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.

Анализ параллельной оптимизации умножения матрицы на вектор

Работу выполнил

Федоров В. В.

Москва 2020

Постановка задачи и формат данных.

Задача: Реализовать две версии параллельного умножения матрицы на вектор – с разбиением матрицы по строкам (rowwise) и по столбцам (columnwise), выполнить замеры времени выполнения программы, а также подсчитать ускорение и эффективность.

Формат командной строки: <файл с матрицей A> <файл с вектором b> <файл для записи вектора c> <версия умножения (r — rowwise/c — columnwise)> <файл для вывода времени выполнения>

Формат файла-матрицы: Матрица представляется в виде бинарного файла следующего формата:

Тип	Значение	Описание
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

Элементы матрицы хранятся построчно.

Формат файла-вектора: Вектор представляется в виде бинарного файла следующего формата:

Тип	Значение	Описание
Число типа int	N – натуральное число	Число строк матрицы
Массив чисел типа double	N элементов	Массив элементов матрицы

Элементы матрицы хранятся построчно.

Описание алгоритма.

Математическая постановка: Результатом умножения матрицы $A \in \mathbb{R}^{n \times m}$ на вектор $b \in \mathbb{R}^m$ является вектор $c \in \mathbb{R}^n$, элементы которого вычисляются по формуле:

$$c_i = \sum_{j=1}^{m} A_{ij} b_j; i = 1..n$$

Существует 2 версии параллельной оптимизации данного алгоритма.

- Разбиение по строкам (rowwise): Пусть программа запускается на p процессах. Каждому процессу подается блок строк матрица A. Размер этого блока n_s у процесса s определяется по формуле:

  $$n_s = \frac{n}{p} + (s < n (mod\ p))$$

  Например, при n = 13 и p = 5 первым трем процессам будет отдано 3 строки, а последним двум — 2. Кроме строк матрицы A, каждому процессу подается копия вектора b. После умножения процессом $A_s \in \mathbb{R}^{n_s \times m}$ - выделенной ему части матрицы A — на вектор $b \in \mathbb{R}^m$ он получит вектор $c_s \in \mathbb{R}^{n_s}$, для получения конечного ответа нужно последовательно соединить полученные каждым процессом векторы в один.

- Разбиение по столбцам (rowwise): Каждому процессу подается блок столбцов матрица A. Размер этого блока m_s у процесса s определяется по формуле:

  $$m_s = \frac{m}{p} + (s < m (mod\ p))$$

  Кроме строк матрицы A, каждому процессу часть вектора b размером m_s. После

умножения процессом $A_s \in \mathbb{R}^{n \times m_s}$ - выделенной ему части матрицы A — на вектор $b_s \in \mathbb{R}^{m_s}$ он получит вектор $c_s \in \mathbb{R}^n$, для получения конечного ответа нужно просуммировать все полученные векторы в один.

Анализ данных программы: Для оценки времени работы программы использовалась функция MPI_Wtime.

Верификация: Для проверки корректности работы программы она тестировалась на 20 заранее сгенерированных тестах.

Результаты выполнения.

Результаты:

Замеры времени проводились на системе BlueGene/P. Использовались матрицы следующих размеров: 512×512; 1024×1024; 2048×2048; 4096×4096; 1024×4096; 4096×1024 . Замеры проводились на следующем числе процессов: 32; 64; 128; 256; 512. В случае 512 процессов замеры проводились на двух видах мэппинга (размещения процессов в топологии BlueGene/P — трехмерном торе) — стандартном и случайно сгенерированном. Каждый замер проводился как для rowwise-версии программы, так и для columnwise.

Время выполнения, с - rowwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	0,003532	0,003114	0,003088	0,003532	0,005823	0,006187
1024	1024	0,013080	0,011385	0,010741	0,010796	0,013387	0,012123
2048	2048	0,051510	0,044269	0,041116	0,039934	0,042013	0,038717
4096	4096	0,205932	0,175934	0,162584	0,156307	0,154479	0,146064
4096	1024	0,051341	0,044185	0,041067	0,041067	0,041655	0,038770
1024	4096	0,051718	0,044340	0,041247	0,039955	0,041605	0,038875

