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
Параллельный алгоритм умножения матрицы на вектор

Разработка параллельной MPI программы и исследование ее эффективности

Работу выполнил
Чепурнов А. В.

Москва 2018

Постановка задачи и формат данных

Задача: Разработать параллельную программу с использованием технологии MPI, реализующую алгоритм умножения плотной матрицы на вектор $Ab = c$. Провести исследование эффективности разработанной программы на системе Blue Gene/P.
Формат командной строки: <имя файла матрицы A> <имя файла вектора b> <имя файла вектора c>
Формат файла-матрицы: Матрица представляется в виде бинарного файла следующего формата:

Тип	Значение	Описание
Число типа char	'd' – тип double	Тип элементов матрицы
Число типа int	N – натуральное число	Число строк матрицы
Число типа int	M – натуральное число	Число столбцов матрицы
Массив чисел типа double	$N \times M$ элементов	Массив элементов матрицы

Формат файла-вектора: Вектор представляется как матрица размера $N \times 1$.

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.

Аппаратное обеспечение: Исследования проводились на вычислительном комплексе IBM Blue Gene/P.

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

Анализ ускорения: Ускорение, получаемое при использовании параллельного алгоритма для p процессоров, высчитывалось как отношение времени выполнения задачи на одном процессоре к времени параллельного выполнения задачи при использовании p процессоров.

Анализ эффективности: Эффективность параллельного алгоритма при использовании p процессоров, высчитывалась как отношение ускорения к числу процессоров.

Основные функции:

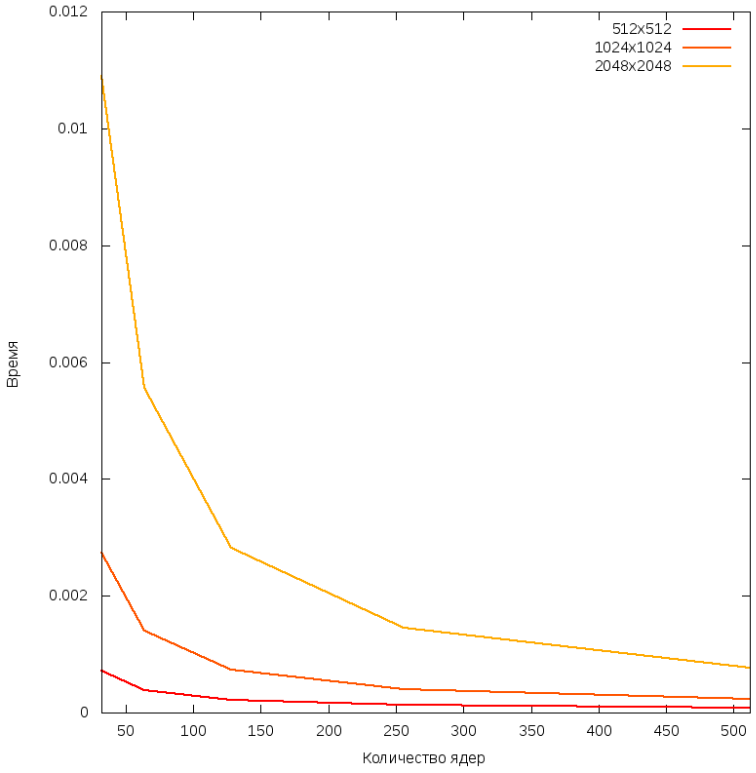
- **Разбор командной строки.** В рамках функции осуществляется анализ и разбор командной строки.
- **Параллельное чтение из файла.** В рамках функции осуществляется параллельное чтение элементов матрицы из файла с помощью средств MPI, а также анализ совместимости матрицы и вектора.
- **Умножение матрицы на вектор.** В рамках функции осуществляется умножение матрицы на вектор в зависимости от соотношения N и M.

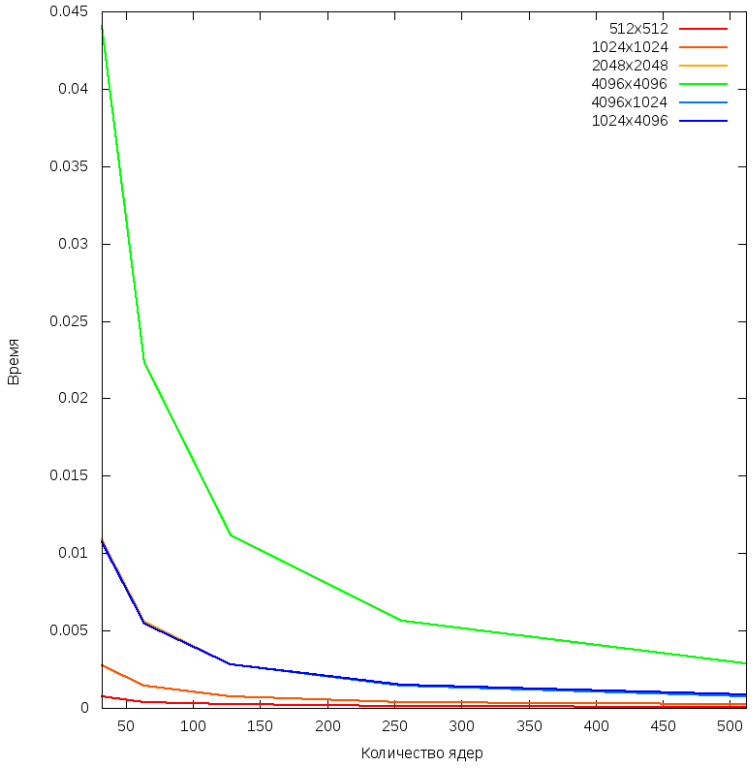
Результаты выполнения

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.

