МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу

«Суперкомпьютеры и параллельная обработка данных»

Разработка параллельной версии программы для сортировки данных

Вариант 2: MPI

ОТЧЕТ

о выполненном задании

студентки 328 учебной группы факультета ВМК МГУ
Богатенковой Анастасии Олеговны

гор. Москва
2019 г.

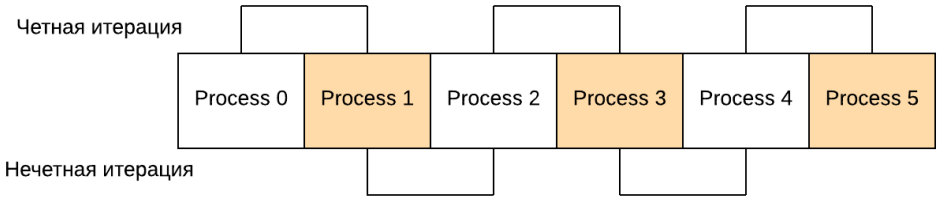
Содержание

1 Постановка задачи

Разработать параллельную версию программы для сортировки данных с использованием технологии MPI, а затем исследовать масштабируемость полученной программы, построить графики зависимости времени её выполнения от числа используемых ядер и объёма входных данных.

2 Алгоритм

Выбран алгоритм сортировки "пузырьком". Реализована схема, напоминающая чет-нечетную сортировку. Массив, который необходимо отсортировать, разбивается на части равного размера (в реализованной программе предполагается, что число процессов делит размер массива нацело). Каждая часть посылается отдельному процессу для сортировки с помощью функции MPI_Scatter(). Процессы сортируют свою часть с помощью обычной сортировки пузырьком, затем происходит обмен данными между процессами-соседями.



Каждый из процессов выполняет tasks итераций обмена с соседями, где tasks - число MPI процессов. Обмен данными, то есть слияние двух отсортированных массивов в один отсортированный происходит с помощью блокирующих операций MPI_Send(), MPI_Recv(). На четной итерации, как показано на рисунке, процессы с четными номерами обмениваются массивами с соседями справа, а процессы с нечетными номерами - с соседями слева. 2 массива объединяются в один, который сортируется и снова распределяется между двумя процессами. На нечетной итерации происходят аналогичные действия, только процессы с четными и нечетными номерами меняются местами. В итоге, после tasks итераций, получается отсортированный массив, который собирается в главном процессе с помощью функции MPI_Gather().

3 Результаты замеров времени выполнения

Ниже приведены результаты замеров времени программ на суперкомпьютерах Bluegene и Polus: непосредственно в табличной форме и наглядно на 3D-графиках. Программа была запущена в конфигурациях:

- на Bluegene - 1, 2, 4, 8, 16, 32, 64, 128, 256 ядер;
- на Polus - 1, 2, 4, 8, 16, 32 ядер.

Также для сравнения MPI-версия программы была запущена на ноутбуке (OS: Linux Mint 19.1 Tessa x86 64; Kernel: 4.15.0-43-generic; CPU: Intel Celeron N2920 (4) @ 1.999GHz; Memory: 2430MB / 3827MB)

