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

Спецкурс: системы и средства параллельного программирования.

Отчёт № 4.
Алгоритм умножения матрицы на вектор с использованием технологии MPI.

Работу выполнил
Васильев С.М.

Москва 2018

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм умножения матрицы на вектор с использованием технологии MPI. Исследовать производительность программмы.
Формат командной строки: <имя файла матрицы A> <имя файла матрицы b> <имя файла матрицы c> <режим, порядок индексов>.
Формат файла-матрицы: Матрица представляются в виде бинарного файла следующего формата:

Тип	Значение	Описание
Число типа char	T – f (float) или d (double)	Тип элементов
Число типа size_t	N – натуральное число	Число строк матрицы
Число типа size_t	M – натуральное число	Число столбцов матрицы
Массив чисел типа T	NxM элементов	Массив элементов матрицы

Элементы матрицы хранятся построчно.

Формат файла-вектора: файл матрица размером Nx1.

Описание алгоритма.

Если $N \geq M$, элементы матрицы равномерно распределяются по процессам блоками строк, если $N < M$ – блоками столбцов. Каждый процесс перемножает свой участок на вектор b. Вектор c находится как сумма векторов, полученных каждым процессом.

Программа запускалась на IBM Blue Gene/P.

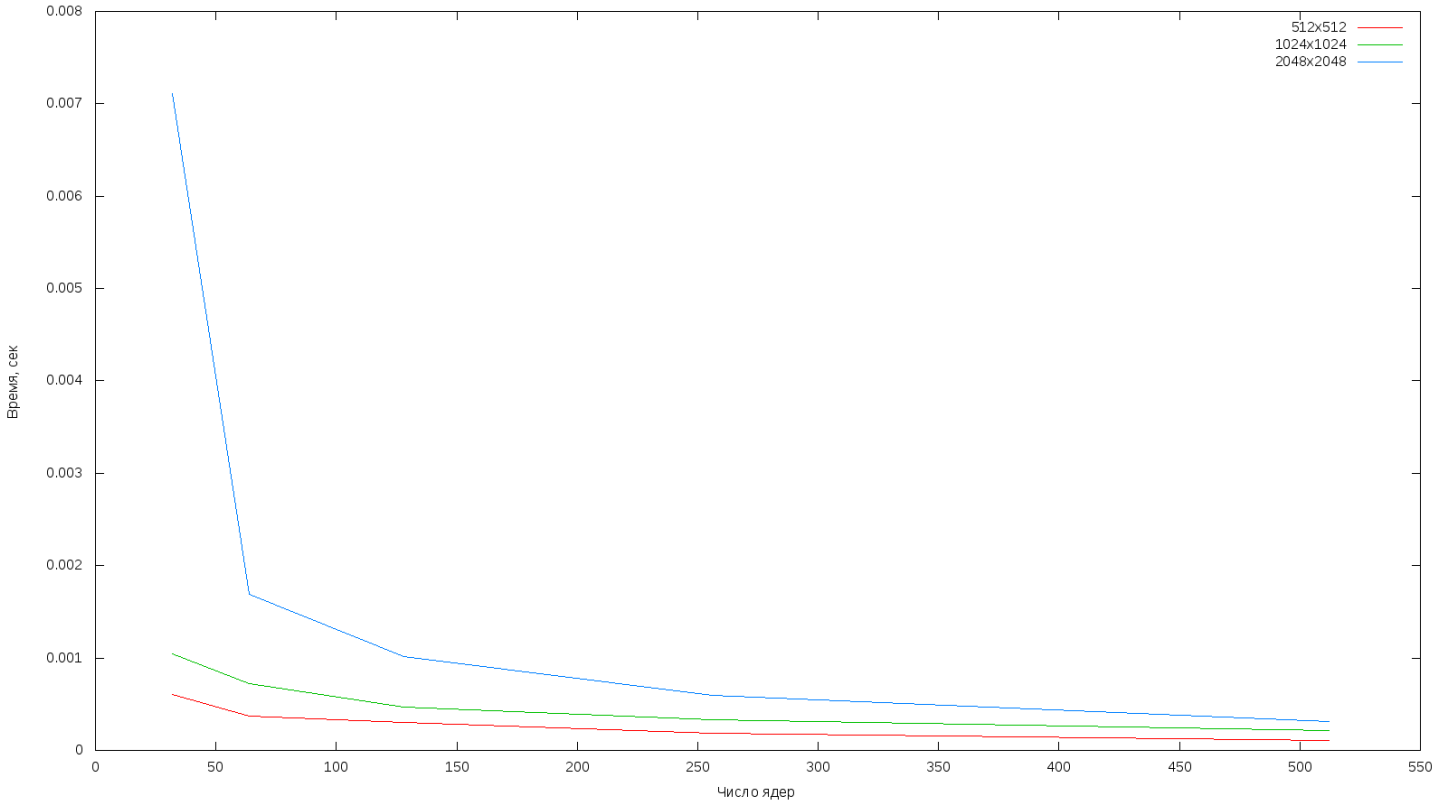
Замер времени производился функцией MPI_Wtime().