Среднее время выполнения процесса (в секундах):

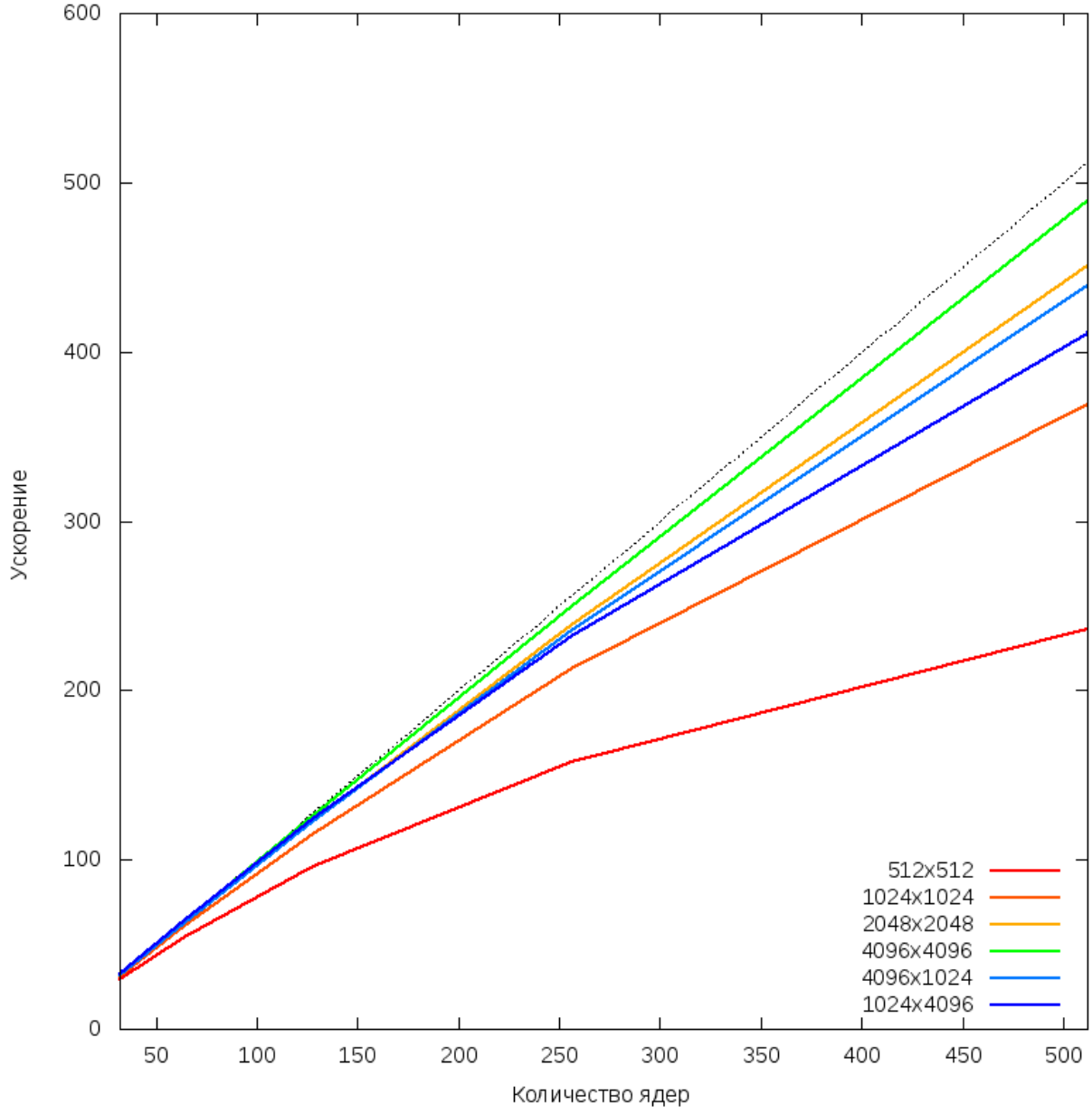
n	m	32	64	128	256	512	512 (мэппинг)
512	512	0.000722441	0.000390714	0.000220707	0.000134472	8.98388e-05	8.97071e-05
1024	1024	0.00274403	0.00140921	0.000739714	0.000401632	0.00023136	0.000231508
2048	2048	0.0109061	0.00555021	0.00282446	0.00145222	0.000770687	0.000770472
4096	4096	0.0440562	0.0222838	0.0111542	0.00563587	0.00286911	0.00286906
4096	1024	0.0107801	0.00546721	0.00278807	0.00145037	0.000779041	0.000778791
1024	4096	0.0107294	0.00545935	0.00282699	0.00151285	0.000854172	0.000853908





