МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математика и суперкомпьютерное моделирование»

Отчет

по дисциплине «Суперкомпьютерное моделирование»

Специальность – 01.05.01 Фундаментальные математика и механика

Профиль подготовки – Вычислительная математика и вычислительная механика

Выполнил студент: Нестеров В.О.

Группа: 18ВФ1

Руководитель: к.т.н., доцент Антонов А.В.

2022

Message Passing Interface (MPI, интерфейс передачи сообщений) — программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком и другими.

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существуют его реализации для большого числа компьютерных платформ. Используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу.

Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Java, Си и C++.

В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP ориентирован на системы с общей памятью (многоядерные с общим кэшем). Обе технологии могут использоваться совместно, чтобы оптимально использовать в кластере многоядерные системы.

Реализуем умножение матриц на языке программирования С++, с применением инструмента распараллеливания MPI.

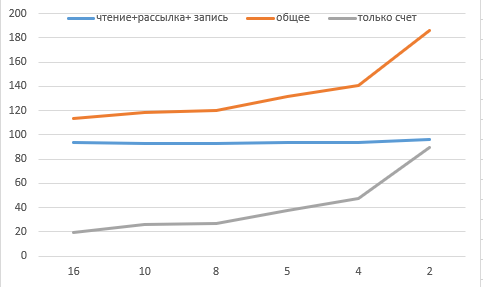
# Работа программы и результаты вычислений

В программе реализовано умножение друг на друга 2-х матриц размерами

. Вначале обе матрицы параллельно считываются на 0 и 1 процессе, следом матрица отправляется с помощью функции на все имеющиеся процессы целиком, а матрица с помощью функции .

Далее на каждом процессе происходит само перемножение матриц. В конце с помощью функции используя операцию редукции собираем на 0-м процессе готовую матрицу. Там же её записываем в файл.   
 По пути работы программы были расставлены временные метки, с помощью которых вычисляется время чтения матриц из файла, рассылки данных между процессами и записи матриц в файл.

Все полученные данные отражены на графике (Рисунок 1).



На графике видно, что при увеличении количества процессов, время счета уменьшается, аналогично уменьшается общее время работы программы, запись и чтение файлов остаются неизменными, так как используются одинаковые матрицы. В данной работе влияние времени рассылки на общее время работы программы остается минимальным, возможно из-за использования более мощных вычислительных машин чем в предоставленной аудитории.