Ускорение - rowwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	1,000000	1,134232	1,143782	1,000000	0,606560	0,570874
1024	1024	1,000000	1,148880	1,217764	1,211560	0,977067	1,078941
2048	2048	1,000000	1,163568	1,252797	1,289878	1,226049	1,330423
4096	4096	1,000000	1,170507	1,266619	1,317484	1,333074	1,409875
4096	1024	1,000000	1,161955	1,250177	1,250177	1,232529	1,324246
1024	4096	1,000000	1,166396	1,253861	1,294406	1,243072	1,330367

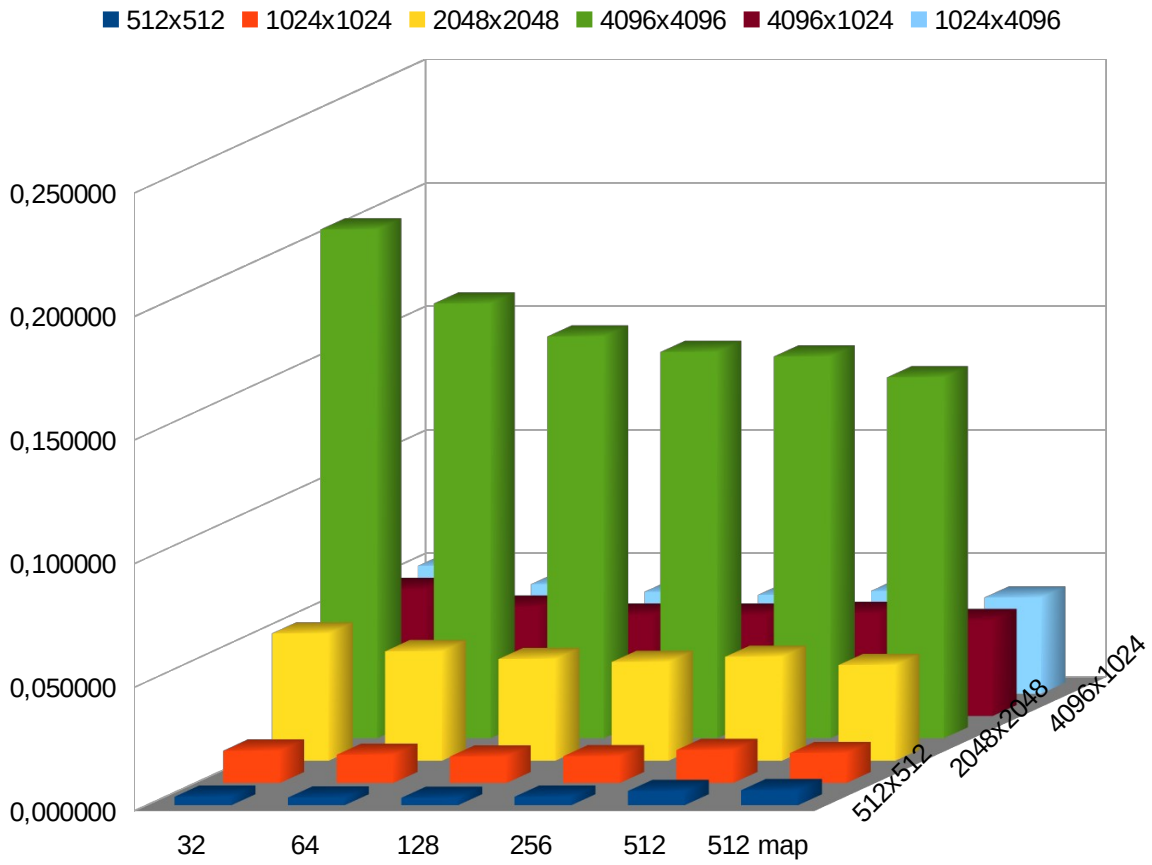
Эффективность - rowwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	1,000000	0,567116	0,285946	0,125000	0,037910	0,035680
1024	1024	1,000000	0,574440	0,304441	0,151445	0,061067	0,067434
2048	2048	1,000000	0,581784	0,313199	0,161235	0,076628	0,083151
4096	4096	1,000000	0,585254	0,316655	0,164686	0,083317	0,088117
4096	1024	1,000000	0,580978	0,312544	0,156272	0,077033	0,082765
1024	4096	1,000000	0,583198	0,313465	0,161801	0,077692	0,083148

