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

Системы и средства параллельного программирования.

Отчёт № 4.

Параллельный алгоритм умножения матрицы на вектор.

Работу выполнил
Сайбель Т. А.

Москва 2020

Постановка задачи и формат данных.

Задача:

1.Разработать параллельную программу с использованием технологии MPI. Предусмотреть равномерное распределение элементов матрицыблоками строк или столбцов, в зависимости от соотношения m и n. Вектора b и c распределены по процессам равномерно.

2.Исследовать эффективность разработанной программы в зависимости от размеров матрицы и количества используемых процессов. Построить графики времени работы, ускорения и эффективности разработанной программы. Время на ввод/вывод данных не включать.

3.Исследовать влияние мэппинга параллельной программы на время работы программы.

4.Построить таблицы: времени, ускорения, эффективности.

5.Построить графики – для каждого из заданных значений размеров матрицы (512x512, 1024x1024, 2048x2048, 4096x4096, 4096x1024, 1024x4096).

6.Подготовить отчет о выполнении задания, включающий таблицы с временами, графики, текст программы. Сделать выводы по полученным результатам (объяснить убывание или возрастание производительности параллельной программы при увеличении числаиспользуемых процессоров, сравнить поведение параллельной программы в зависимости от размеров матрицы).

Формат командной строки:

Параметры, передаваемые в командной строке:

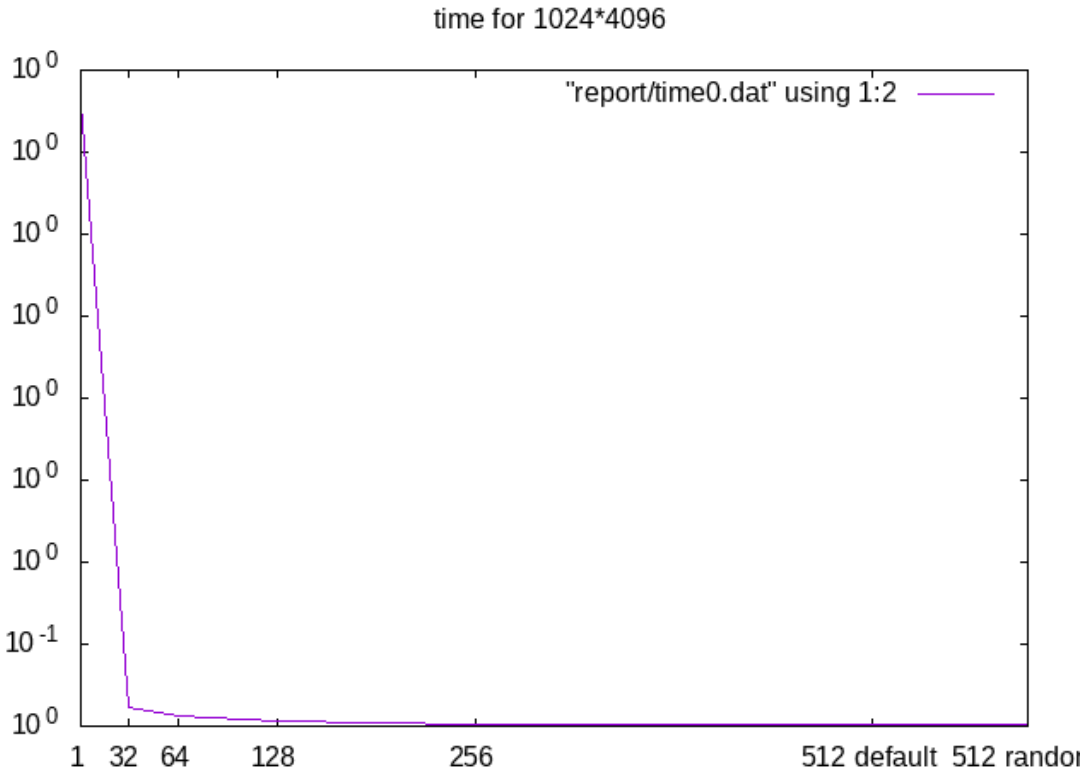
- имя файла
– матрица A размером m x n
- имя файла - вектор в
- имя файла – результат, вектор с