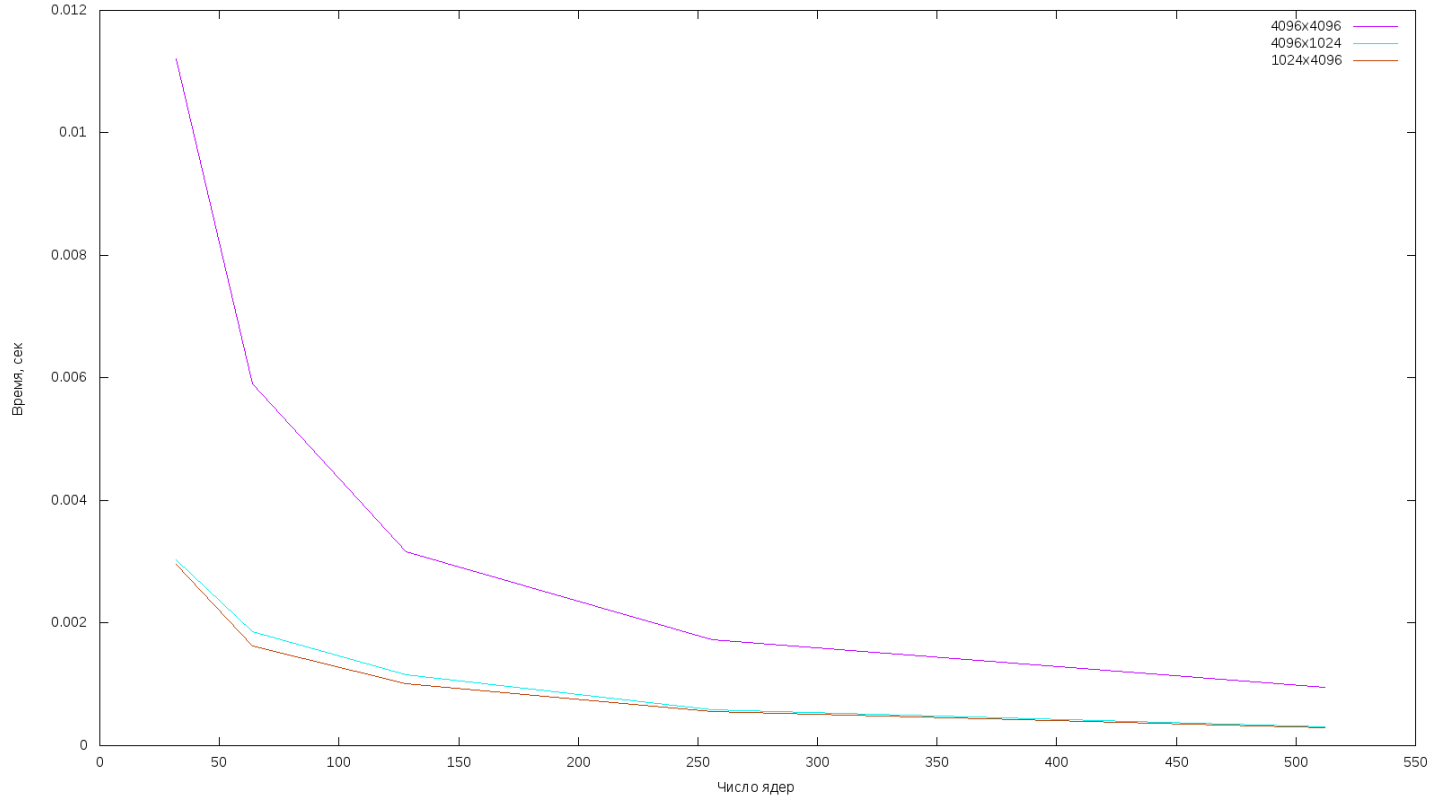
Результаты выполнения.