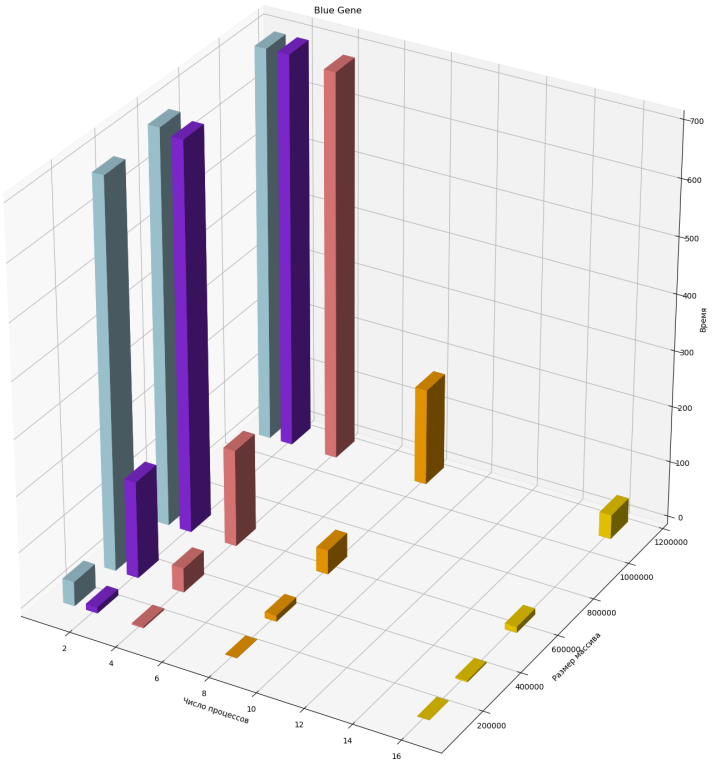
Каждая конфигурация была запущена 5 раз. Ниже приведены усредненные результаты.

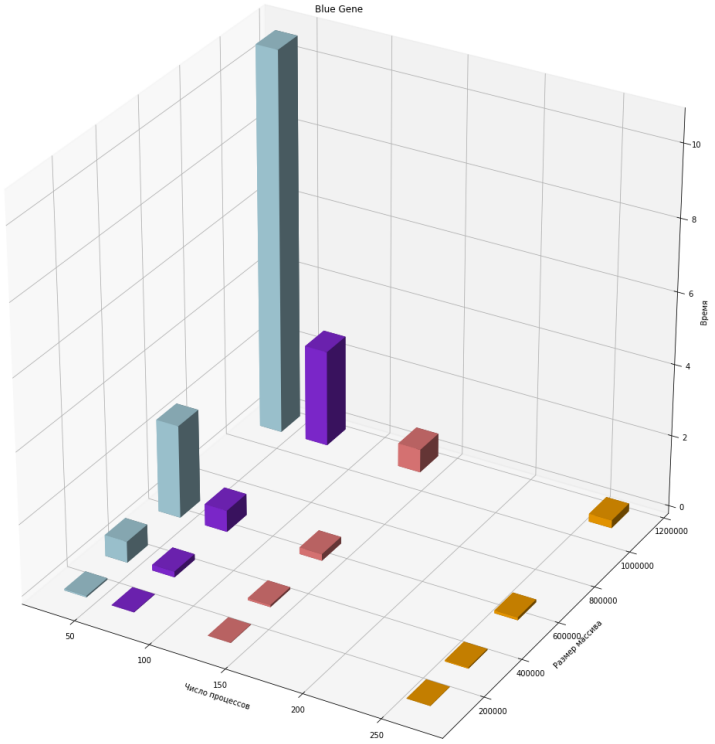
Blue Gene	компиляция: mpixlc_r mpi.c -o mpi -O2 запуск: mpisubmit.bg -n число_ядер mpi -- размер массива								
размер массива \ число процессов	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
1024	0.009	0.002	0.0008	0.0003	0.0003				
8192	0.577	0.145	0.037	0.0099	0.003				
32768	10.7	2.6	0.58	0.14	0.03				
65536	43.1	10.7	2.6	0.5	0.15	0.04	0.01	0.009	0.006
262144	689.2	172.4	43.1	10.7	2.6	0.59	0.16	0.057	0.032
524288	> 15 мин	690.06	172.5	43.1	10.7	2.6	0.6	0.184	0.078
1048576	> 15 мин	> 15 мин	690.09	172.5	43.1	10.7	2.69	0.65	0.22

