



Московский государственный университет имени М.В.Ломоносова  
Факультет вычислительной математики и кибернетики

ПАРАЛЛЕЛЬНЫЕ (ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ)  
ВЫЧИСЛЕНИЯ

**Отчет по заданию №3**  
**«Рекурсивная координатная**  
**бисекция»**

*Богатенкова Анастасия Олеговна*  
*528 группа*

17 декабря 2021 г.

# Содержание

1 Постановка задачи	2
2 Описание метода решения	2
3 Описание используемой вычислительной системы	4
4 Результаты численных экспериментов	5
5 Анализ полученных результатов	20
6 Приложение	24

# 1 Постановка задачи

**Дано:**

- $n_1, n_2$  – размеры двумерной сетки, топологически эквивалентной индексному прямоугольнику  $n_1 \times n_2$ ;
- $k$  – число частей (доменов) на которое требуется разбить сетку;
- $fx(i, j)$  и  $fy(i, j)$  – функции, задающие  $x$  и  $y$  координату узла сетки с индексами  $i, j$ ;

Узел сетки с индексами  $(i, j)$  связан с соседними существующими по  $i, j$  узлами:  $(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)$ .

**Цель:**

Написать параллельную программу для вычислительной системы с распределенной памятью, обеспечивающую разбиение методом рекурсивной координатной бисекции узлов сетки на  $k$  доменов. Число вершин в доменах должно быть равно с точностью до одной вершины величине  $\frac{n_1 \cdot n_2}{k}$ ).

**На выходе:**

1. Файл, содержащий  $n_1 \times n_2$  строк, в каждой из которых 5 чисел:  
 $i \ j \ X_{ij} \ Y_{ij} \ d$  // номера вершины в сетке, координаты вершины, номер домена  $[0, ..., k - 1]$ ;
2. Число разрезанных рёбер;
3. Время выполнения декомпозиции (только декомпозиции, без учета формирования сетки и вывода результатов).

# 2 Описание метода решения

В общих чертах алгоритм декомпозиции реализован следующим образом:

1. Нахождение номера элемента массива (и номера процесса при  $p > 1$ ), разделяющего его на две части для дальнейших рекурсивных вызовов;
2. Нахождение оси ( $x$  или  $y$ ), вдоль которой разбиение массива даст минимальное количество разрезанных рёбер. Для этого проводилось следующее:
  - (а) Сортировка по конкретной оси ( $x$  или  $y$ );

- (b) Подсчёт точек, соединённых ребром и попавших в разные массивы;
  - (c) Выбор массива, отсортированного по оси с минимальным результатом числа разрезанных рёбер.
3. Рекурсивные вызовы для каждого из подмассивов со своим количеством доменов для дальнейшей декомпозиции.

Реализация алгоритма на 1 процессе имеет следующие особенности:

- Для сортировки использовалась функция `std :: sort` из стандартной библиотеки;
- Базисом рекурсии является случай  $k = 1$ , в этом случае все точки попадают в один домен;
- `localBisect` – функция, в которой реализована последовательная версия алгоритма;
- `findBisectCoord` – функция нахождения оптимальной оси и сортировки массива;
- `countEdges` – функция подсчёта числа разрезанных рёбер на 1 процессе.

Параллельная версия алгоритма более сложная:

- Для сортировки использовалась сортировка Бэтчера, реализованная в предыдущем задании;
- Базисом рекурсии являются случаи  $k = 1$  (все точки попадают в один домен) и  $p = 1$  (запускается последовательная версия алгоритма);
- `recursiveBisect` – функция, в которой реализована параллельная версия алгоритма;
- `findBisectCoordParallel` – функция нахождения оптимальной оси и сортировки массива в параллельном случае;
- `countEdgesParallel` – функция подсчёта числа разрезанных рёбер в параллельном случае, когда массив находится на нескольких процессах.

В параллельной версии выбора оси происходит сортировка (Бэтчера) массива по каждой из осей, затем массив делится на две части и элементы перераспределяются, после чего происходит подсчет разрезанных рёбер, на основании

этого выбирается массив, отсортированный по одной из осей, для продолжения работы алгоритма.

При разделении массива на две части в параллельном случае, находится номер разделяющего процесса, на котором располагается массив с разделяющим элементом. На этом процессе все элементы массива, находящиеся до разделяющего элемента, равномерно пересыпаются на процессы с меньшими номерами (если разделяющий процесс – самый первый, он пересыпает элементы с большими номерами процессам с большими номерами). При этом отправленные элементы заменяются на фиктивные. Если элементов для отправки не хватает, отправляются фиктивные элементы. Равномерное распределение элементов необходимо для корректной работы сортировки Бэтчера, которая требует, чтобы на каждом процессе было равное количество элементов массива.

После перераспределения элементов происходит подсчёт рёбер:

1. каждый процесс первой группы процессов получает от каждого процесса из второй группы массив;
2. для каждой пары процессов происходит подсчёт числа разрезанных рёбер на одном из процессов первой группы (аналогично последовательной версии);
3. с помощью функции *MPI\_Reduce* процесс с нулевым номером получает сумму рёбер и рассыпает её остальным процессам с помощью *MPI\_Bcast*.

После того, как массив был отсортирован по нужной оси, запускаются рекурсивные вызовы для каждого из подмассивов. Для этого создаются новые группы процессов (коммуникаторы) с помощью функции *MPI\_Comm\_split* и для каждой из этих групп параллельный алгоритм работает аналогично.

### 3 Описание используемой вычислительной системы

Программа тестировалась на суперкомпьютере «IBM BlueGene/P». IBM Blue Gene/P – массивно-параллельная вычислительная система, которая состоит из двух стоек, включающих 8192 процессорных ядер ( $2 \times 1024$  четырехъядерных вычислительных узлов), с пиковой производительностью 27.9 терафлопс (27.8528 триллионов операций с плавающей точкой в секунду).

Характеристики вычислительной системы представлены в таблице 1.

Число стоек	2
Вычислительных узлов	1024
Процессоры IBM PowerPC 450	2048
Число процессорных ядер	4
Общее число ядер	8192
Оперативная память на узле	2 Гбайт
Число операций в секунду на ядро	3.4 GFlop/s
Коммуникационная сеть	трехмерный тор
Латентность (ближайший сосед)	0.1 $\mu s$ (32-б. пакет), 0.8 $\mu s$ (256-б. пакет)
Пропускная способность интерконнекта	425 MB/s
Система хранения данных	GPFS
Операционная система	Linux

Таблица 1: Характеристики «IBM BlueGene/P»

## 4 Результаты численных экспериментов

Сортировка проводилась на сетках размера  $128 \times 128$ ,  $256 \times 256$ ,  $512 \times 512$  при  $k = 16, 31, 64$ . Для двух рассмотренных сеток  $x = 10i$ ,  $y = 10j$  и  $x = 10i + 10j + i$ ,  $y = 10j - 10i + j$  временные результаты получились практически одинаковыми, при этом число разрезанных рёбер на второй сетке больше, чем на первой. В таблицах 2, 3, 4, 5, 6, 7 и на графиках (рисунки 1, 2, 3, 4, 5, 6, 7, 8, 9) приведены результаты работы алгоритма:

- $p$  – число процессов;
- $size$  – размер сетки;
- $T$  – время работы алгоритма (сек);
- $S = \frac{T_1}{T}$  – ускорение, полученное при численном расчете;
- $E = \frac{T_1}{T \cdot p}$  – эффективность, полученная при численном расчете;
- $edges$  – число разрезанных рёбер.

В силу того, что результаты для двух сеток аналогичны, графики сделаны только для второй сетки.

