Национальный исследовательский ядерный университет «МИФИ»

(Московский Инженерно-Физический Институт)

Кафедра №42 «Криптология и кибербезопасность»

Лабораторная работа №6

«Коллективные операции в MPI»

Тимин Александр Б21-515 (2023г.)

Рабочая среда:

- Процессор: AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz, 8 ядер (16 логических)
- Оперативная память: 16.0 GB DDR4 3200 МГц
- ОС: Windows 10 Pro 22H2 64-bit operating system, x64-based processor
- Среда разработки: Microsoft Visual Studio 2022 (v143)
- Версия OpenMP: 200203 (/openmp:llvm)

Алгоритм

Программа запускается на **P** процессах, на главном процессе (rank = 0) генерируется и заполняется псевдослучайными числами 10 массивов размера **N**. Для каждого массива вызывается функция сортировки Шелла. После выбора параметра **d** (см схему алгоритма в лабораторной работе №3) вызывается функция широковещательного обмена **MPI_Bcast**, которая транслирует на все процессы массив с главного процесса. Каждый процесс выполняет сортировку вставками своей части массива, определенной мегаультрасложным выражением (**k** % **numtasks** == **rank**). Далее функция **MPI_Gather** транслирует массивы со всех процессов на главный, который в свою очередь собирает из них исходный массив, но уже более упорядоченный. В остальном алгоритм идентичен алгоритму из ЛР №3.

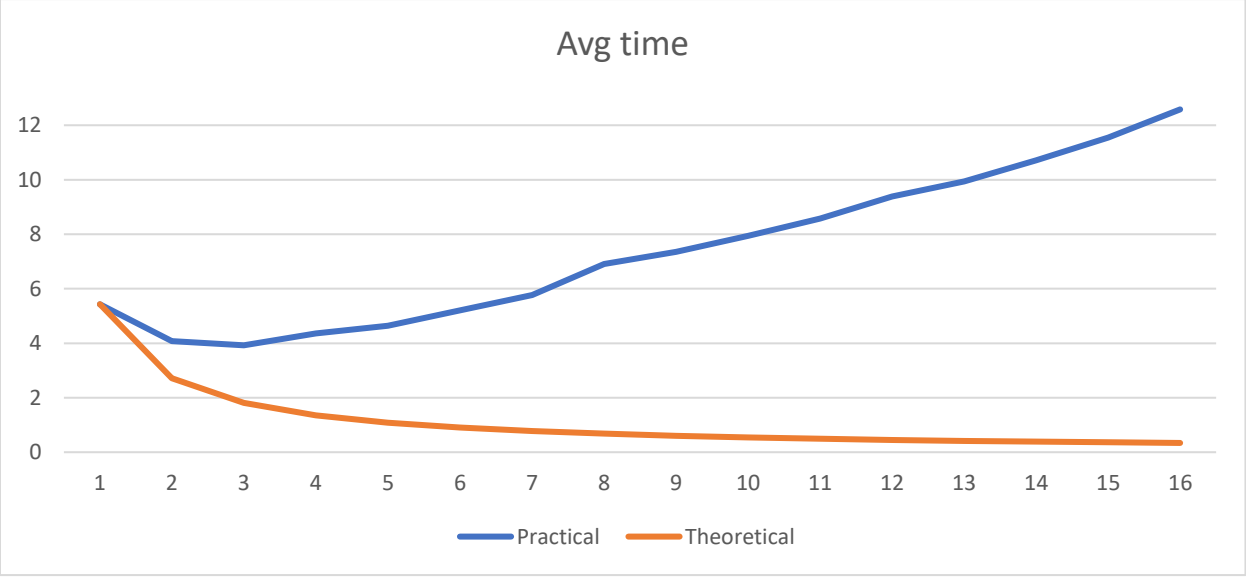
Ход работы

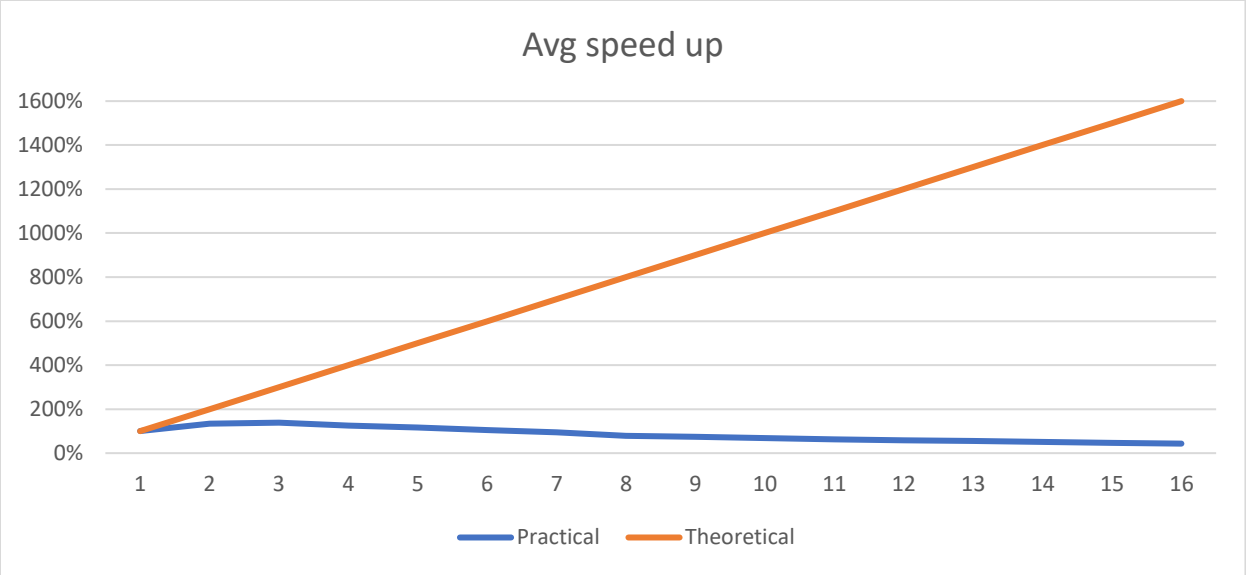
Была модифицирована программа из третьей лабораторной работы таким образом, чтобы использовать возможности библиотеки mpi.h там, где ранее использовалась технология OpenMP для параллельной работы алгоритма. Программа была запущена 16 раз на разном количестве процессов (от 1 до 16), временные характеристики записывались в файл для дальнейшей обработки. На основании продолжительности выполнения алгоритма были вычислены следующие характеристики: среднее время, среднее ускорение и средняя эффективность. По полученным данным были построены соответствующие графики.