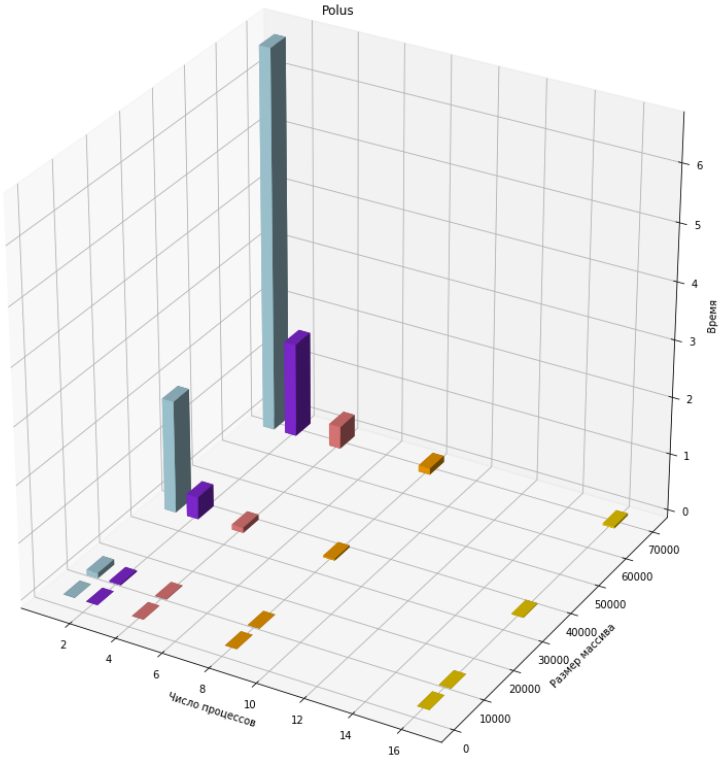
Polus	module load SpectrumMPI; компиляция: mpixlc mpi.c -o mpi -O2 запуск: mpisubmit.pl -p число_процессов mpi -- размер_массива						
размер массива \ число процессов	**1**	**2**	**4**	**8**	**16**	**32**	
1024	0.000005	0.000026	0.000252	0.0005	0.000005		
8192	0.09	0.02	0.006	0.002	0.0009		
32768	1.97	0.39	0.09	0.03	0.008		
65536	6.7	1.65	0.39	0.11	0.03	0.01	
262144	110.9	28.05	6.9	1.8	0.45	0.2	
524288	446.1	113.1	29.3	7.5	1.76	0.5	
1048576	> 15 мин	454.35	116.09	31.6	7.45	1.8	

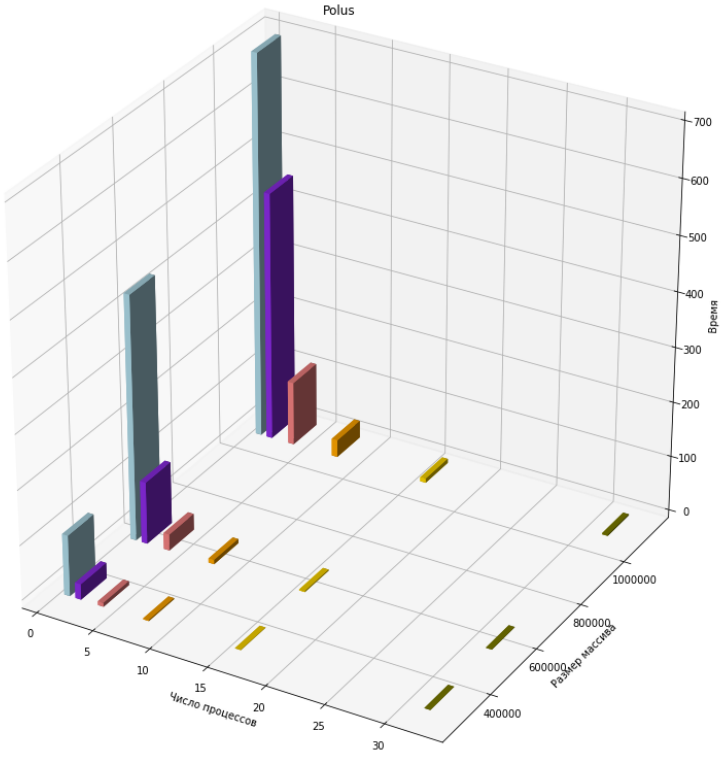
Laptop	компиляция: mpicc mpi.c -o mpi -std=c11 -O2 запуск: mpirun -n число_процессов ./mpi размер_массива		
размер массива \ число процессов	**1**	**2**	**4**
1024	0.003	0.0007	0.0007
8192	0.16	0.03	0.01
32768	2.6	0.65	0.17
65536	10.4	2.6	0.7
262144	169.06	42.04	12.18
524288	687.4	171.2	48.4