Таблица 2: Результаты для  $k = 16$ , сеть  $x = 10i$ ,  $y = 10j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	16.773	1.000	1.000	768
2	$128 \times 128$	13.543	1.238	0.619	768
4	$128 \times 128$	8.069	2.079	0.520	768
8	$128 \times 128$	4.031	4.161	0.520	768
16	$128 \times 128$	2.100	7.988	0.499	768
32	$128 \times 128$	1.058	15.846	0.495	768
64	$128 \times 128$	0.537	31.217	0.488	768
128	$128 \times 128$	0.277	60.578	0.473	768
256	$128 \times 128$	0.155	108.514	0.424	768
1	$256 \times 256$	271.833	1.000	1.000	1536
2	$256 \times 256$	218.367	1.245	0.622	1536
4	$256 \times 256$	129.827	2.094	0.523	1536
8	$256 \times 256$	70.109	3.877	0.485	1536
16	$256 \times 256$	36.380	7.472	0.467	1536
32	$256 \times 256$	16.664	16.313	0.510	1536
64	$256 \times 256$	8.353	32.543	0.508	1536
128	$256 \times 256$	4.197	64.776	0.506	1536
256	$256 \times 256$	2.125	127.903	0.500	1536
1	$512 \times 512$	4342.100	1.000	1.000	3072
2	$512 \times 512$	3490.010	1.244	0.622	3072
4	$512 \times 512$	2074.890	2.093	0.523	3072
8	$512 \times 512$	1120.040	3.877	0.485	3072
16	$512 \times 512$	580.770	7.476	0.467	3072
32	$512 \times 512$	290.495	14.947	0.467	3072
64	$512 \times 512$	145.341	29.875	0.467	3072
128	$512 \times 512$	66.522	65.273	0.510	3072
256	$512 \times 512$	33.325	130.297	0.509	3072

Таблица 3: Результаты для  $k = 16$ , сеть  $x = 10i + 10j + i$ ,  $y = 10j - 10i + j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	16.773	1.000	1.000	1224
2	$128 \times 128$	13.542	1.239	0.619	1224
4	$128 \times 128$	8.069	2.079	0.520	1224
8	$128 \times 128$	4.031	4.161	0.520	1224
16	$128 \times 128$	2.100	7.988	0.499	1224
32	$128 \times 128$	1.059	15.844	0.495	1224
64	$128 \times 128$	0.537	31.215	0.488	1224
128	$128 \times 128$	0.277	60.593	0.473	1224
256	$128 \times 128$	0.155	108.509	0.424	1224
1	$256 \times 256$	271.834	1.000	1.000	2464
2	$256 \times 256$	218.363	1.245	0.622	2464
4	$256 \times 256$	129.827	2.094	0.523	2464
8	$256 \times 256$	70.107	3.877	0.485	2464
16	$256 \times 256$	36.380	7.472	0.467	2464
32	$256 \times 256$	16.665	16.312	0.510	2464
64	$256 \times 256$	8.354	32.541	0.508	2464
128	$256 \times 256$	4.197	64.776	0.506	2464
256	$256 \times 256$	2.125	127.918	0.500	2464
1	$512 \times 512$	4342.130	1.000	1.000	4936
2	$512 \times 512$	3490.010	1.244	0.622	4936
4	$512 \times 512$	2074.880	2.093	0.523	4936
8	$512 \times 512$	1120.030	3.877	0.485	4936
16	$512 \times 512$	580.772	7.476	0.467	4936
32	$512 \times 512$	290.496	14.947	0.467	4936
64	$512 \times 512$	145.343	29.875	0.467	4936
128	$512 \times 512$	66.524	65.272	0.510	4936
256	$512 \times 512$	33.325	130.298	0.509	4936

Таблица 4: Результаты для  $k = 31$ , сеть  $x = 10i$ ,  $y = 10j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	17.302	1.000	1.000	1264
2	$128 \times 128$	14.163	1.222	0.611	1264
4	$128 \times 128$	8.463	2.044	0.511	1264
8	$128 \times 128$	4.232	4.088	0.511	1264
16	$128 \times 128$	2.205	7.846	0.490	1264
32	$128 \times 128$	1.393	12.420	0.388	1264
64	$128 \times 128$	0.628	27.536	0.430	1264
128	$128 \times 128$	0.307	56.396	0.441	1264
256	$128 \times 128$	0.166	104.093	0.407	1264
1	$256 \times 256$	279.792	1.000	1.000	2498
2	$256 \times 256$	228.224	1.226	0.613	2498
4	$256 \times 256$	136.152	2.055	0.514	2498
8	$256 \times 256$	104.493	2.678	0.335	2498
16	$256 \times 256$	53.135	5.266	0.329	2498
32	$256 \times 256$	21.847	12.807	0.400	2498
64	$256 \times 256$	8.857	31.590	0.494	2498
128	$256 \times 256$	4.605	60.756	0.475	2498
256	$256 \times 256$	2.263	123.650	0.483	2498
1	$512 \times 512$	4482.770	1.000	1.000	4967
2	$512 \times 512$	3653.800	1.227	0.613	4967
4	$512 \times 512$	2178.380	2.058	0.514	4967
8	$512 \times 512$	1177.170	3.808	0.476	4967
16	$512 \times 512$	847.996	5.286	0.330	4967
32	$512 \times 512$	393.327	11.397	0.356	4967
64	$512 \times 512$	171.407	26.153	0.409	4967
128	$512 \times 512$	71.193	62.967	0.492	4967
256	$512 \times 512$	34.591	129.592	0.506	4967

Таблица 5: Результаты для  $k = 31$ , сеть  $x = 10i + 10j + i$ ,  $y = 10j - 10i + j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	17.302	1.000	1.000	1775
2	$128 \times 128$	14.161	1.222	0.611	1775
4	$128 \times 128$	8.463	2.044	0.511	1775
8	$128 \times 128$	4.232	4.089	0.511	1775
16	$128 \times 128$	2.205	7.845	0.490	1775
32	$128 \times 128$	1.393	12.417	0.388	1775
64	$128 \times 128$	0.628	27.535	0.430	1775
128	$128 \times 128$	0.307	56.425	0.441	1775
256	$128 \times 128$	0.167	103.722	0.405	1775
1	$256 \times 256$	279.795	1.000	1.000	3576
2	$256 \times 256$	228.220	1.226	0.613	3576
4	$256 \times 256$	136.151	2.055	0.514	3576
8	$256 \times 256$	104.493	2.678	0.335	3576
16	$256 \times 256$	53.137	5.266	0.329	3576
32	$256 \times 256$	21.848	12.807	0.400	3576
64	$256 \times 256$	8.857	31.589	0.494	3576
128	$256 \times 256$	4.606	60.748	0.475	3576
256	$256 \times 256$	2.264	123.611	0.483	3576
1	$512 \times 512$	4482.810	1.000	1.000	7183
2	$512 \times 512$	3653.800	1.227	0.613	7183
4	$512 \times 512$	2178.380	2.058	0.514	7183
8	$512 \times 512$	1177.170	3.808	0.476	7183
16	$512 \times 512$	848.000	5.286	0.330	7183
32	$512 \times 512$	393.331	11.397	0.356	7183
64	$512 \times 512$	171.410	26.153	0.409	7183
128	$512 \times 512$	71.193	62.967	0.492	7183
256	$512 \times 512$	34.592	129.590	0.506	7183

Таблица 6: Результаты для  $k = 64$ , сеть  $x = 10i$ ,  $y = 10j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	17.613	1.000	1.000	1792
2	$128 \times 128$	13.957	1.262	0.631	1792
4	$128 \times 128$	8.275	2.128	0.532	1792
8	$128 \times 128$	4.133	4.262	0.533	1792
16	$128 \times 128$	2.151	8.190	0.512	1792
32	$128 \times 128$	1.103	15.964	0.499	1792
64	$128 \times 128$	0.566	31.142	0.487	1792
128	$128 \times 128$	0.293	60.200	0.470	1792
256	$128 \times 128$	0.164	107.594	0.420	1792
1	$256 \times 256$	284.471	1.000	1.000	3584
2	$256 \times 256$	224.660	1.266	0.633	3584
4	$256 \times 256$	132.967	2.139	0.535	3584
8	$256 \times 256$	71.678	3.969	0.496	3584
16	$256 \times 256$	37.164	7.655	0.478	3584
32	$256 \times 256$	17.354	16.393	0.512	3584
64	$256 \times 256$	8.775	32.418	0.507	3584
128	$256 \times 256$	4.411	64.498	0.504	3584
256	$256 \times 256$	2.235	127.297	0.497	3584
1	$512 \times 512$	4560.570	1.000	1.000	7168
2	$512 \times 512$	3599.130	1.267	0.634	7168
4	$512 \times 512$	2129.420	2.142	0.535	7168
8	$512 \times 512$	1147.300	3.975	0.497	7168
16	$512 \times 512$	594.398	7.673	0.480	7168
32	$512 \times 512$	302.468	15.078	0.471	7168
64	$512 \times 512$	152.626	29.881	0.467	7168
128	$512 \times 512$	69.862	65.279	0.510	7168
256	$512 \times 512$	35.002	130.293	0.509	7168

Таблица 7: Результаты для  $k = 64$ , сеть  $x = 10i + 10j + i$ ,  $y = 10j - 10i + j$