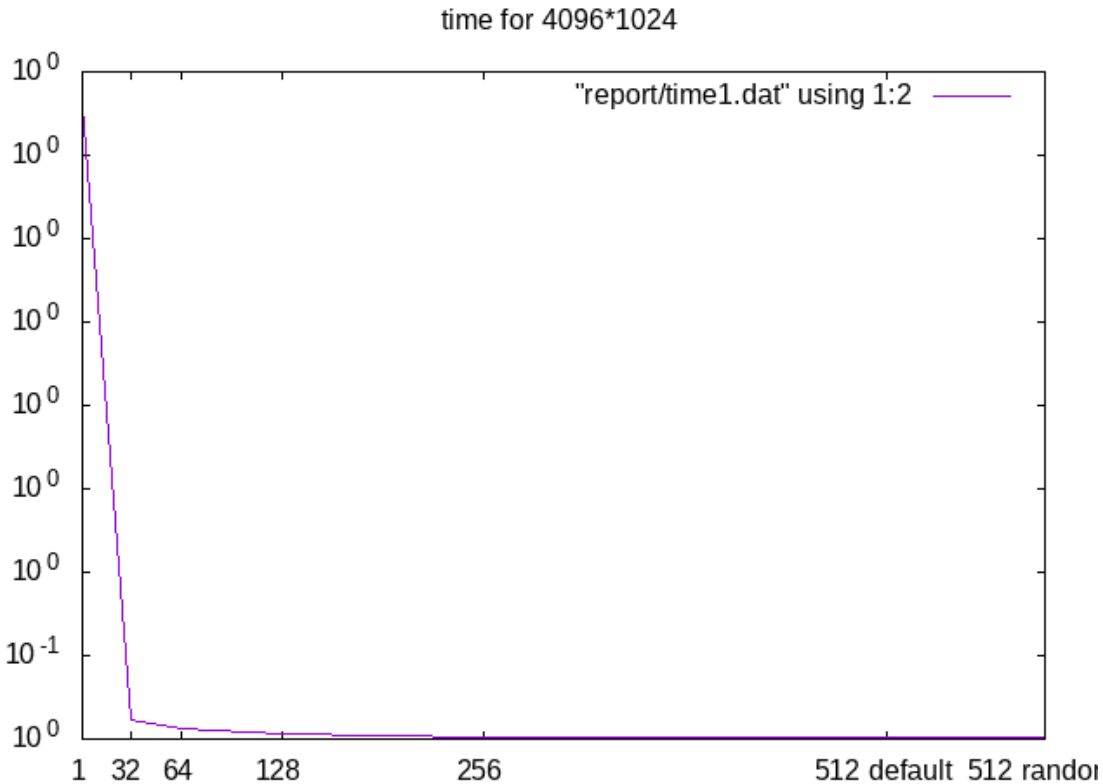
Формат задания матрицы A – как в первом задании

