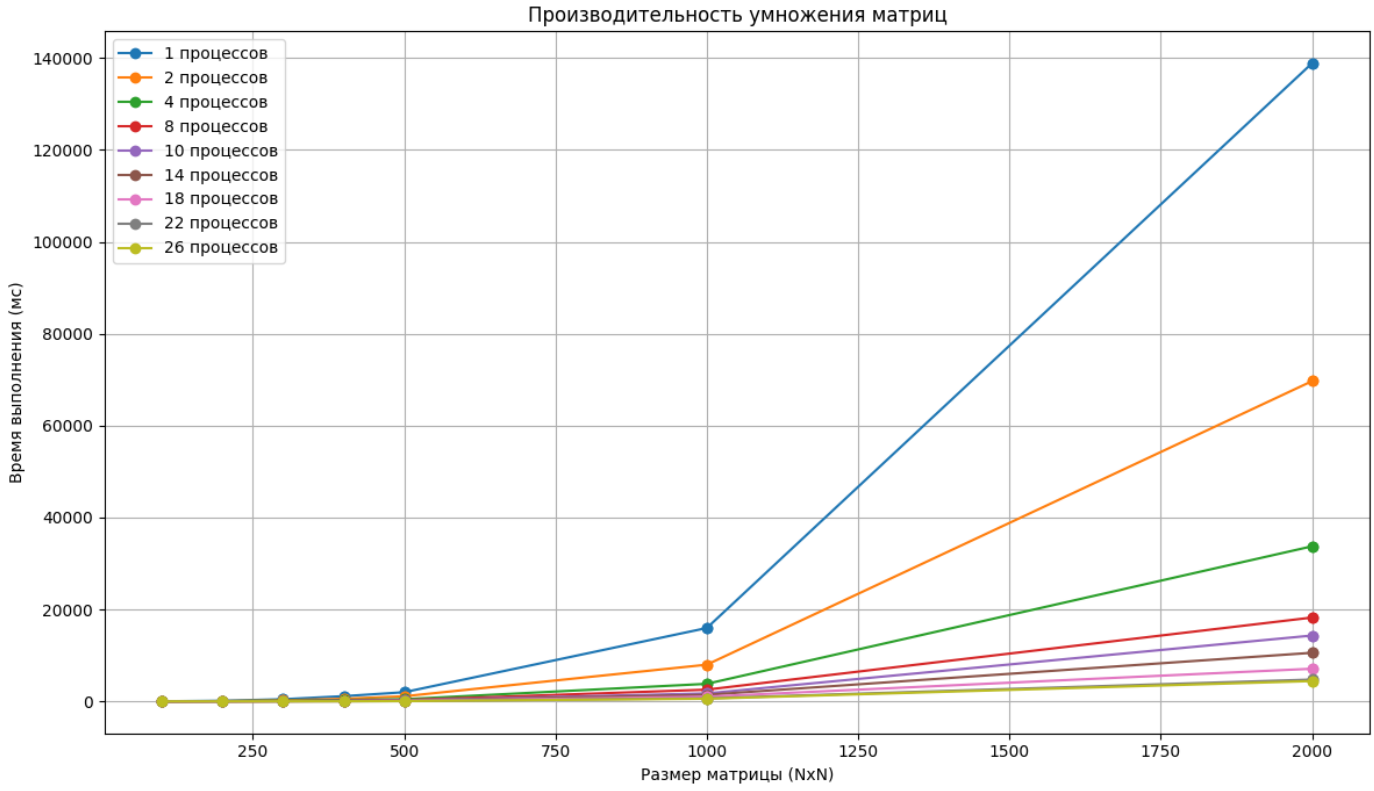
Структура лабораторной работы (2 часть):

- `mpi_super.cc` - основная программа для выполнения параллельного матричного умножения с использованием MPI (для работы с суперкомпьютером). В этом файле реализован просто класс матриц, со всеми необходимыми в условиях данной лабораторной работы, методами: конструктор, генератор рандомной матрицы, перемножение матриц, вывод). Функция перемножения матриц реализована вне класса. В `main()` реализована основная часть кода необходимая для корректной работы программы.(Отличие от mpi.cc заключается в исключении работы с директорией и разного рода работы с файлами, реализован обычный вывод в терминал, а с помощью использования start.pbs получаем файлы с данными, выводимыми в терминале.

В файле result расположены исходные матрицы и результаты их перемножения, а также в файле statistic.txt находится статистика времени перемножения матриц.

Результаты верификации расположены в файле compare_result.txt

Статистический график расположен в файле stat_sc.png



На основании анализа графиков производительности (stat.png и stat_sc.png) можно сделать следующие выводы о реализации матричного умножения с использованием технологии MPI:

1. **Локальные тесты (stat.png):**
   График демонстрирует четкую зависимость времени выполнения от количества потоков и размера матриц. Наблюдается существенное ускорение вычислений при увеличении числа потоков с 1 до 12, особенно заметное для матриц размером от 1000x1000 и выше. При этом графики для разного количества потоков сохраняют предсказуемую форму, что свидетельствует о стабильности работы алгоритма.
2. **Тесты на суперкомпьютере (stat_sc.png):**
   Результаты показывают исключительную масштабируемость решения при использовании до 26 процессов. Время выполнения операций сокращается пропорционально увеличению количества задействованных процессов, демонстрируя эффективность распараллеливания. Особенно впечатляющие результаты достигнуты для матриц размером 2000x2000, где при использовании 26 процессов время вычислений сократилось в несколько раз по сравнению с однопроцессным выполнением.
3. **Корректность вычислений:**
   Результаты верификации, приведенные в файле compare_result.txt, подтверждают абсолютную точность вычислений при использовании MPI-реализации. Отсутствие расхождений с эталонными значениями, полученными через модуль numpy, свидетельствует о правильности реализации алгоритма.
4. **Эффективность распараллеливания:**
   Оба графика наглядно демонстрируют, что технология MPI обеспечивает высокую степень параллелизма и отличную масштабируемость решения. При этом сохраняется линейная зависимость времени выполнения от сложности задачи, что соответствует теоретическим ожиданиям.

Таким образом, реализация матричного умножения с использованием MPI доказала свою эффективность как на локальных машинах, так и в условиях суперкомпьютерных вычислений, обеспечивая значительный прирост производительности при сохранении абсолютной точности результатов.