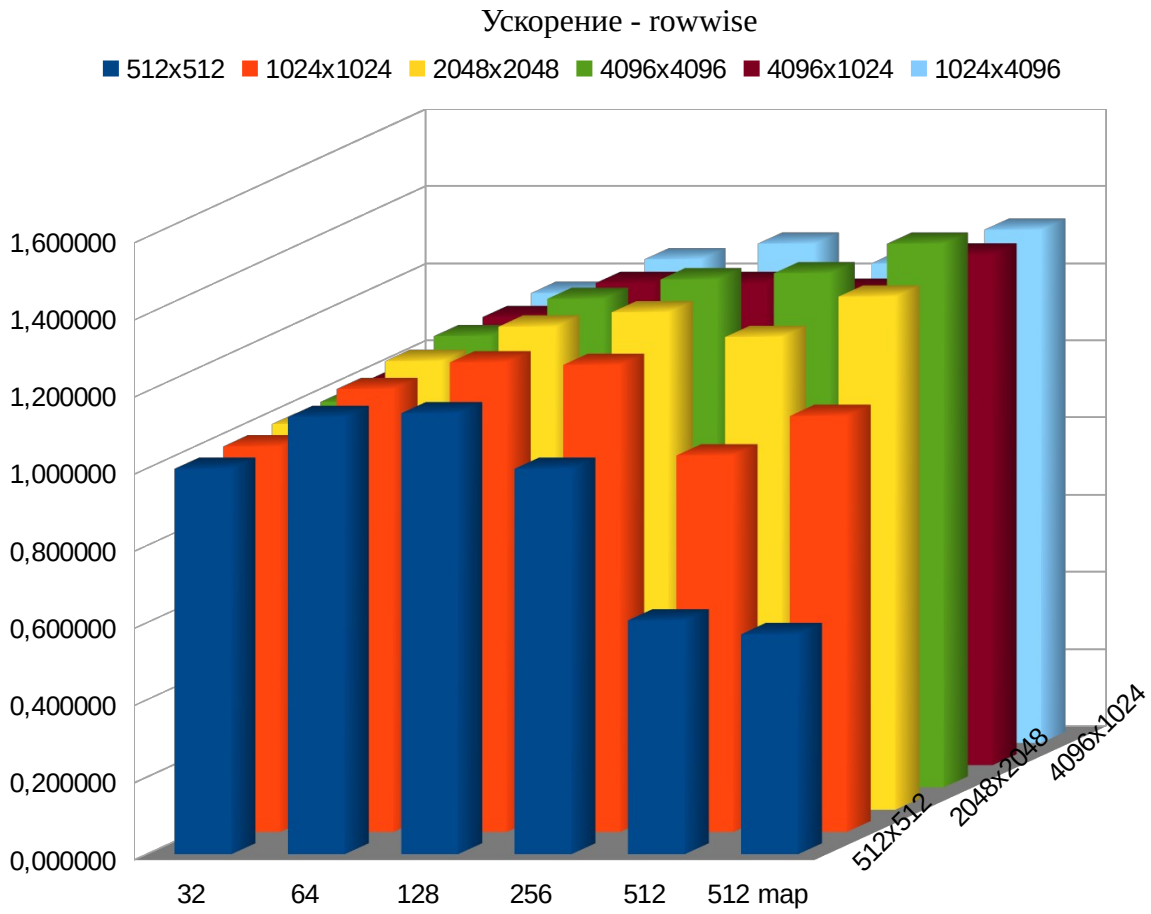
Время выполнения, с - columnwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	0,003446	0,003022	0,002855	0,002899	0,003143	0,002525
1024	1024	0,013049	0,011302	0,010609	0,010307	0,010319	0,007937
2048	2048	0,051036	0,043936	0,040916	0,039433	0,038645	0,029467
4096	4096	0,202974	0,174597	0,161883	0,155634	0,151517	0,114273
4096	1024	0,051286	0,044041	0,041063	0,039523	0,038736	0,029647
1024	4096	0,051047	0,043908	0,040794	0,039203	0,038515	0,029174