$p$	$size$	$T$	$S$	$E$	$edges$
1	$128 \times 128$	17.613	1.000	1.000	2640
2	$128 \times 128$	13.955	1.262	0.631	2640
4	$128 \times 128$	8.274	2.129	0.532	2640
8	$128 \times 128$	4.133	4.262	0.533	2640
16	$128 \times 128$	2.151	8.190	0.512	2640
32	$128 \times 128$	1.103	15.964	0.499	2640
64	$128 \times 128$	0.566	31.142	0.487	2640
128	$128 \times 128$	0.292	60.221	0.470	2640
256	$128 \times 128$	0.163	107.818	0.421	2640
1	$256 \times 256$	284.473	1.000	1.000	5336
2	$256 \times 256$	224.651	1.266	0.633	5336
4	$256 \times 256$	132.964	2.139	0.535	5336
8	$256 \times 256$	71.674	3.969	0.496	5336
16	$256 \times 256$	37.164	7.655	0.478	5336
32	$256 \times 256$	17.354	16.392	0.512	5336
64	$256 \times 256$	8.775	32.417	0.507	5336
128	$256 \times 256$	4.410	64.499	0.504	5336
256	$256 \times 256$	2.235	127.304	0.497	5336
1	$512 \times 512$	4560.600	1.000	1.000	10738
2	$512 \times 512$	3599.110	1.267	0.634	10738
4	$512 \times 512$	2129.410	2.142	0.535	10738
8	$512 \times 512$	1147.290	3.975	0.497	10738
16	$512 \times 512$	594.396	7.673	0.480	10738
32	$512 \times 512$	302.467	15.078	0.471	10738
64	$512 \times 512$	152.626	29.881	0.467	10738
128	$512 \times 512$	69.863	65.279	0.510	10738
256	$512 \times 512$	35.002	130.295	0.509	10738

Рис. 1: Временные результаты сортировки для сеток разного размера при  $k = 16$

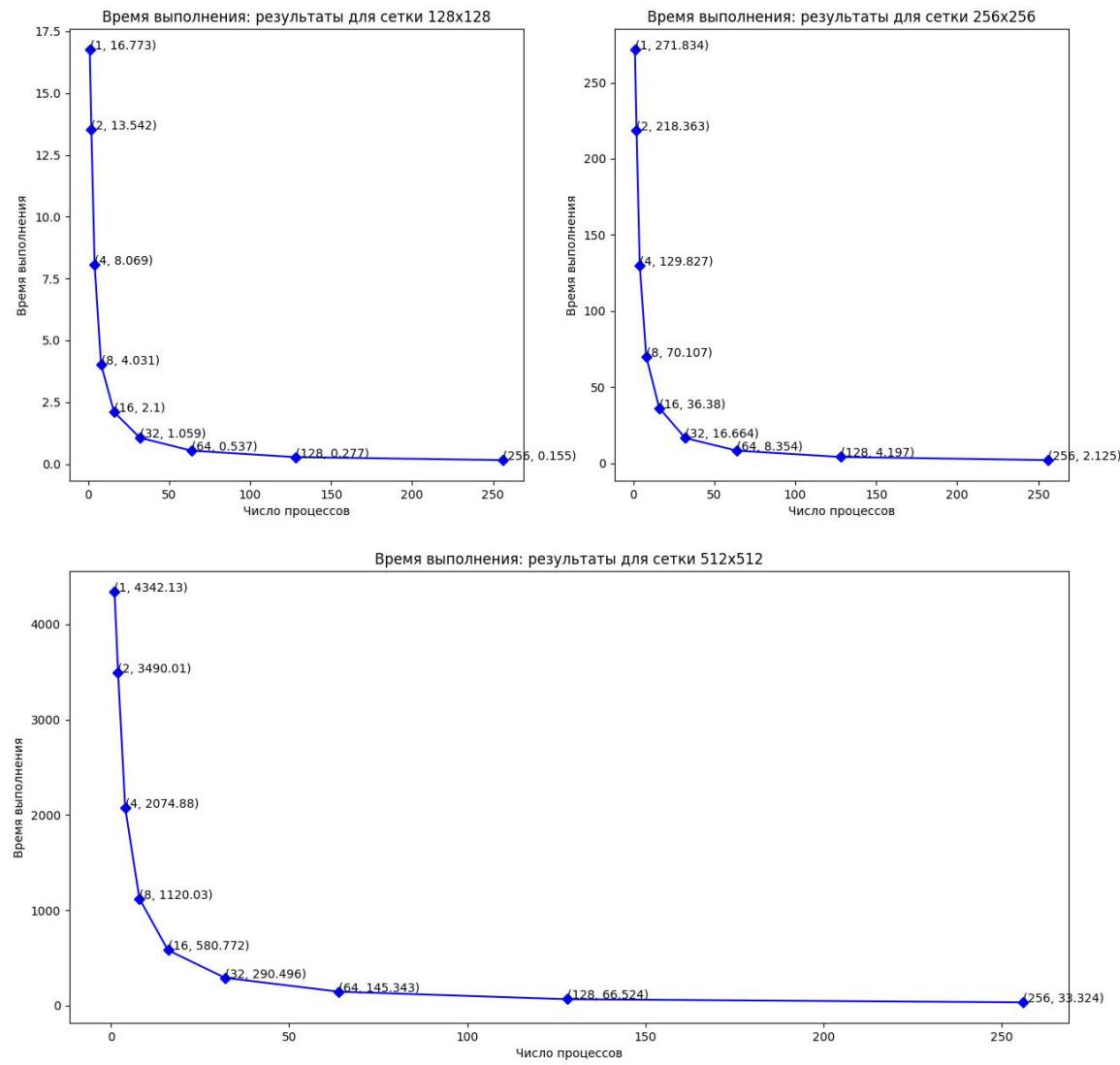


Рис. 2: Ускорение: результаты сортировки для сеток разного размера при  $k = 16$

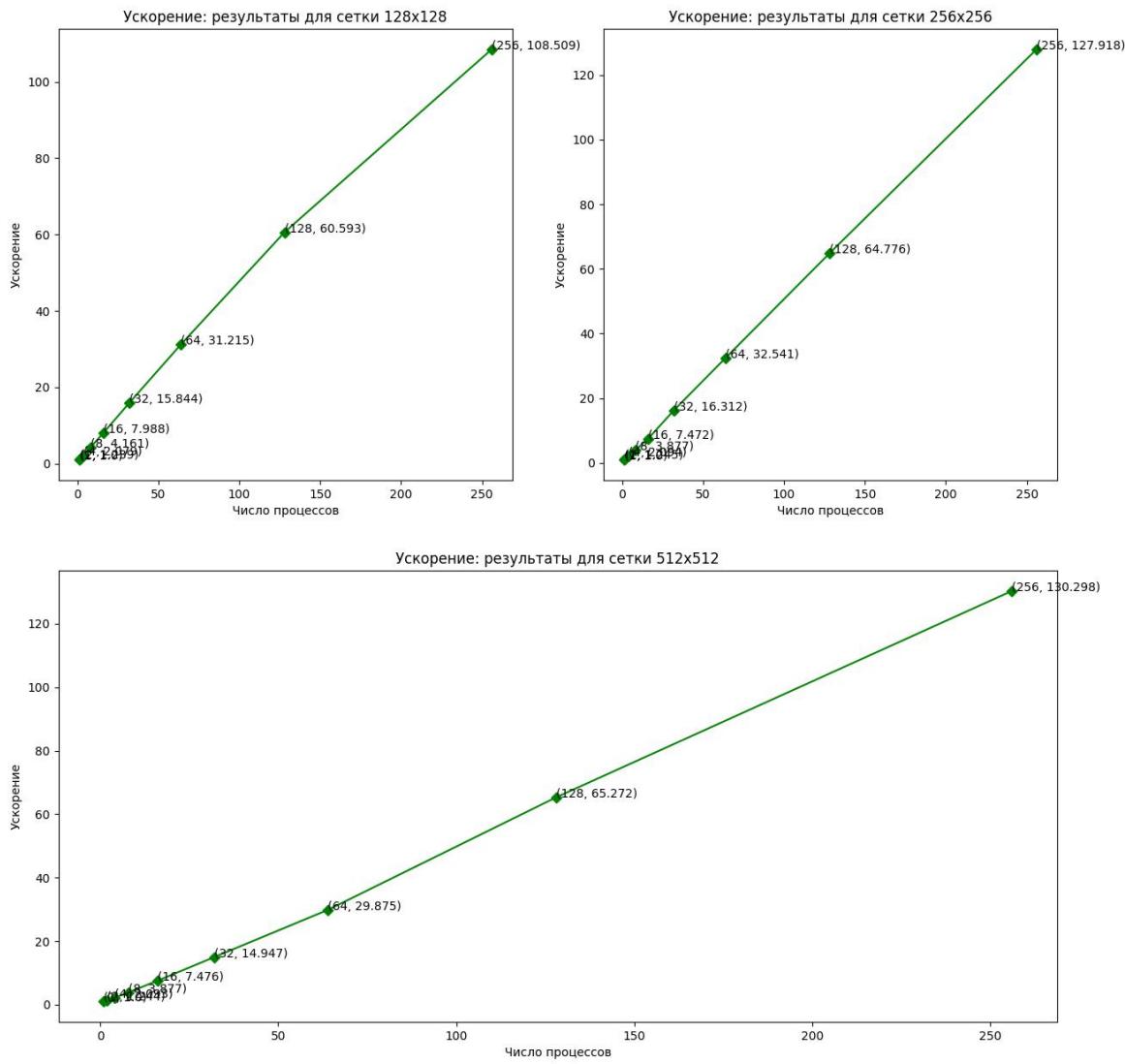


Рис. 3: Эффективность: результаты сортировки для сеток разного размера при  $k = 16$