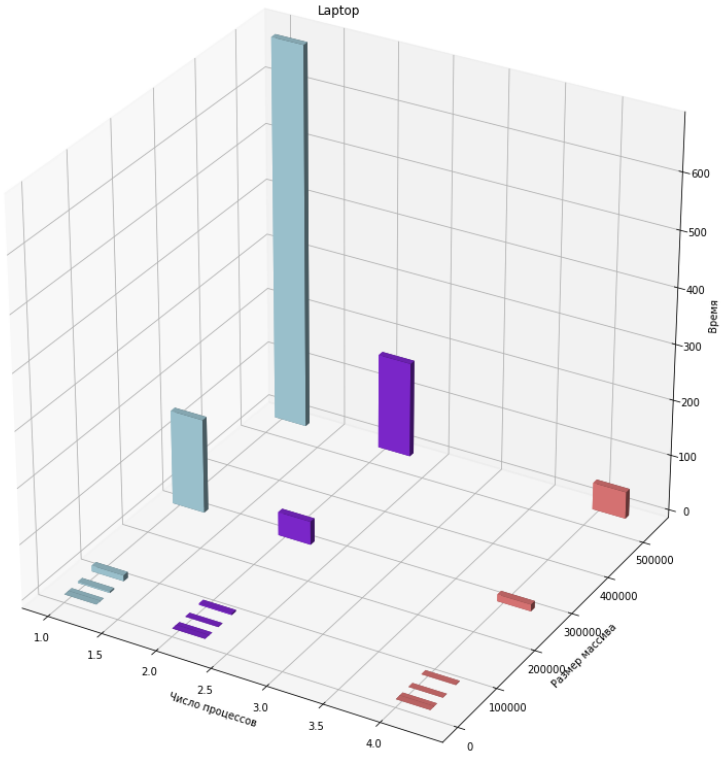
3.1 3-D Графики




Blue Gene
Время
0
2
4
6
8
10
Число процессов
50
100
150
200
250
Размер массива
200000
400000
600000
800000
1000000
1200000


Polus
Время
0
1
2
3
4
5
6
Размер массива
0
10000
20000
30000
40000
50000
60000
70000
Число процессов
2
4
6
8
10
12
14
16


Polus
Время
700
600
500
400
300
200
100
0
Размер массива
1000000
800000
600000
400000
Число процессов
0
5
10
15
20
25
30



4 Выводы

Из данных, приведенных в таблицах, можно заключить, что распараллеливание программы дало значительный выигрыш по времени, особенно на массивах большого размера. Как показали эксперименты, на массивах небольшого размера и при небольшом числе нитей Polus выигрывает у всех машин, однако при увеличении числа ядер, очень хорошие результаты показывает Blue Gene. Ухудшения производительности не наблюдалось, эксперименты было решено прекратить в силу того, что время работы алгоритма перестало сильно изменяться. На моем ноутбуке при увеличении числа MPI процессов наблюдалось

ухудшение производительности, так как число ядер - 4.

В результате работы была получена параллельная версия алгоритма сортировки, использующая алгоритм чет-нечетной сортировки и технологию MPI. Программа была протестирована на нескольких устройствах, а полученные результаты проанализированы.