Проводились умножения матриц 512 × 512, 1024 × 1024, 2048 × 2048, 4096 × 4096, 4096 × 1024, 1024 × 4096. Алгоритм запускался на 32, 64, 128, 256 и 512 ядрах. Альтернативный вариант мэппинга для 512 ядер генерировался случайно.

Время выполнения.

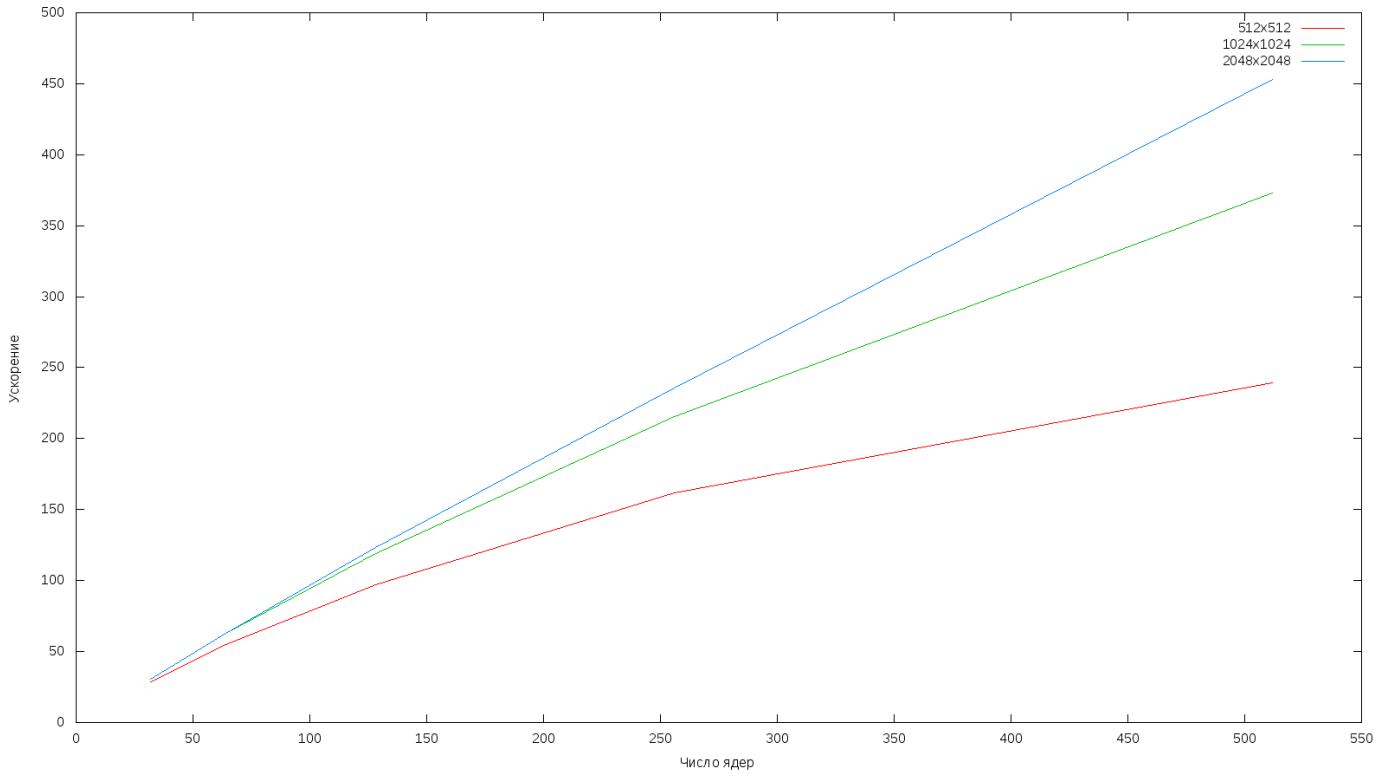
N	M	32	64	128	256	512	512 map
512	512	0.000606134	0.000373499	0.000298834	0.000189511	0.000103615	0.000103427
1024	1024	0.001044071	0.000725734	0.000468901	0.000329431	0.000217492	0.000214716
2048	2048	0.00711341	0.00169129	0.00101812	0.000592362	0.000311457	0.000311453
4096	4096	0.0112133	0.00590993	0.00316391	0.00172847	0.00094612	0.00094341
4096	1024	0.00303454	0.00186805	0.00116322	0.00059172	0.00031495	0.00031128
1024	4096	0.00296438	0.00162902	0.00101318	0.00055292	0.00029178	0.00028834