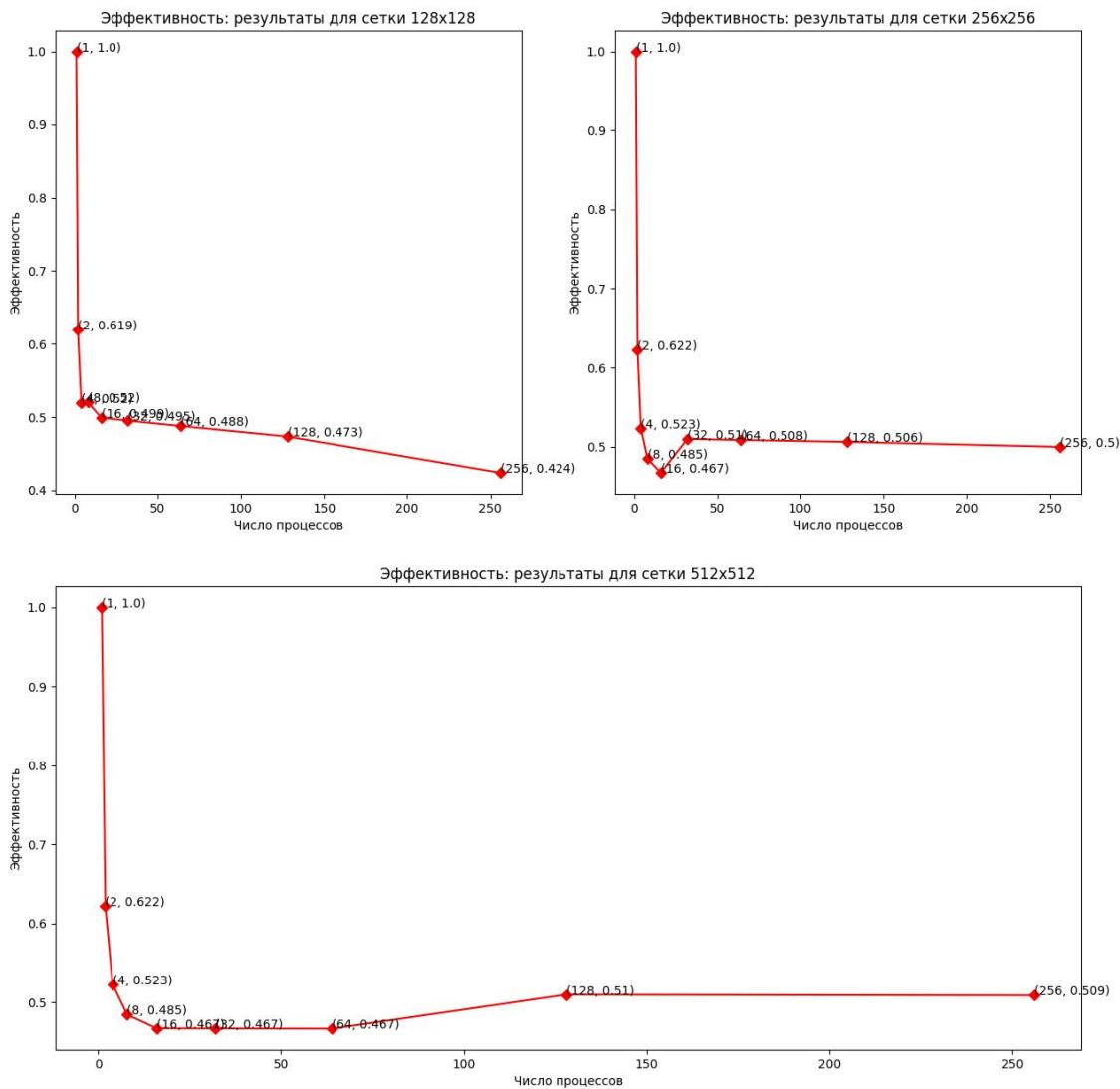


Рис. 4: Временные результаты сортировки для сеток разного размера при  $k = 31$

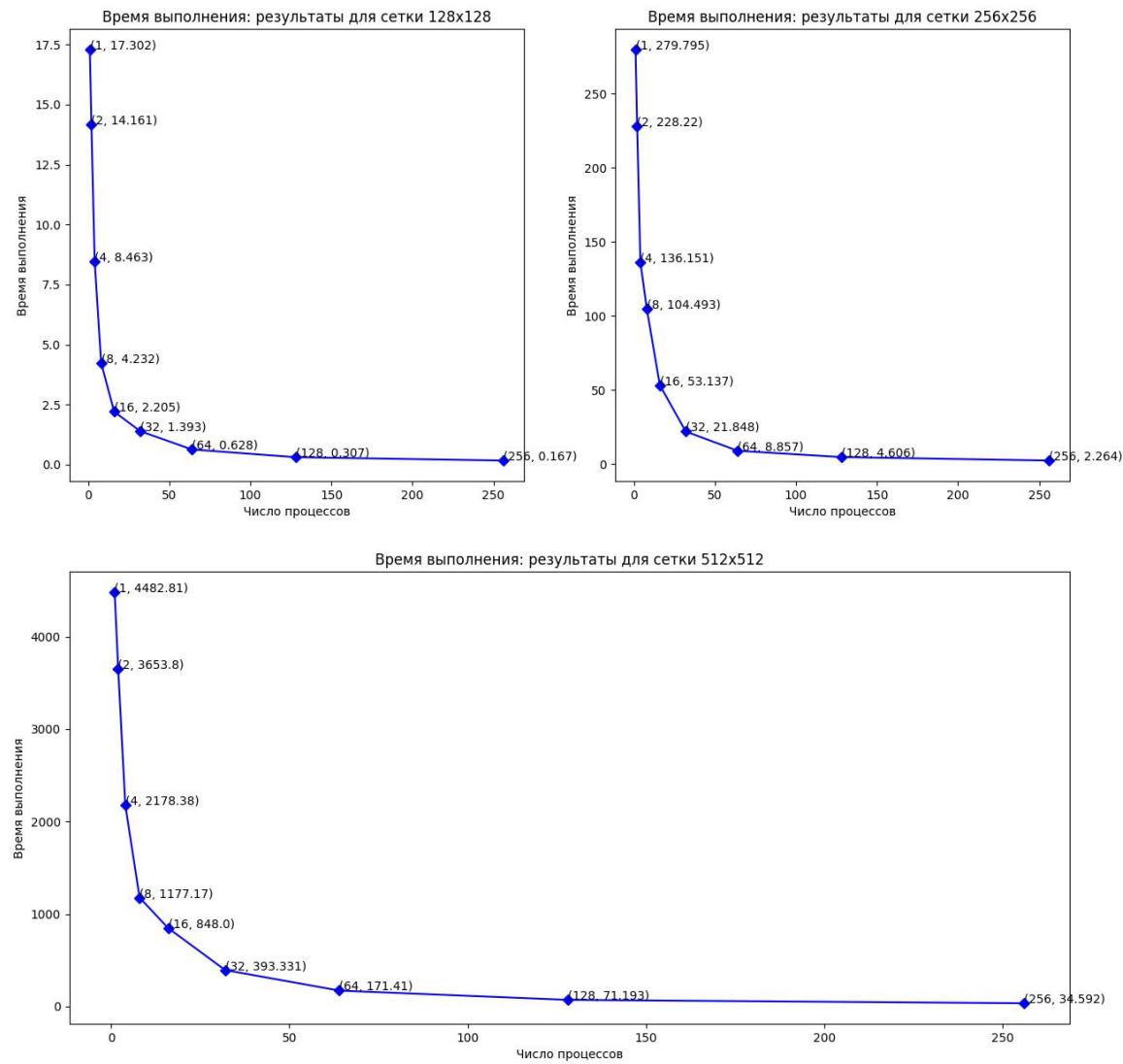


Рис. 5: Ускорение: результаты сортировки для сеток разного размера при  $k = 31$

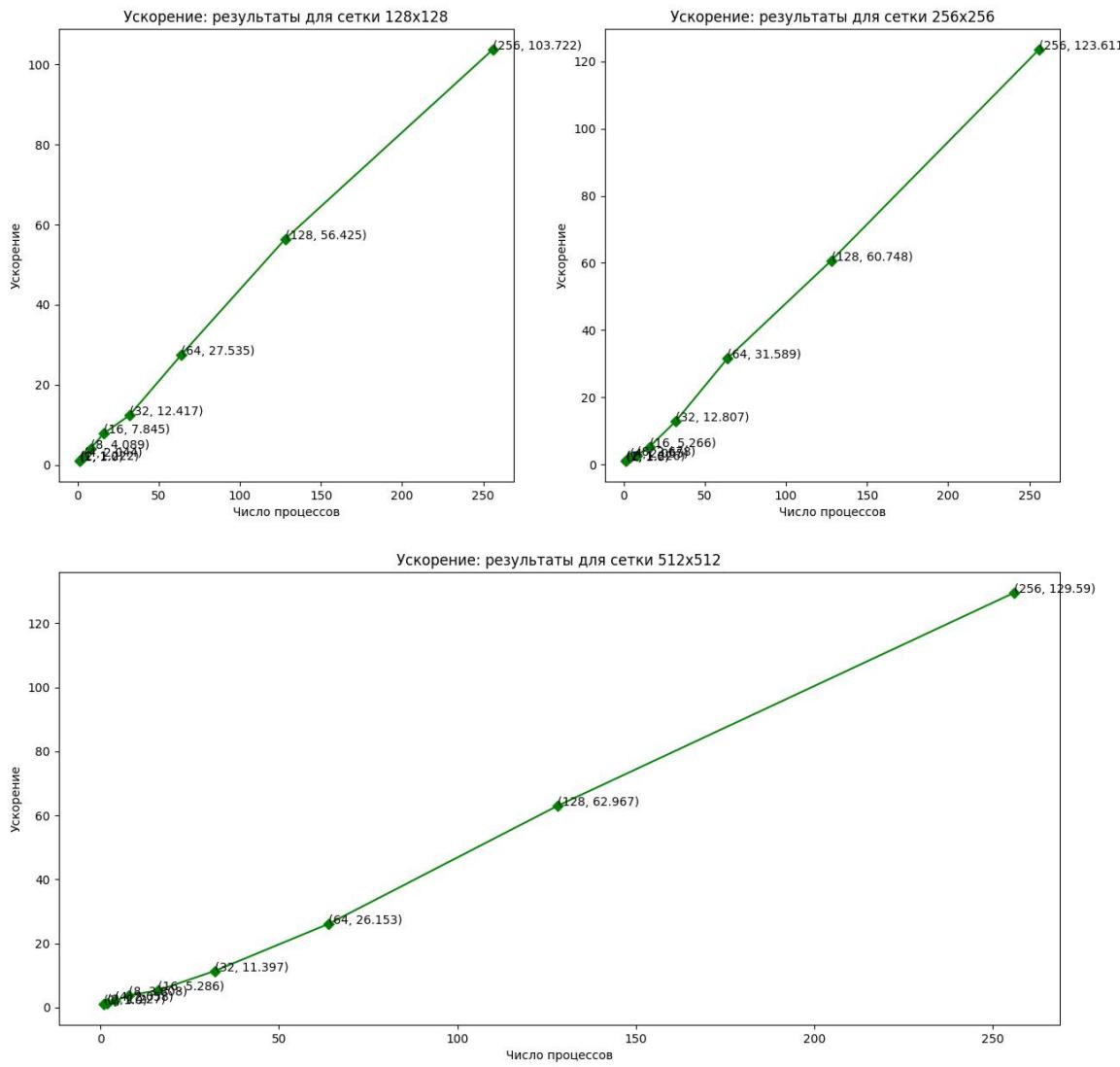


Рис. 6: Эффективность: результаты сортировки для сеток разного размера при  $k = 31$