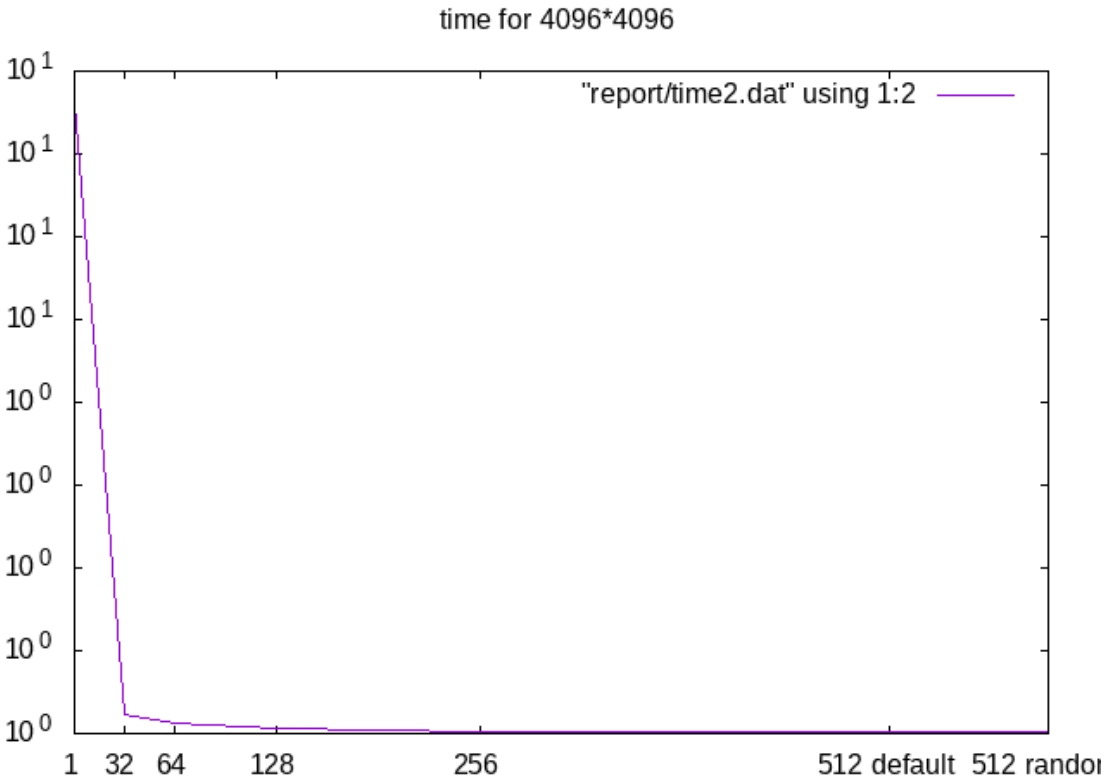
Время выполнения

Таблица 1: Время выполнения

m	n	map	1	32	64	128	256	512
1024	4096		3.80203	0.118569	0.05942	0.0298	0.01506	0.007684
4096	1024	default	3.80309	0.119151	0.05958	0.0297	0.01490	0.007457
4096	4096		15.2572	0.477754	0.23887	0.1194	0.05972	0.02986
1024	4096							0.00768
4096	1024	random						0.00745
4096	4096							0.02986