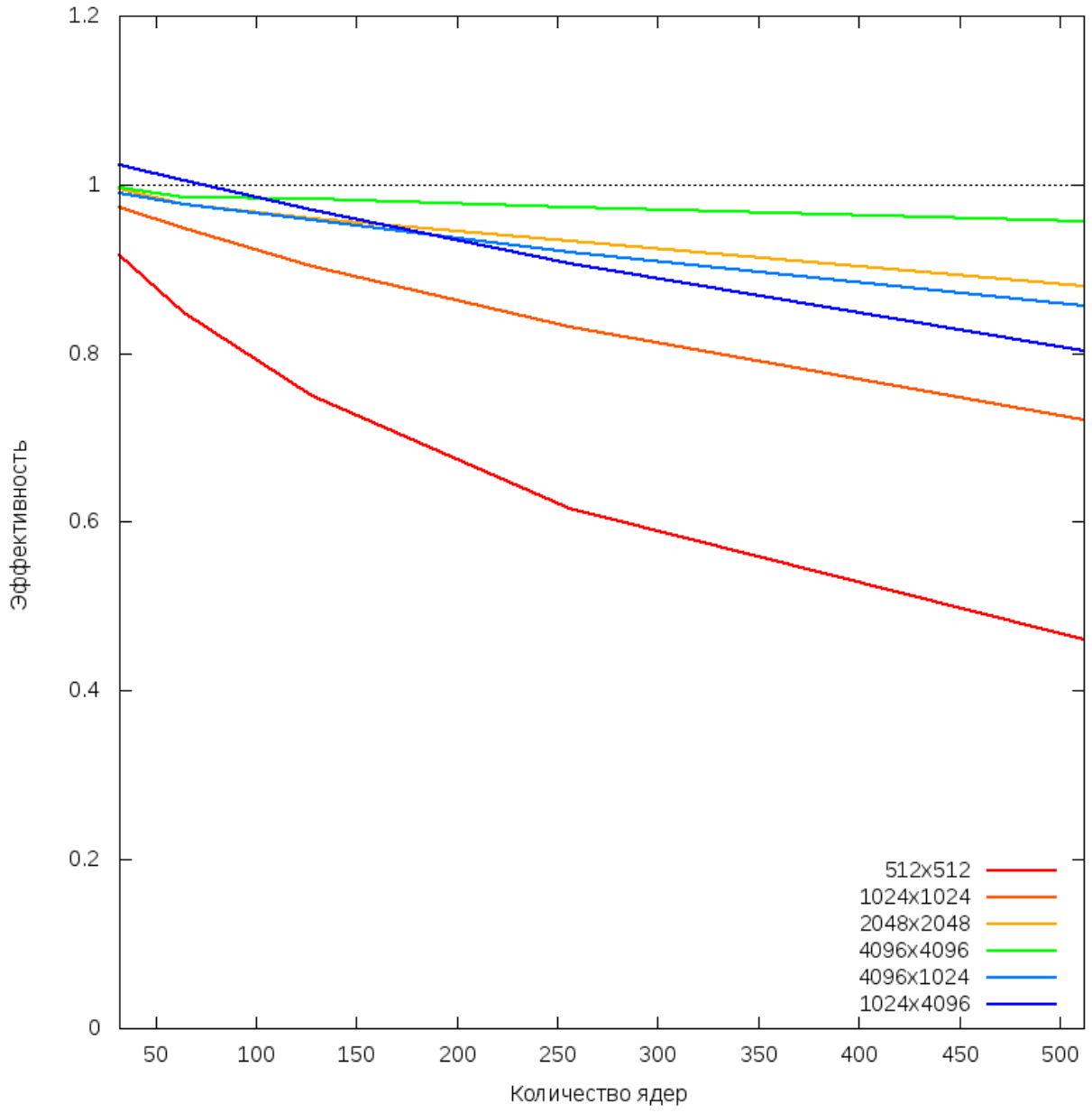
Ускорение работы алгоритма:

n	m	32	64	128	256	512	512 (мэппинг)
512	512	29.3243	54.2215	95.9874	157.543	235.812	236.159
1024	1024	31.132	60.6206	115.487	212.7	369.239	369.003
2048	2048	31.8214	62.5286	122.872	238.977	450.309	450.434
4096	4096	31.8697	63.0081	125.877	249.129	489.371	489.38
4096	1024	31.6958	62.497	122.552	235.584	438.596	438.736
1024	4096	32.7174	64.3003	124.174	232.038	410.969	411.096



Эффективность работы алгоритма:

n	m	32	64	128	256	512	512 (мэппинг)
512	512	0.916384	0.847211	0.749902	0.615402	0.46057	0.461248
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	1.02242	1.00469	0.970109	0.906398	0.802674	0.802922



Основные выводы

Исследования показывают, что с увеличением количества ядер среднее время выполнения процесса 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.