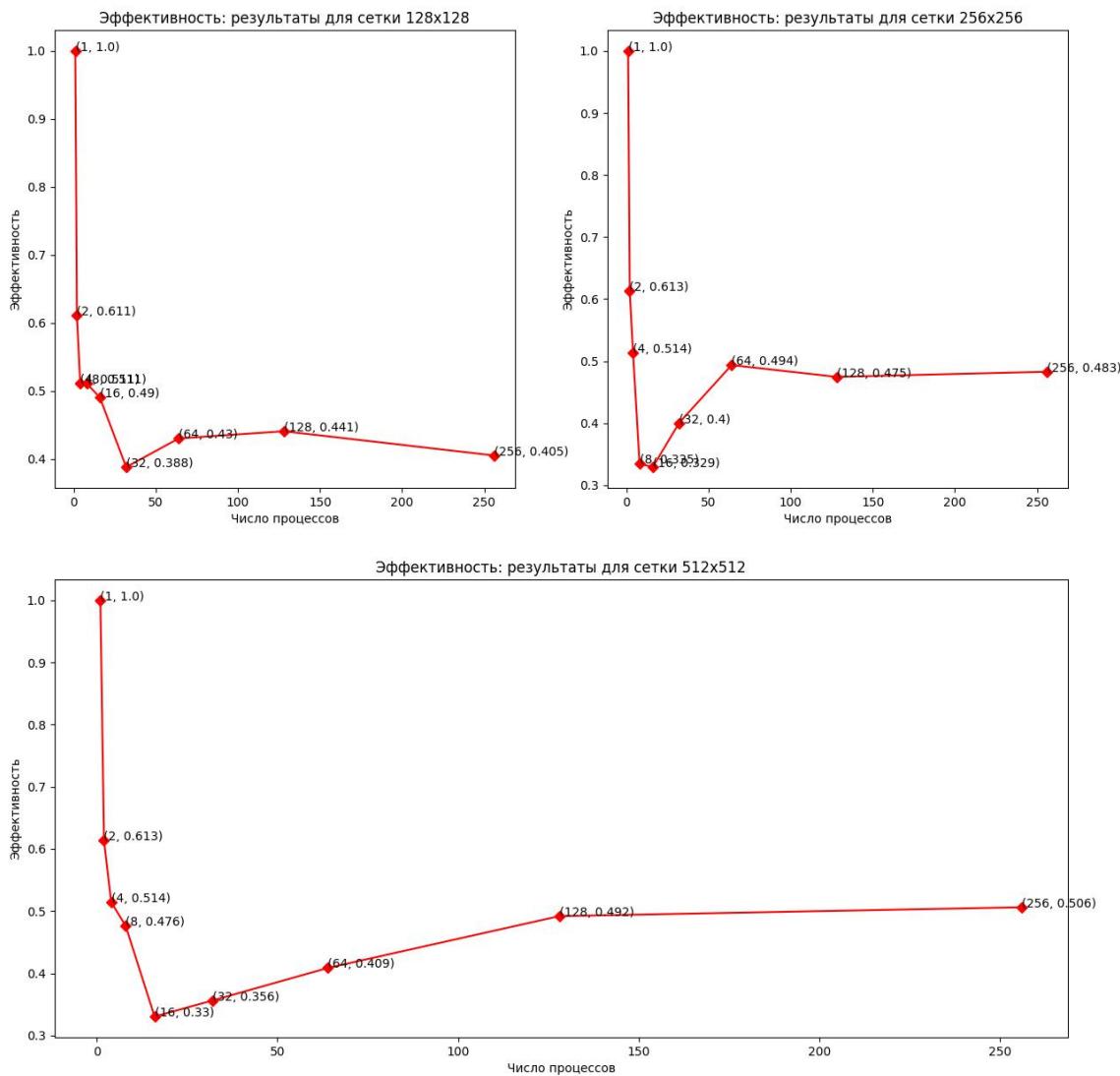


Рис. 7: Временные результаты сортировки для сеток разного размера при  $k = 64$

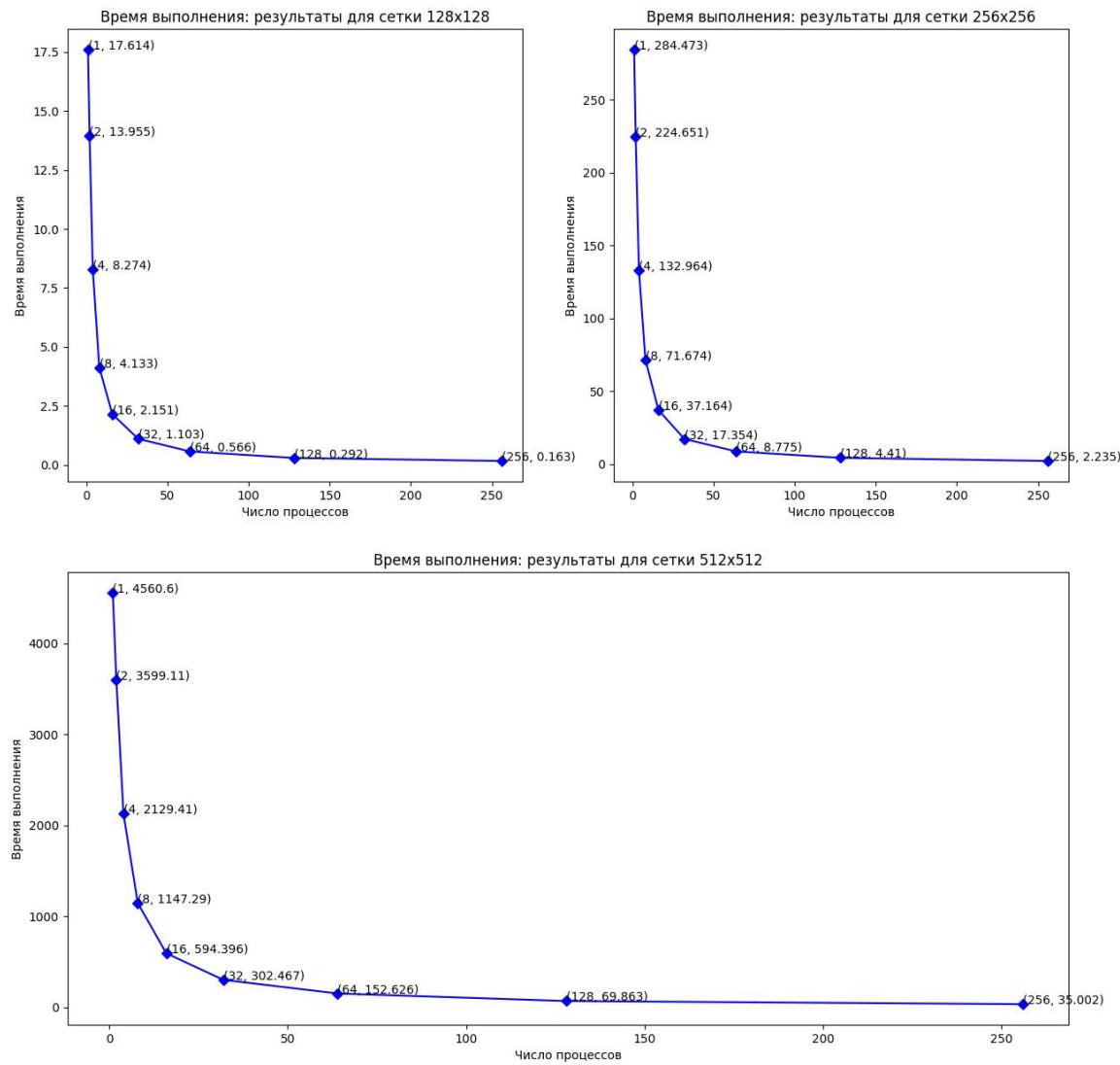


