Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Спецкурс: системы и средства параллельного программирования

Отчёт № 3
Оценка влияния числа процессов на время выполнения параллельного алгоритма поиска простых чисел с помощью "решета Эратосфена"

Работу выполнил
Чепурнов А. В.

Москва 2018

Постановка задачи и формат данных

Задача: Реализовать параллельный алгоритм поиска простых чисел в заданном диапазоне с помощью "решета Эратосфена". Оценить суммарное время выполнения для всех процессов и максимальное время выполнения среди всех процессов (без учёта времени ввода/вывода) в зависимости от числа процессов. Использовать MPI.
Формат командной строки: <первое число из диапазона> <последнее число из диапазона> <имя выходного файла для хранения списка простых чисел в текстовом виде через пробелы>

Описание алгоритма

Для нахождения всех простых чисел не больше заданного числа n, следуя методу Эратсфена, нужно выполнить следующие шаги:

1. Выписать подряд все целые числа от двух до n (2, 3, 4, …, n).
2. Пусть переменная p изначально равна двум — первому простому числу.
3. Зачеркнуть в списке числа от 2p до n считая шагами по p (это будут числа кратные p: 2p, 3p, 4p, …).
4. Найти первое незачёркнутое число в списке, большее чем p, и присвоить значению переменной p это число.
5. Повторять шаги 3 и 4, пока возможно.

Идея алгоритма состоит в разбиении поиска на 2 этапа: на 1-ом этапе, который выполняется последовательно, находятся все простые числа в диапазоне $[2, \sqrt{n}]$ с помощью классического метода решета Эратосфена. Найденные простые числа, на 2-ом этапе, позволяют вычеркнуть все составные числа в диапазоне $[\sqrt{n}, n]$. Параллелизация заключается в том, что диапазон $[\sqrt{n}, n]$ разбивается на поддиапазоны, в которых поиск простых чисел может происходить независимо.

Аппаратное обеспечение: Вычисления проводились на узлах вычислительного комплекса IBM Blue Gene/P.

Анализ времени выполнения: Для оценки времени выполнения программы использовалась функция MPI_Wtime().

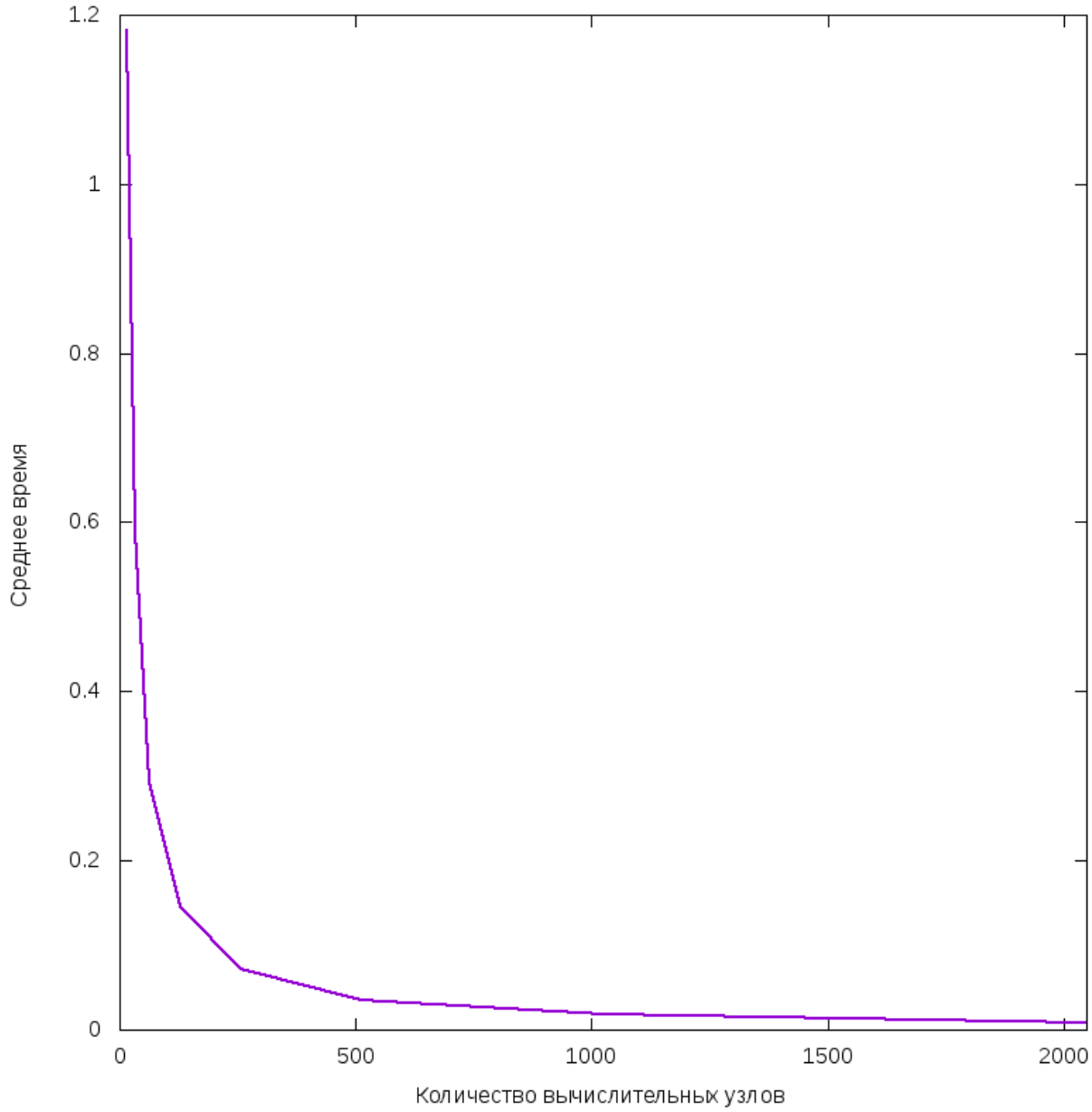
Основные функции:

- **Разбор командной строки.** В рамках функции осуществляется анализ и разбор командной строки.
- **Заполнение сетки.** В рамках функции осуществляется заполнение битовые сетки простых чисел для заданного диапазона.
- **Передача сообщений.** В рамках функции осуществляется передача массивов простых чисел и информации о времени выполнения процессов главному процессу с помощью MPI функций.

Результаты выполнения

Проводился поиск простых чисел на диапазоне от 1 до 10^8. Алгоритм запускался на 16, 32, 64, 128, 256, 512, 1024 и 2048 вычислительных узлах.

Количество вычислительных узлов	**Суммарное время выполнения всех процессов**	**Среднее время выполнения одного процесса**	**Максимальное время выполнения одного процесса**
16	18.8977	1.18111	1.29911
32	18.8534	0.589168	0.628817
64	18.7896	0.293588	0.308889
128	18.6825	0.145957	0.152548
256	18.4299	0.0719917	0.0750366
512	17.7621	0.0346916	0.0363614
1024	18.6406	0.0182037	0.0196016
2048	18.3165	0.0089436	0.0098009



Зависимость среднего времени выполнения процесса (в секундах) от количества вычислительных узлов.

Основные выводы

Исследования показывают, что с увеличением количества вычислительных узлов среднее время выполнения процесса уменьшается, а следовательно, ускоряется вся параллельная программа. Это связано с тем, что диапазон разбивается на большее число частей и каждому процессу необходимо обработать меньший поддиапазон.