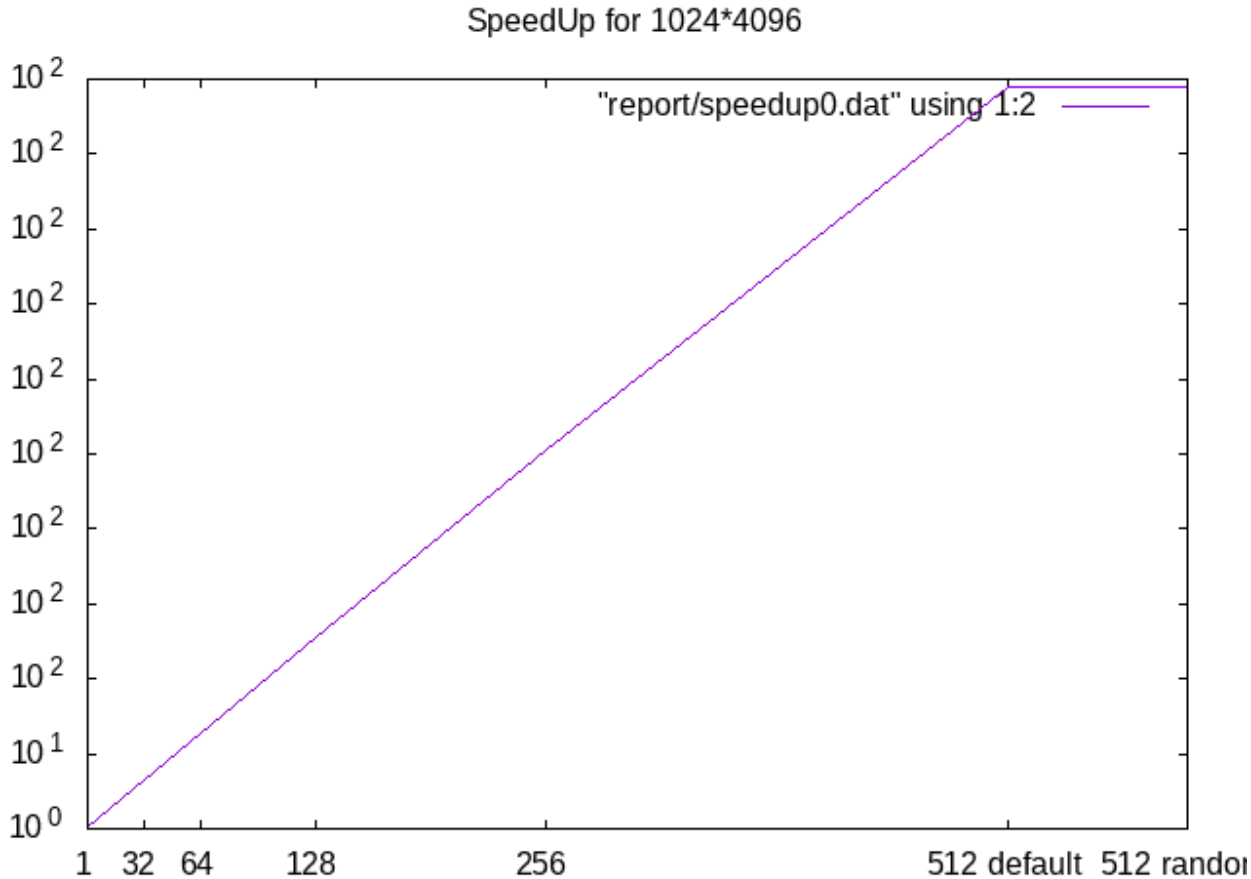


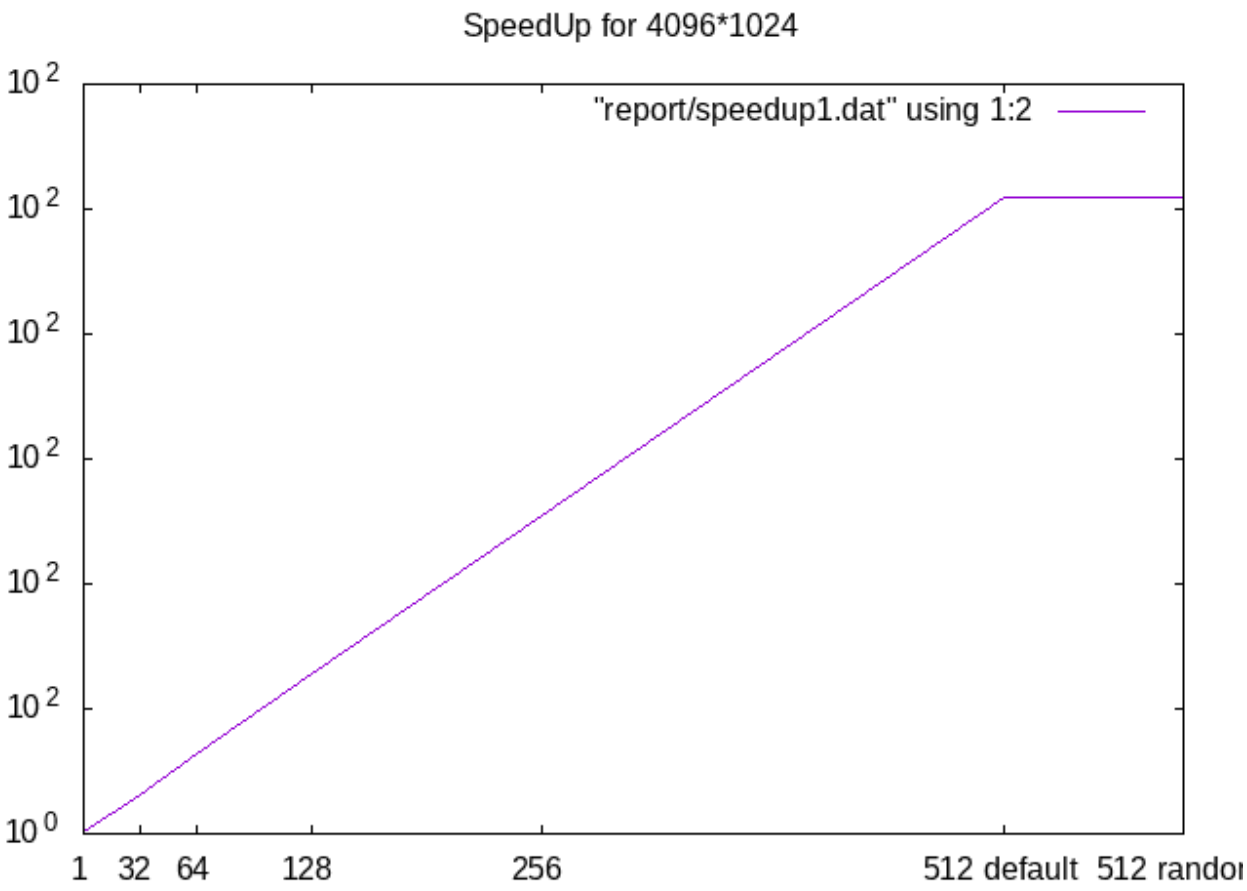


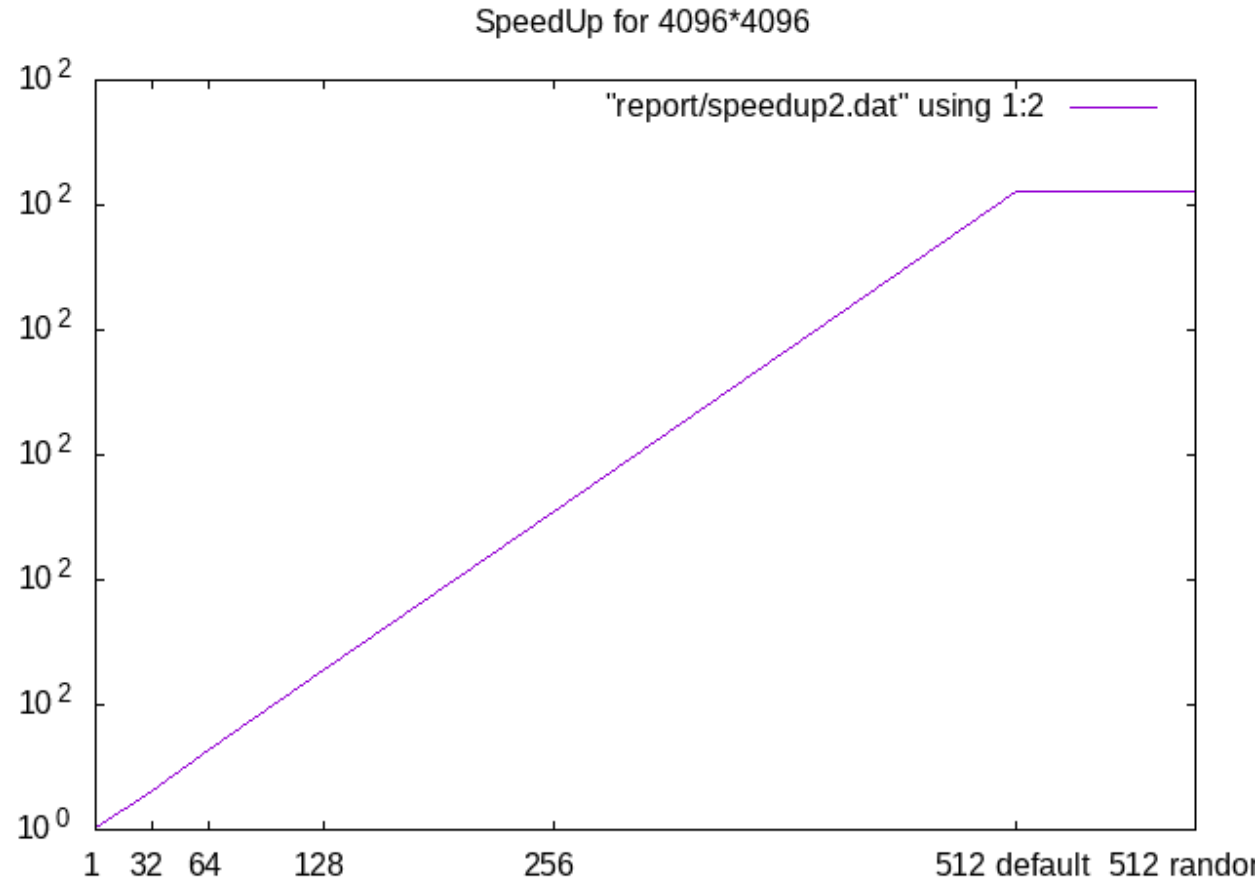
Ускорение

Таблица 2: Ускорение

m	n	map	1	32	64	128	256	512
1024	4096		1	32.0659	63.9847	127.358	252.362	494.783
4096	1024	default	1	31.9181	63.8261	127.642	255.135	509.958
4096	4096		1	31.9353	63.8703	127.735	255.447	510.796
1024	4096							494.765
4096	1024	random						509.969
4096	4096							510.801