Рис. 8: Ускорение: результаты сортировки для сеток разного размера при  $k = 64$

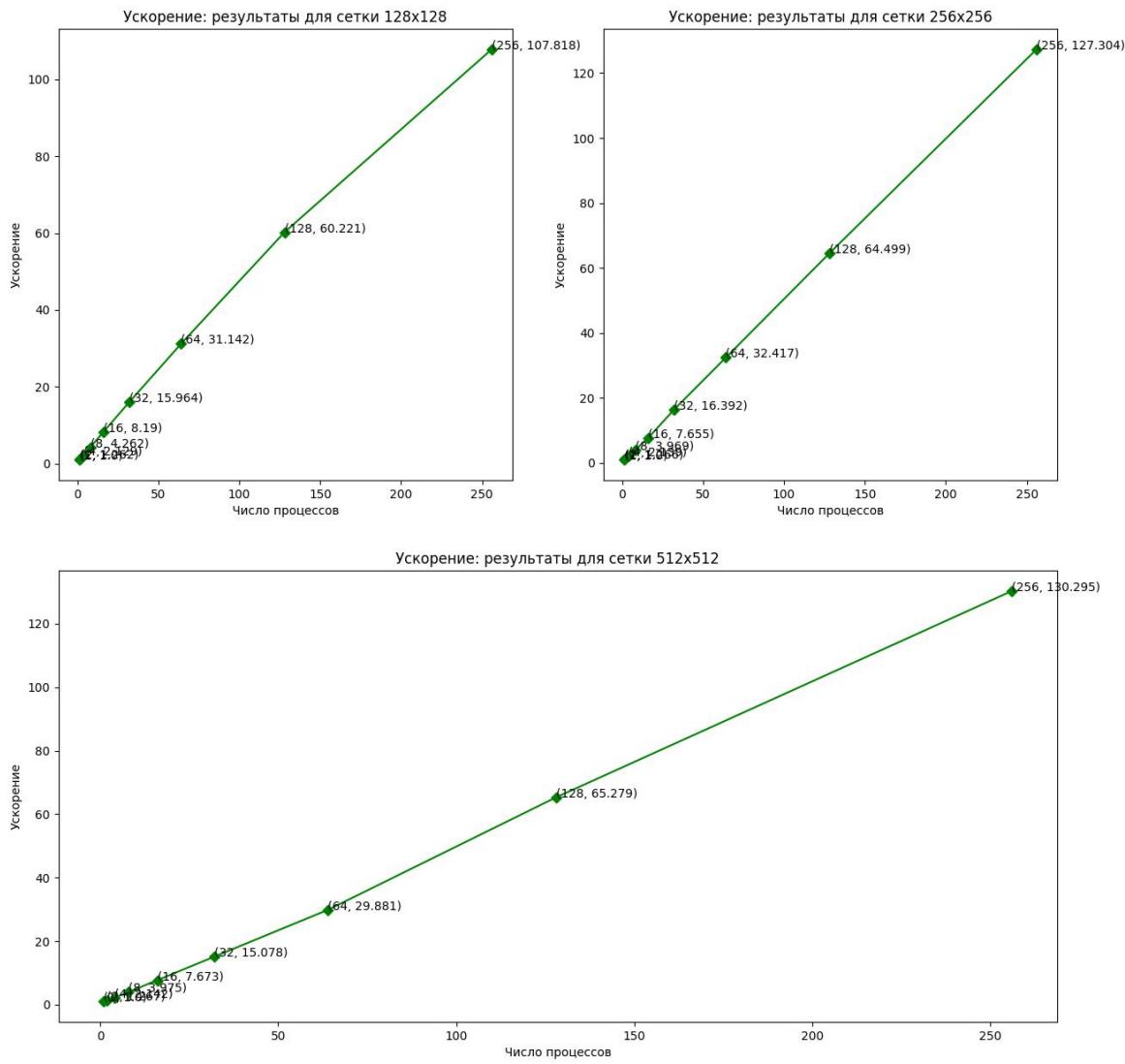
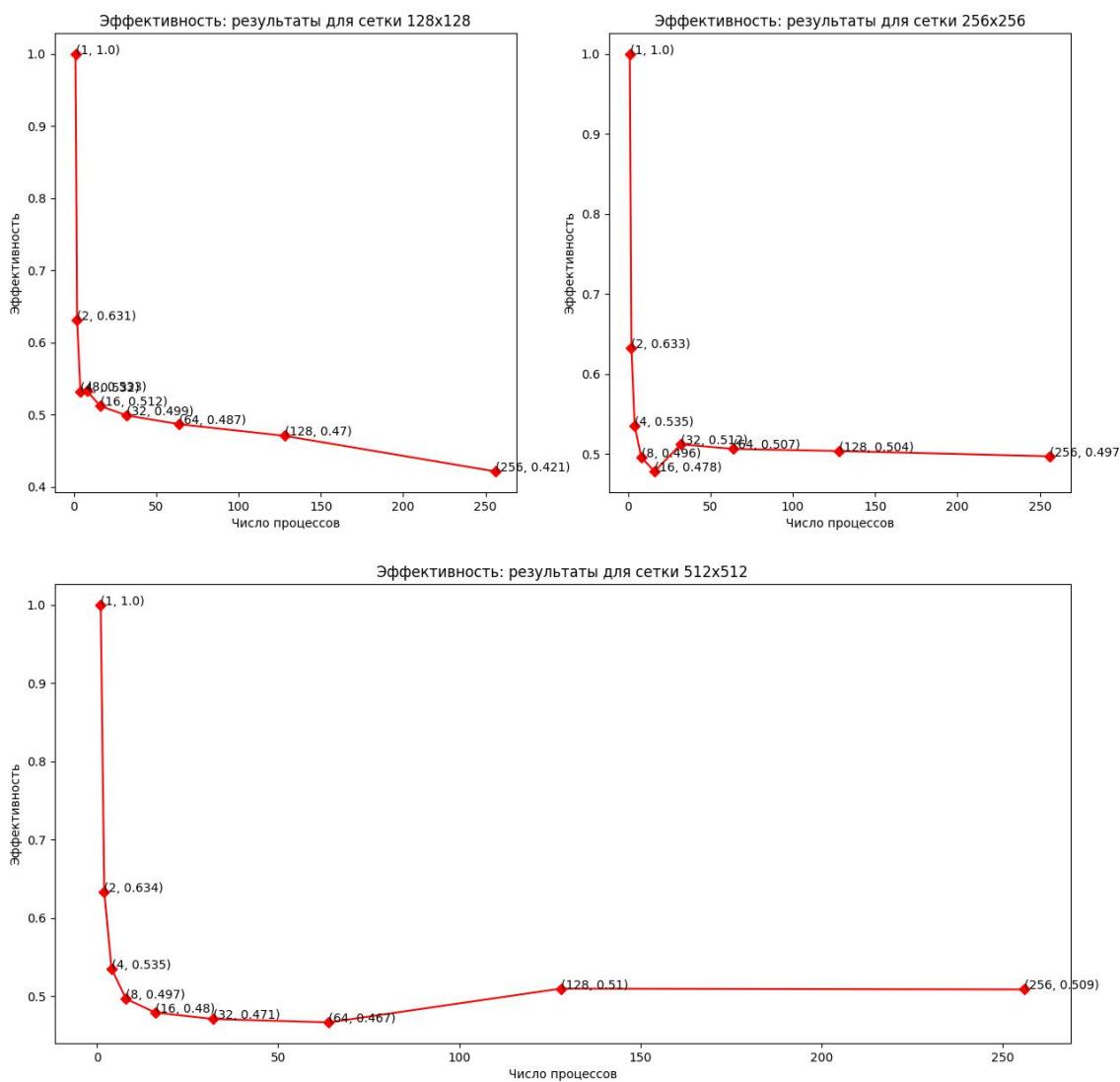