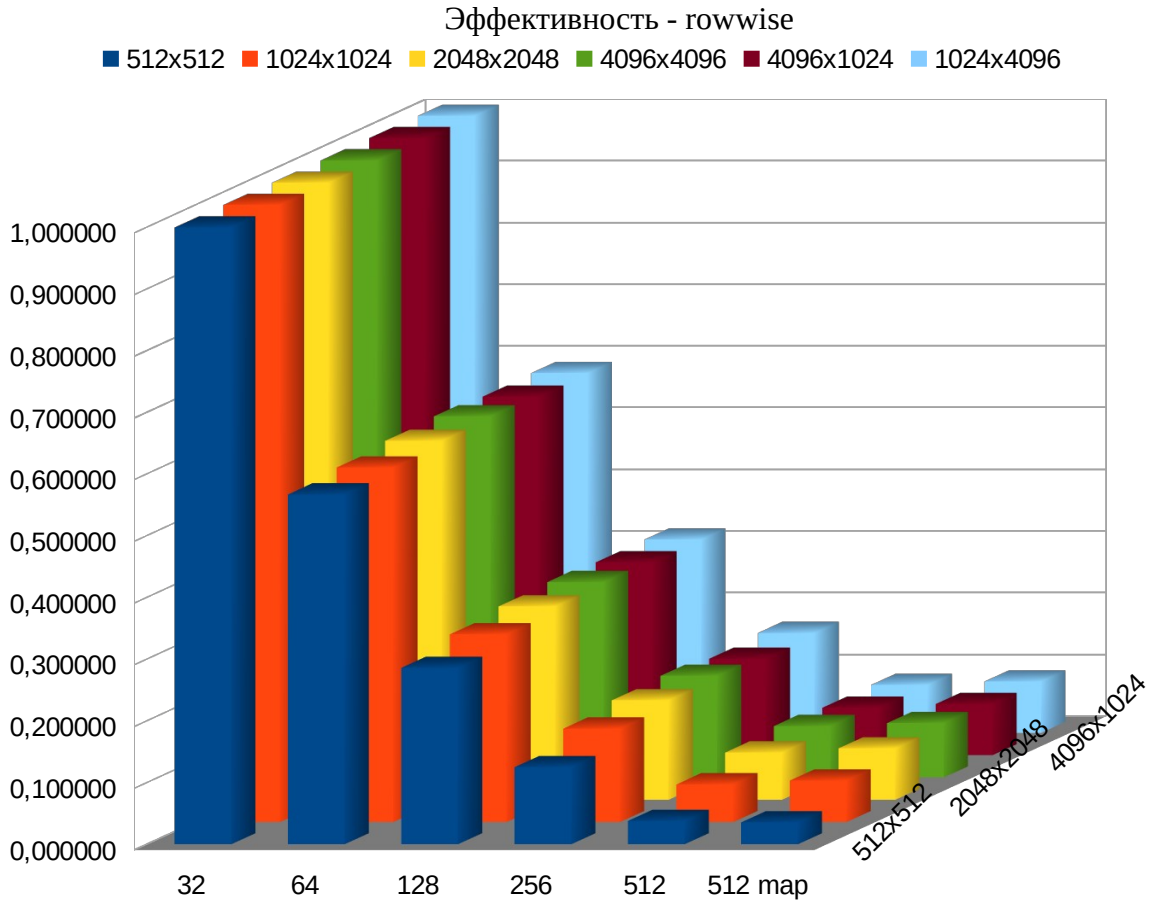
Ускорение, с - columnwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	1,000000	1,140304	1,207005	1,188686	1,096405	1,364752
1024	1024	1,000000	1,154574	1,229993	1,266033	1,264561	1,644072
2048	2048	1,000000	1,161599	1,247336	1,294246	1,320637	1,731971
4096	4096	1,000000	1,162529	1,253831	1,304175	1,339612	1,776220
4096	1024	1,000000	1,164506	1,248959	1,297624	1,323988	1,729888
1024	4096	1,000000	1,162590	1,251336	1,302120	1,325380	1,749743