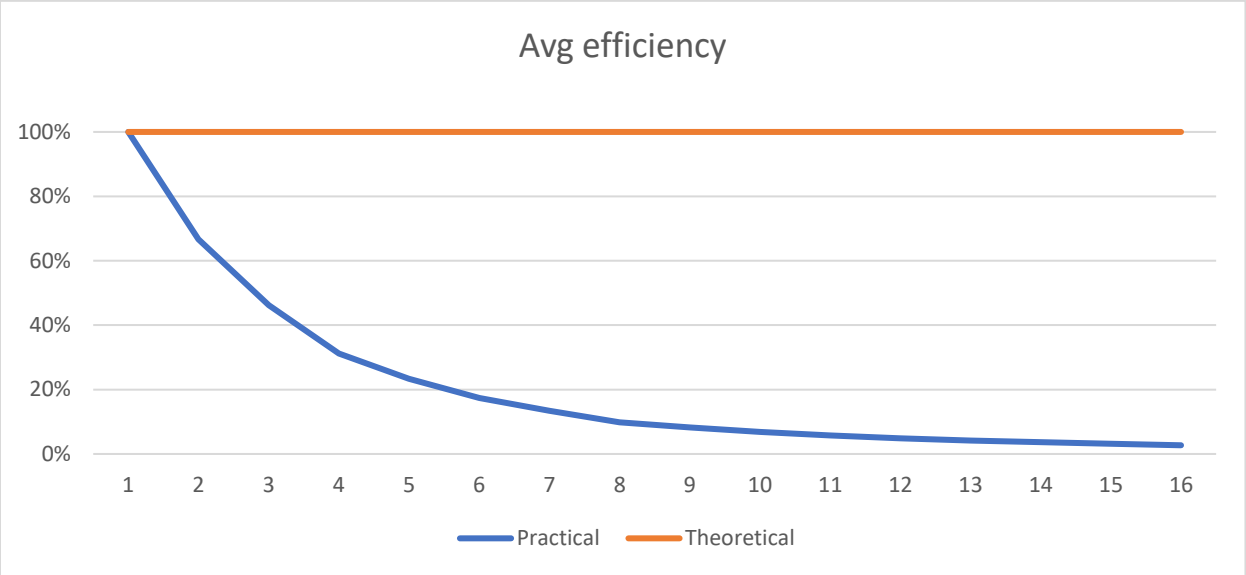
Данные

Procs	Avg time (pr)	Avg time (th)	Avg speed up (pr)	Avg speed up (th)	Avg efficiency (pr)	Avg efficiency (th)
1	5.432913	5.432913	100.00%	100.00%	100.00%	100.00%
2	4.074560	2.716457	133.34%	200.00%	66.67%	100.00%
3	3.923283	1.810971	138.48%	300.00%	46.16%	100.00%
4	4.358439	1.358228	124.65%	400.00%	31.16%	100.00%
5	4.645787	1.086583	116.94%	500.00%	23.39%	100.00%
6	5.201840	0.905486	104.44%	600.00%	17.41%	100.00%
7	5.768329	0.776130	94.19%	700.00%	13.46%	100.00%
8	6.913535	0.679114	78.58%	800.00%	9.82%	100.00%
9	7.351428	0.603657	73.90%	900.00%	8.21%	100.00%
10	7.938674	0.543291	68.44%	1000.00%	6.84%	100.00%
11	8.579089	0.493901	63.33%	1100.00%	5.76%	100.00%
12	9.381282	0.452743	57.91%	1200.00%	4.83%	100.00%
13	9.942418	0.417916	54.64%	1300.00%	4.20%	100.00%
14	10.710286	0.388065	50.73%	1400.00%	3.62%	100.00%
15	11.542122	0.362194	47.07%	1500.00%	3.14%	100.00%
16	12.581007	0.339557	43.18%	1600.00%	2.70%	100.00%

Графики







Заключение

За счет распараллеливания алгоритма при помощи технологии MPI удалось сократить время его исполнения по сравнению с последовательной версией вплоть до запуска на 6 процессах, при этом уже на 4 процессах ускорение пошло на спад (далее идет сильно выраженный линейный рост времени исполнения). В сравнении с версией программы с использованием технологии OpenMP, текущая версия программы уступает по всем параметрам из-за необходимости передавать большие массивы данных между процессами (чего не требовалось при использовании OpenMP).

Приложение

- исходный код программы;
- заголовочный файл;
- «сырые» данные (текст);
- «обработанные» данные (таблица и графики);
- отчет (docx);
- отчет (pdf);
- отчет по лабораторной работе №3.