Рис. 9: Эффективность: результаты сортировки для сеток разного размера при  $k = 64$

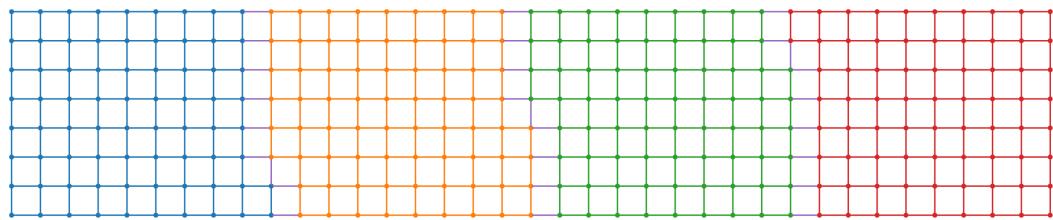


## 5 Анализ полученных результатов

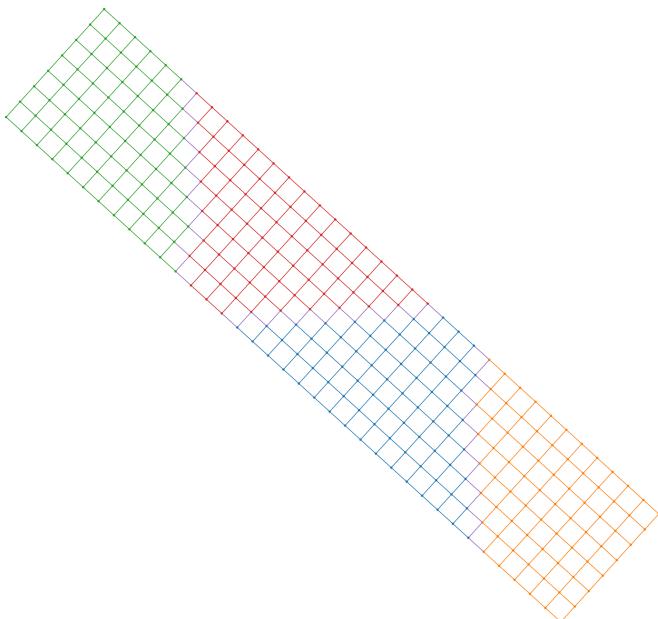
Для проверки результатов был написан скрипт, проверяющий равенство числа вершин в доменах с точностью до одной вершины. Для проверки числа рёбер, подсчитанных в ходе работы программы, подсчитывалось фактическое число

разрезанных рёбер (из полученного выходного файла) и эти числа сравнивались между собой. Кроме того, результаты работы реализованного алгоритма сравнивались с его упрощённой версией, в которой ось сортировки каждый раз меняется на противоположную ( $x$  на  $y$ ,  $y$  на  $x$ ), то есть не производится минимизация числа разрезанных рёбер. Результаты разбиения для трёх сеток размера  $37 \times 8$  при  $k = 4, 7$  показаны на рисунках 10, 11, 12, 13. На рисунках видно, что алгоритм с минимизацией числа разрезанных рёбер в общем случае даёт лучший результат по сравнению с постоянной заменой оси на противоположную.

Рис. 10: Визуализация декомпозиции для разных сеток размера  $37 \times 8$  при  $k = 4$  с минимизацией разрезанных рёбер



(a) 27 разрезанных рёбер

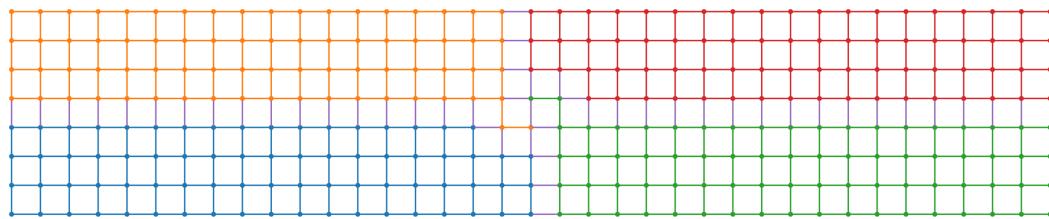


(b) 43 разрезанных ребра

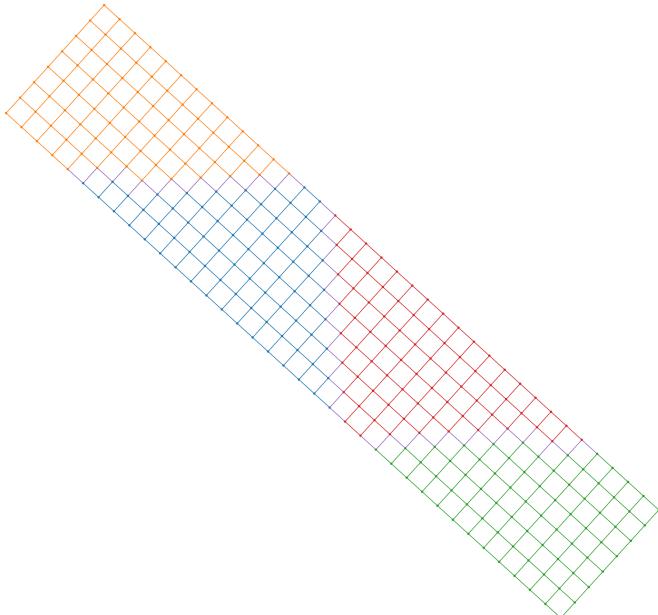


(c) 107 разрезанных рёбер

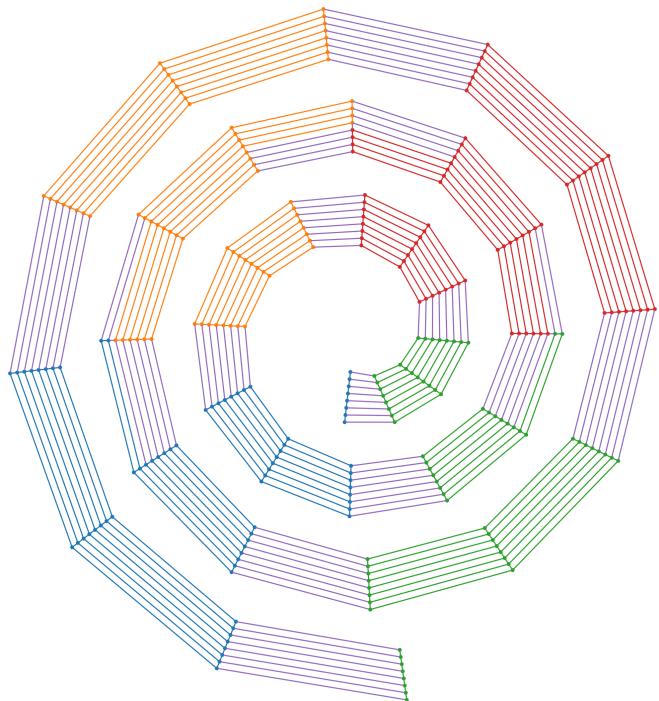
Рис. 11: Визуализация декомпозиции для разных сеток размера  $37 \times 8$  при  $k = 4$  без минимизации разрезанных рёбер