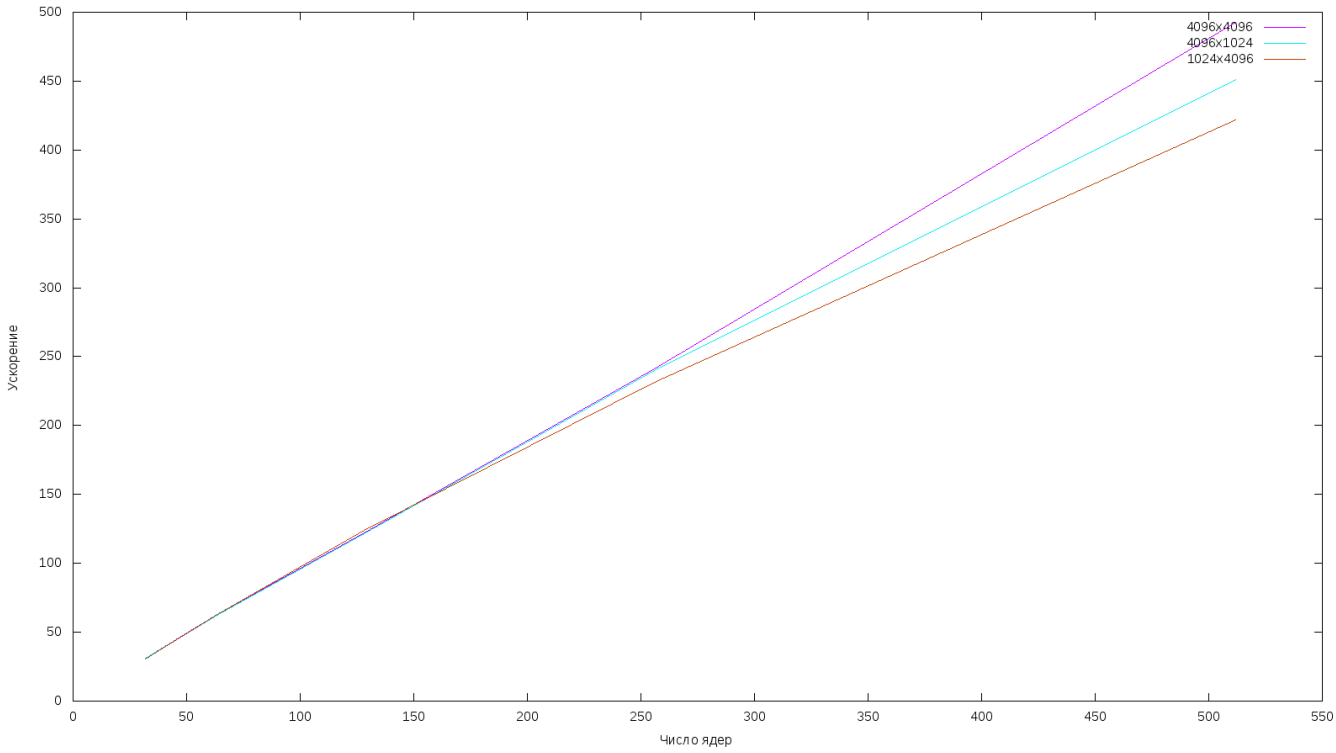

512x512
1024x1024
2048x2048
Время, сек
Число ядер
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
50
100
150
200
250
300
350
400
450
500
550


4096x4096
4096x1024
1024x4096
Время, сек
Число ядер
0.012
0.01
0.008
0.006
0.004
0.002
0
0
50
100
150
200
250
300
350
400
450
500
550

Ускорение.

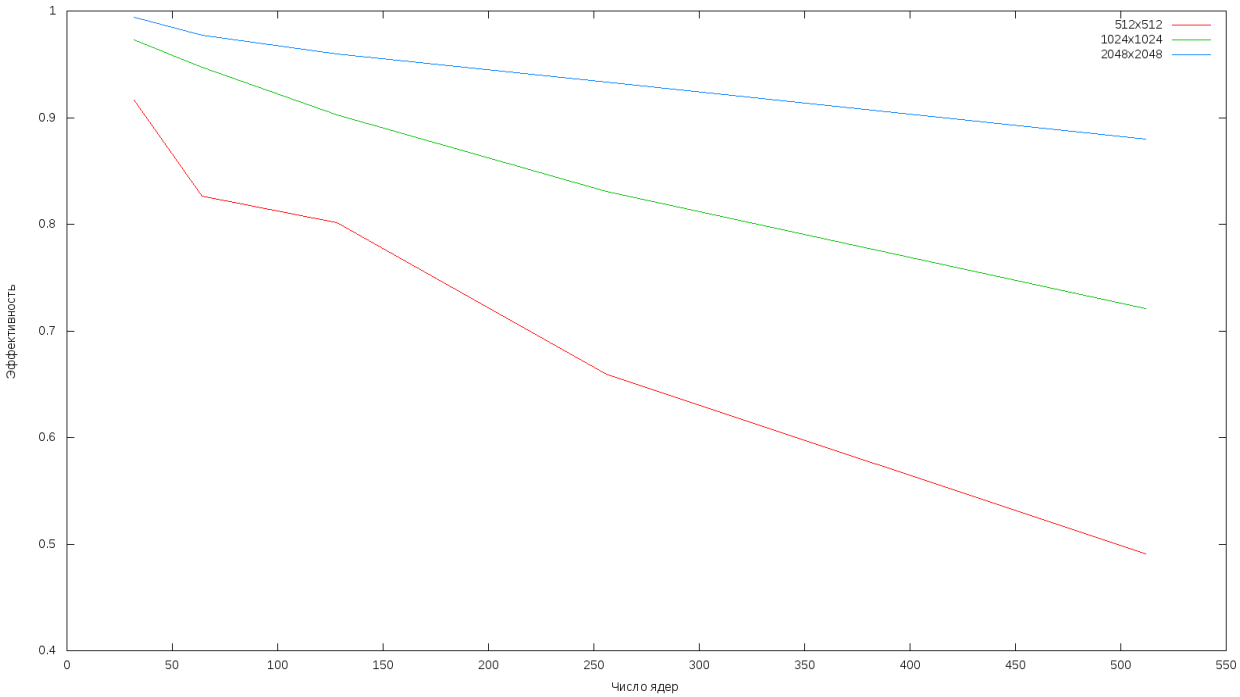
N	M	32	64	128	256	512	512 map
512	512	28.5263	55.2316	96.9975	161.783	239.394	240.514
1024	1024	30.283	62.745	119.129	215.34	372.941	371.441
2048	2048	30.502	63.1018	123.603	235.44	453.274	453.827
4096	4096	30.834	63.61	122.002	241.181	492.781	493.03
4096	1024	31.109	62.912	121.774	239.652	451.078	451.396
1024	4096	30.647	63.371	123.804	231.485	421.649	422.2