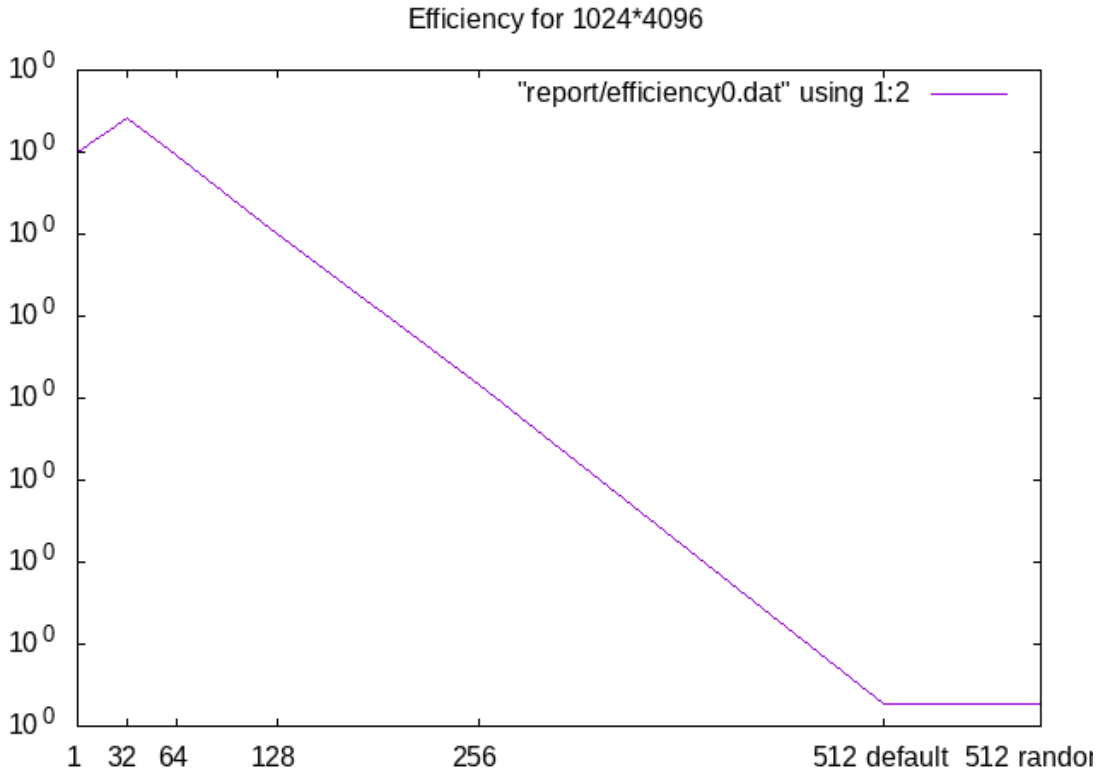

SpeedUp for 4096*1024
"report/speedup1.dat" using 1:2
10 2
10 2
10 2
10 2
10 2
10 2
10 0
1
32
64
128
256
512 default
512 rando

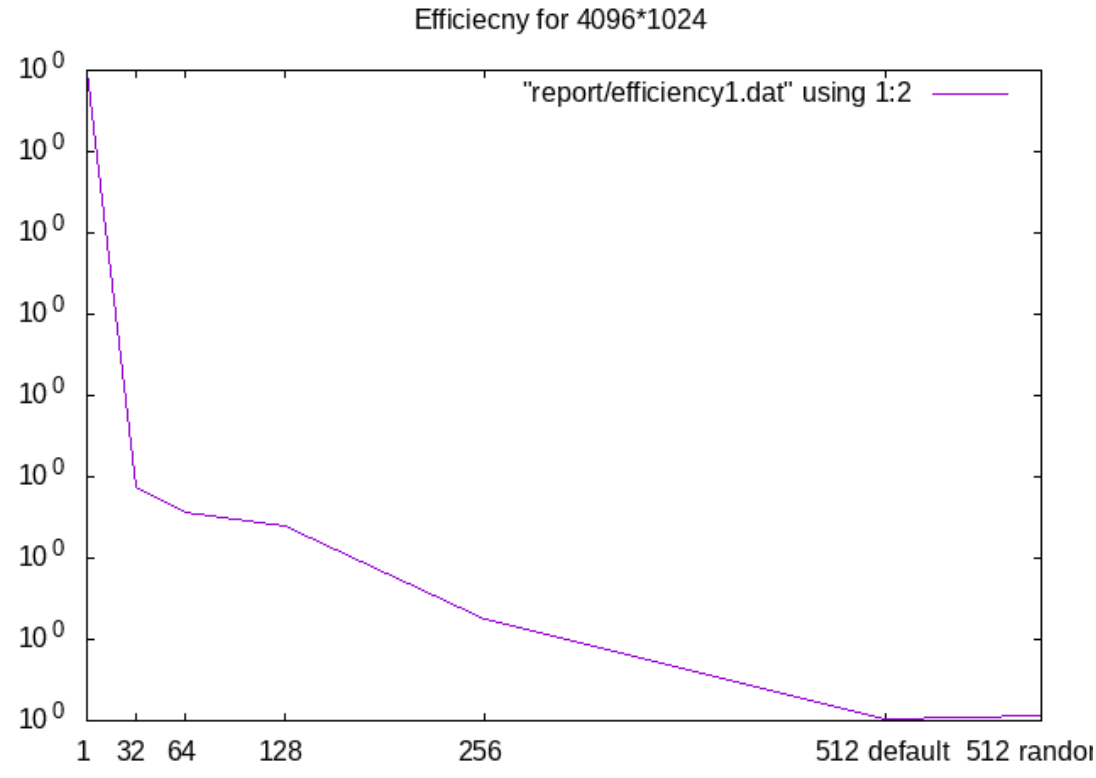

SpeedUp for 4096*4096
"report/speedup2.dat" using 1:2
10 2
10 2
10 2
10 2
10 2
10 2
10 0
1
32
64
128
256
512 default
512 rando