(a) 49 разрезанных рёбер

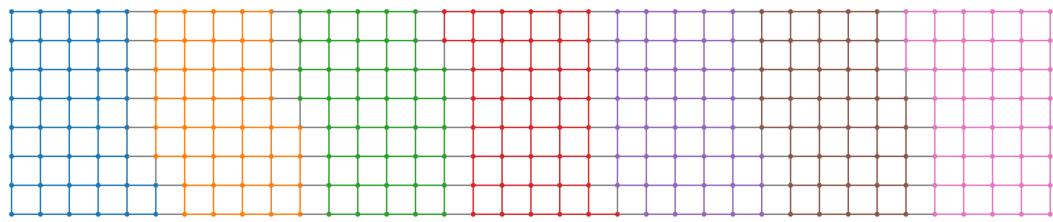


(b) 47 разрезанных рёбер

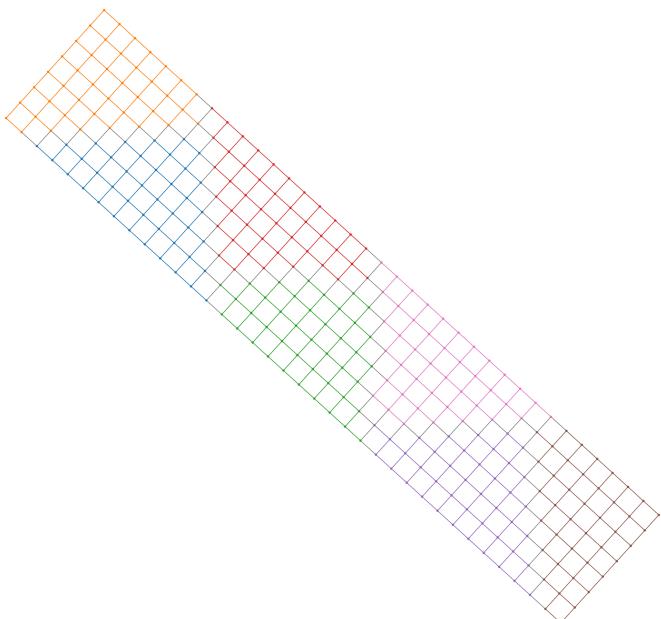


(c) 107 разрезанных рёбер

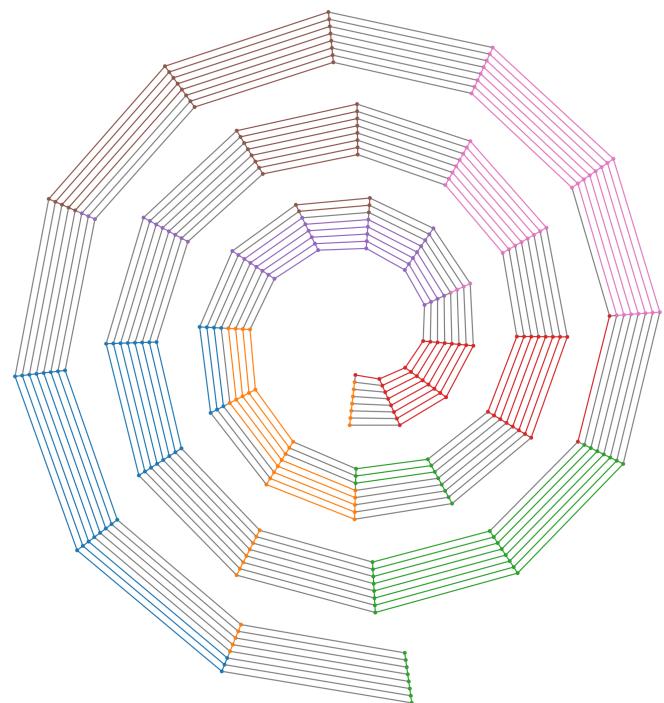
Рис. 12: Визуализация декомпозиции для разных сеток размера  $37 \times 8$  при  $k = 7$  с минимизацией разрезанных рёбер



(a) 54 разрезанных ребра

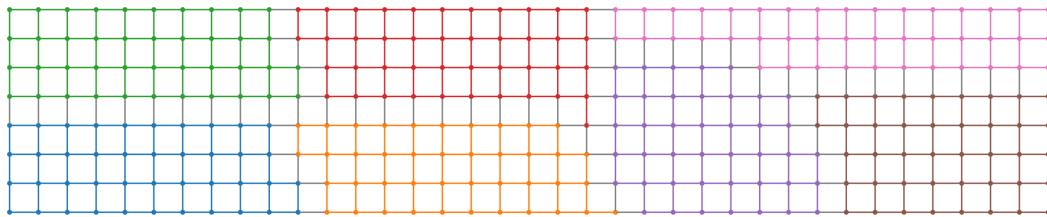


(b) 76 разрезанных рёбер

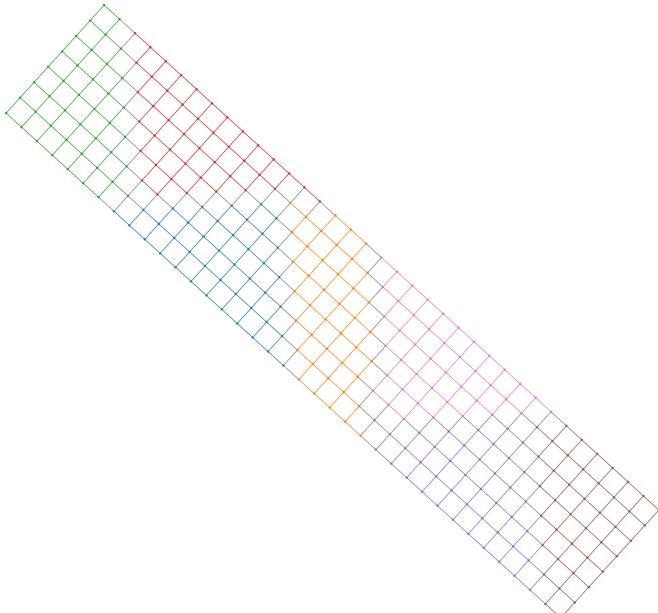


(c) 153 разрезанных ребра

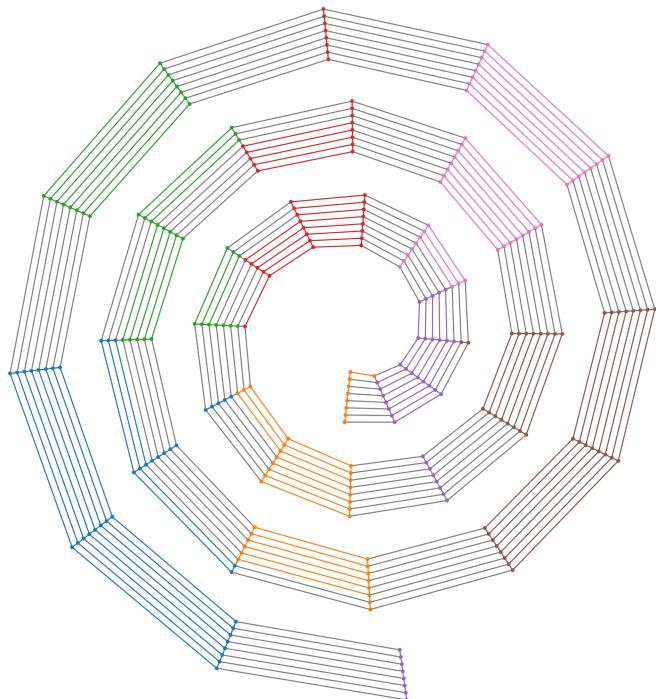
Рис. 13: Визуализация декомпозиции для разных сеток размера  $37 \times 8$  при  $k = 7$  без минимизации разрезанных рёбер



(a) 64 разрезанных ребра



(b) 79 разрезанных рёбер



(c) 159 разрезанных рёбер

## 6 Приложение

Алгоритм декомпозиции реализован на языке C++ в файле *task3.cpp*, сортировка Бэтчера находится в файле *batcher\_sort.cpp*, файл *utils.cpp* содержит вспомогательные функции для генерации массива, вывода результатов, формулы для разных сеток и т.д.

Для расчёта ускорения и эффективности, а также визуализации сеток, написан скрипт *utils.py* на языке Python. Для запуска программы на суперкомпьютере написан *Makefile*.