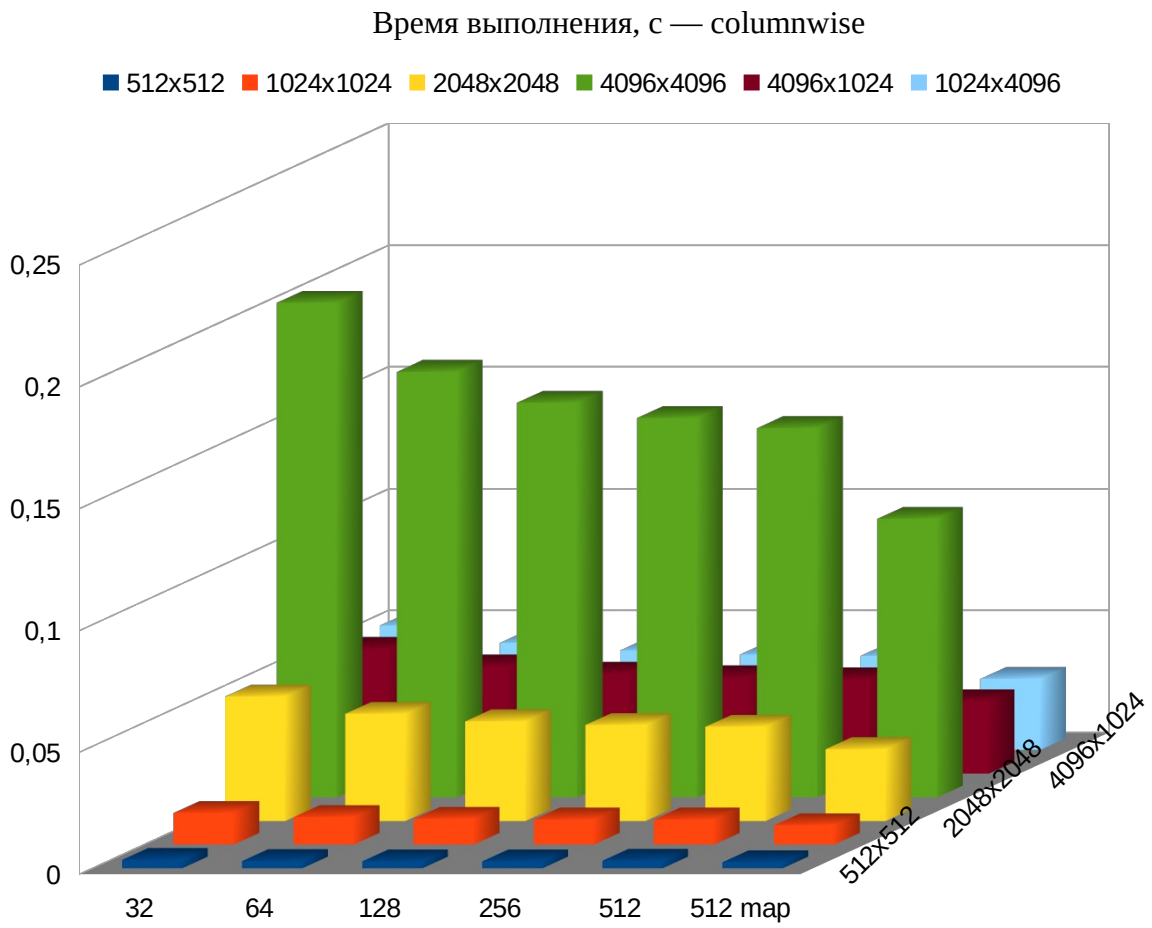
Эффективность, с - columnwise							
n	**m**	**Число процессов**					
		32	**64**	**128**	**256**	**512**	**512 map**
512	512	1,000000	0,570152	0,301751	0,148586	0,068525	0,085297
1024	1024	1,000000	0,577287	0,307498	0,158254	0,079035	0,102755
2048	2048	1,000000	0,580799	0,311834	0,161781	0,082540	0,108248
4096	4096	1,000000	0,581264	0,313458	0,163022	0,083726	0,111014
4096	1024	1,000000	0,582253	0,312240	0,162203	0,082749	0,108118
1024	4096	1,000000	0,581295	0,312834	0,162765	0,082836	0,109359