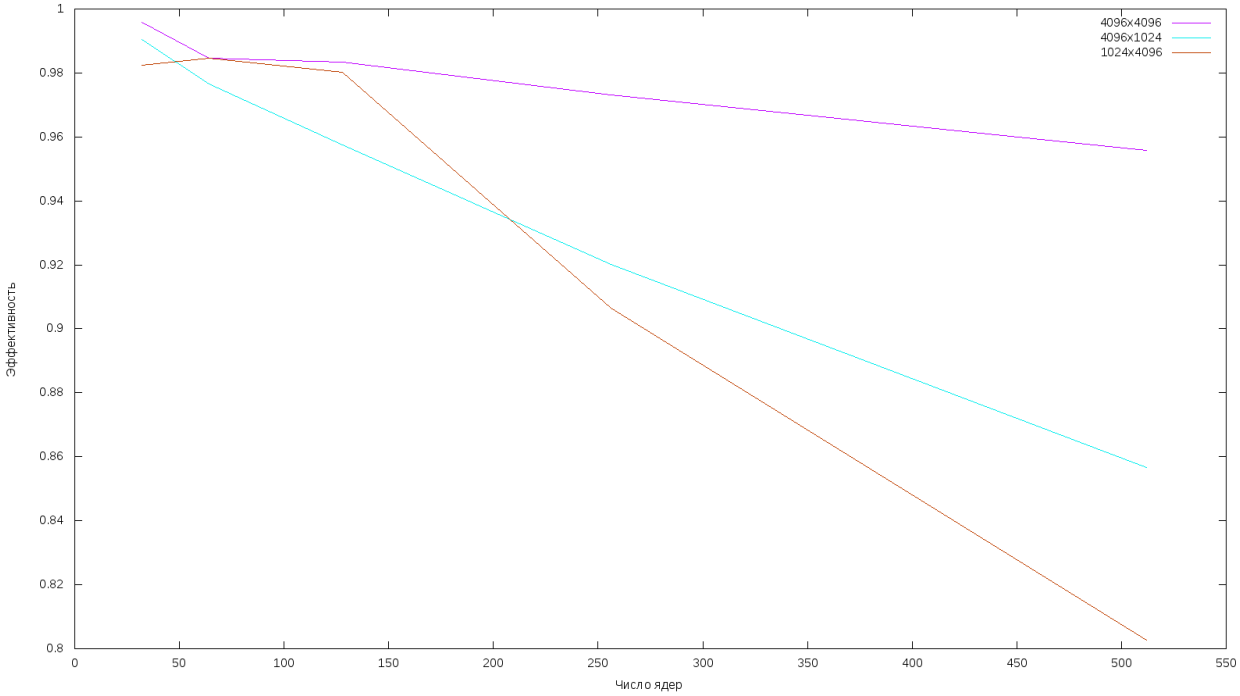




Эффективность.

N	M	32	64	128	256	512	512 map
512	512	0.916384	0.826251	0.801634	0.65957	0.4911	0.49263
1024	1024	0.972875	0.947197	0.902242	0.830859	0.72117	0.720709
2048	2048	0.994419	0.977009	0.959937	0.933504	0.87951	0.879754
4096	4096	0.995928	0.984502	0.983414	0.97316	0.955803	0.95582
4096	1024	0.990494	0.976516	0.957438	0.92025	0.856633	0.856906
1024	4096	0.98242	0.98469	0.980109	0.906398	0.802674	0.802922





Выводы.

С увеличением количества ядер время выполнения программы уменьшается, а следовательно, ускоряется вся параллельная программа. Это связано с тем, что каждому процессу необходимо произвести меньше вычислений. Однако эффективность распараллеливания постепенно уменьшается. Наибольшее ускорение и наилучшая эффективность наблюдается при распределении элементов матрицы по процессам блоками строк (при $N \geq M$). Мэппинг данной параллельной программы не влияет на время её работы, т. к. производится мало пересылок.