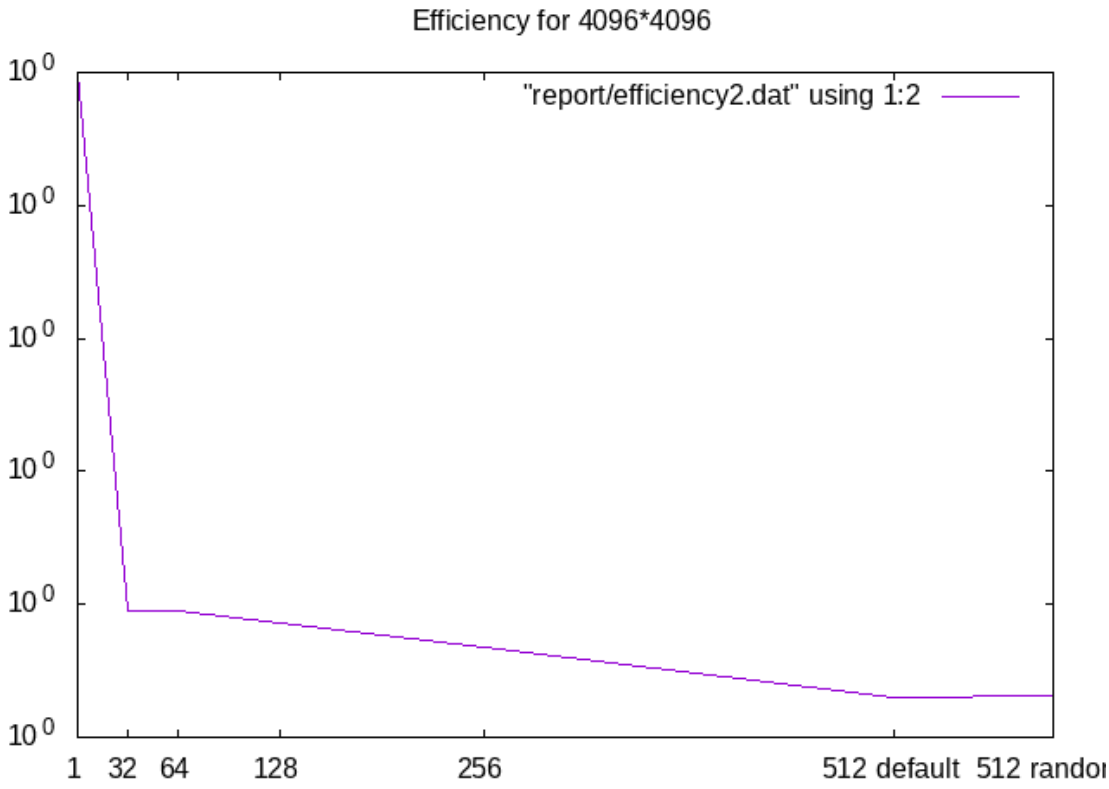
Эффективность

Таблица 3: Эффективность

m	n	map	1	32	64	128	256	512
1024	4096		1	1.00206	0.999761	0.994981	0.985788	0.966373
4096	1024	default	1	0.997442	0.997282	0.997203	0.996621	0.996012
4096	4096		1	0.997978	0.997973	0.997931	0.997841	0.997649
1024	4096							0.966338
4096	1024	random						0.996033
4096	4096							0.997658






Efficiency for 4096*4096
"report/efficiency2.dat" using 1:2
10 0
10 0
10 0
10 0
10 0
10 0
1
32
64
128
256
512 default
512 rando