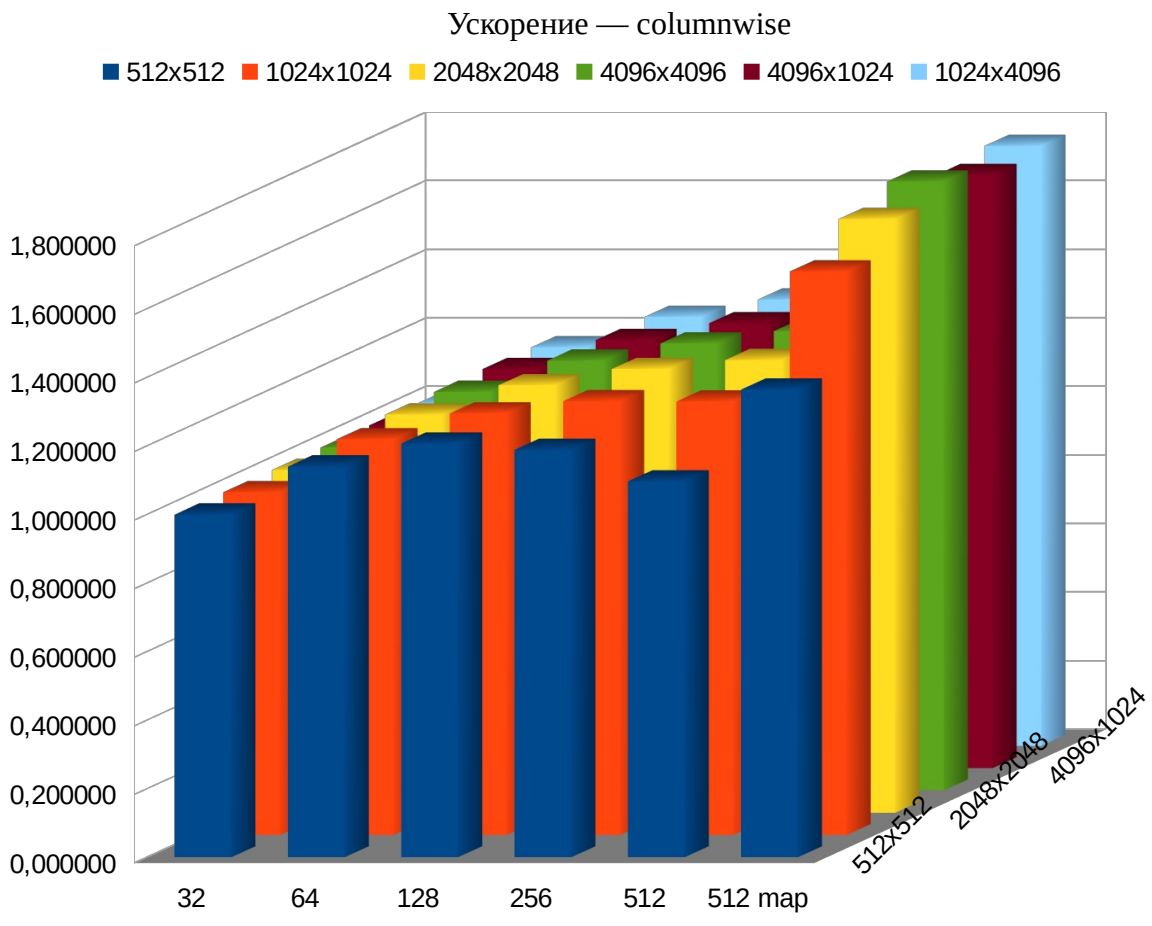
Время выполнения, с - rowwise

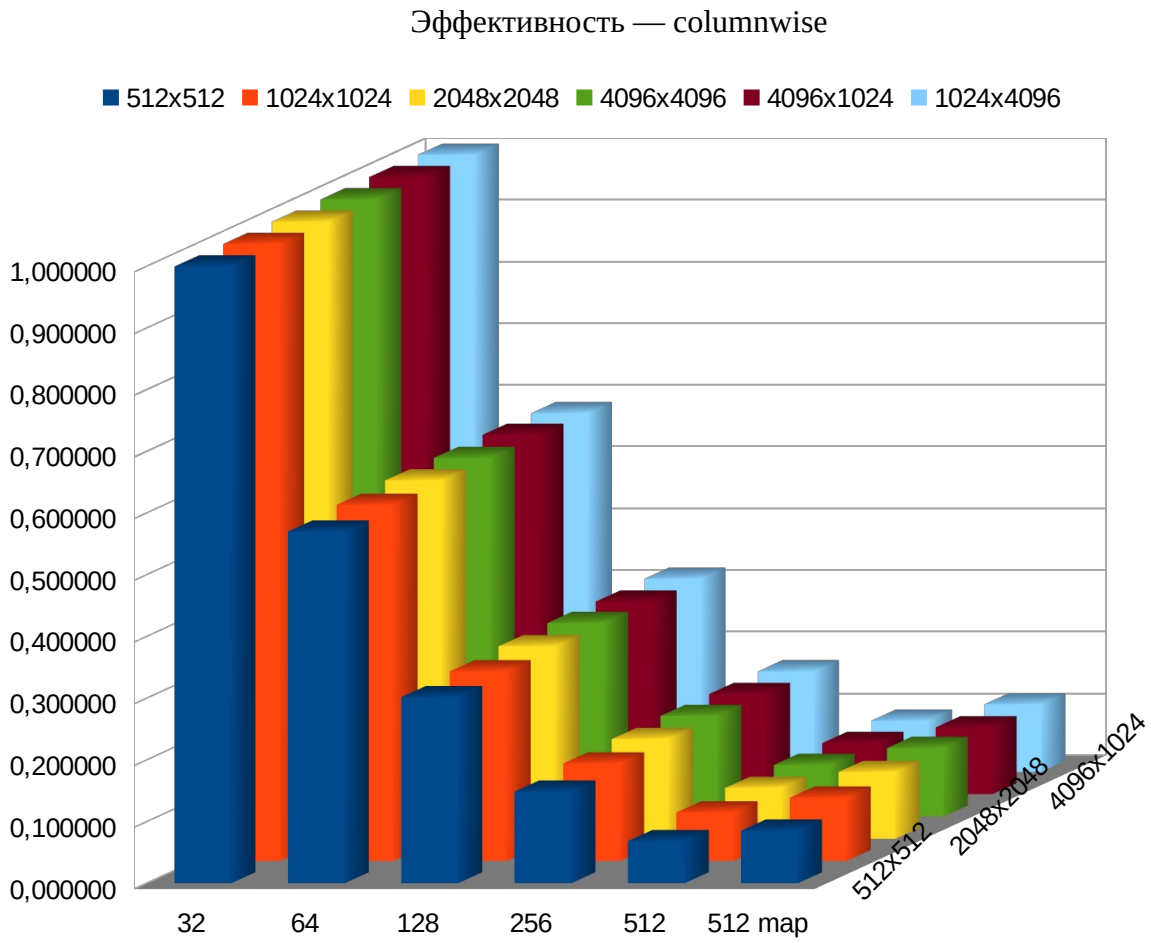



Ускорение - rowwise
512x512
1024x1024
2048x2048
4096x4096
4096x1024
1024x4096
1,600000
1,400000
1,200000
1,000000
0,800000
0,600000
0,400000
0,200000
0,000000
32
64
128
256
512
512 map
512x512
2048x2048
4096x1024


Эффективность - rowwise
512x512
1024x1024
2048x2048
4096x4096
4096x1024
1024x4096
1,000000
0,900000
0,800000
0,700000
0,600000
0,500000
0,400000
0,300000
0,200000
0,100000
0,000000
32
64
128
256
512
512 map
512x512
2048x2048
4096x1024


Время выполнения, с — columnwise
512x512
1024x1024
2048x2048
4096x4096
4096x1024
1024x4096
0,25
0,2
0,15
0,1
0,05
0
32
64
128
256
512
512 map
512x512
2048x2048
4096x1024


Ускорение — columnwise
512x512
1024x1024
2048x2048
4096x4096
4096x1024
1024x4096
1,800000
1,600000
1,400000
1,200000
1,000000
0,800000
0,600000
0,400000
0,200000
0,000000
32
64
128
256
512
512 map
512x512
2048x2048
4096x1024



Основные выводы.

Увеличение числа процессов не дало желанного ускорения, поэтому эффективность оказалась очень низкой. Связано это с накладными расходами на обмен данными между процессами. Columnwise-версия в большинстве случаев оказалась чуть быстрее rowwise-версии. Случайный мэппинг для 512 процессов оказался даже лучше стандартного.