МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

Разработка Q-эффективных программ для реализации алгоритма сортировки Шелла на общей и распределенной памяти кластерной вычислительной системы

Научный руководитель:
доцент кафедры СП,
Алеева В.Н

Автор работы:
студент группы КЭ-401
Кузнецов Е.К.

Челябинск, 2024 г

Цель и задачи исследования

Цель:

Разработка Q-эффективных программ для реализации алгоритма сортировки Шелла на общей и распределенной памяти кластерной вычислительной системы.

Задачи:

- Разработать Q-эффективную программу для реализации алгоритма сортировки Шелла на общей памяти.
- Разработать Q-эффективную программу для реализации алгоритма сортировки Шелла на распределенной.
- Провести функциональное тестирование разработанных программ.
- Оценить динамические характеристики разработанных программ.

Актуальность исследования

- Широкое распространение параллельных программ.
- Проблема повышения их эффективности.
- Решением является применение концепции Q-детерминанта.
- Q-эффективная реализация алгоритма полностью использует ресурс параллелизма алгоритма.
- В настоящее время для алгоритмов сортировки не применялся метод проектирования Q-эффективных программ.

Метод проектирования Q-эффективных программ

1. Построение Q-детерминанта алгоритма.
2. Описание Q-эффективной реализации алгоритма.
3. Разработка программы, выполняющей Q-эффективную реализацию.

Общее представление Q-детерминанта

- Q-детерминант алгоритма имеет вид:

$$y_i = f_i(i = 1, \dots, n)$$

где y_i – выходные данные алгоритма, f_i – Q-термы, которые описывают вычисление y_i в соответствии с алгоритмом.

- Q-детерминант алгоритма сортировки Шелла состоит из условных Q-термов вида:

$$(\hat{u}_i, \hat{w}_i) = \{(u_j, w_{ij})\}_{j \in \{1, \cdots, l\}}$$

где u_j – безусловные логические Q-термы, w_{ij} – безусловные Q-термы.

https://github.com/Uto4kAqq/ShellSortOpenMPI

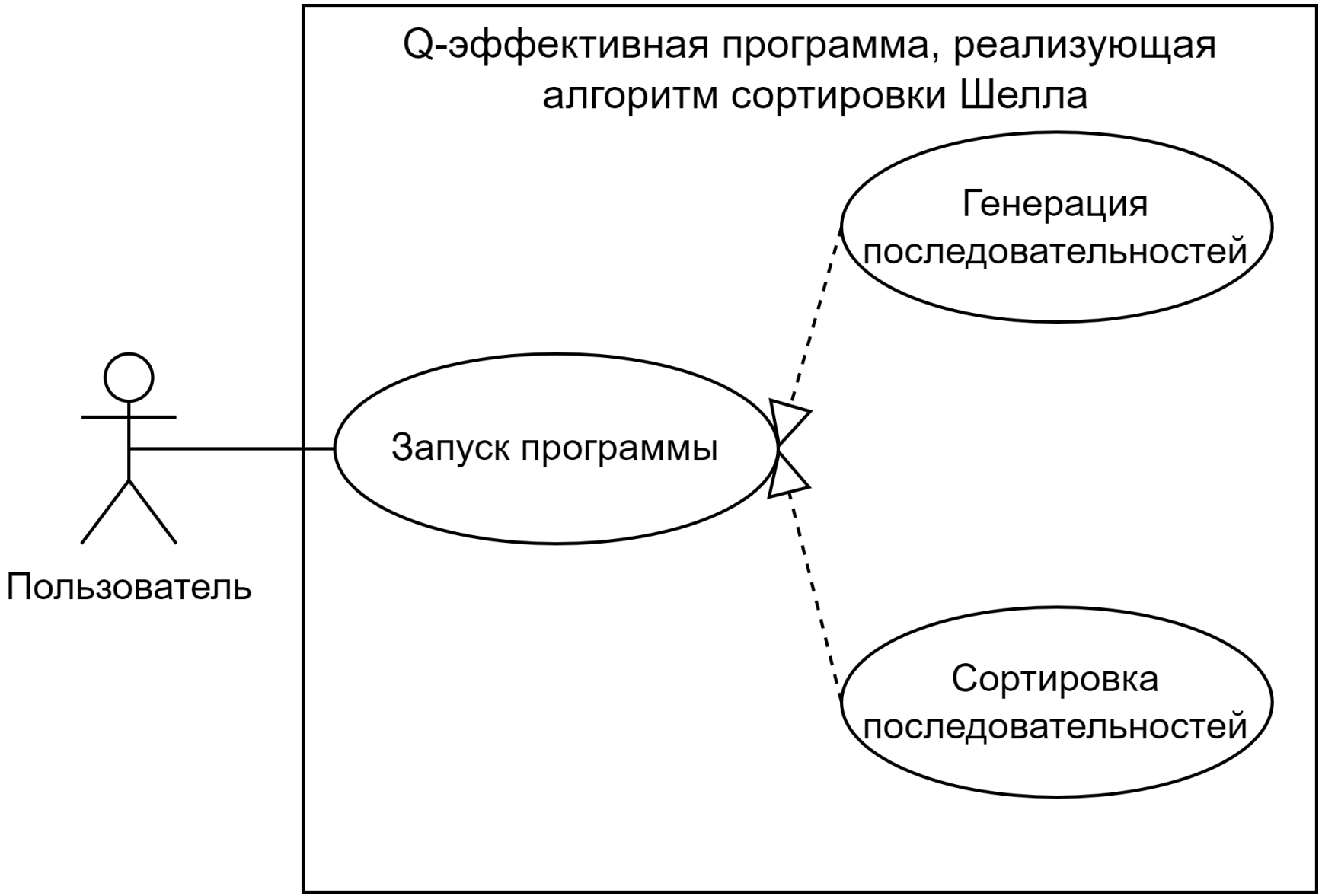
Пример Q-терма алгоритма сортировки Шелла для массива из 4 элементов

A = [A(1), A(2), A(3), A(4)] – исходный массив
B = [B(1), B(2), B(3), B(4)] – результирующий массив

(A(1)<=A(3))&(A(2)<=A(4))&(A(1)<=A(2))&(A(2)>A(3))&(A(1)<=A(3))&
(A(2)<=A(4)) – упрощенное представление логического Q-терма u_2

B = [A(1), A(3), A(2), A(4)] – результат сортировки

Варианты использования



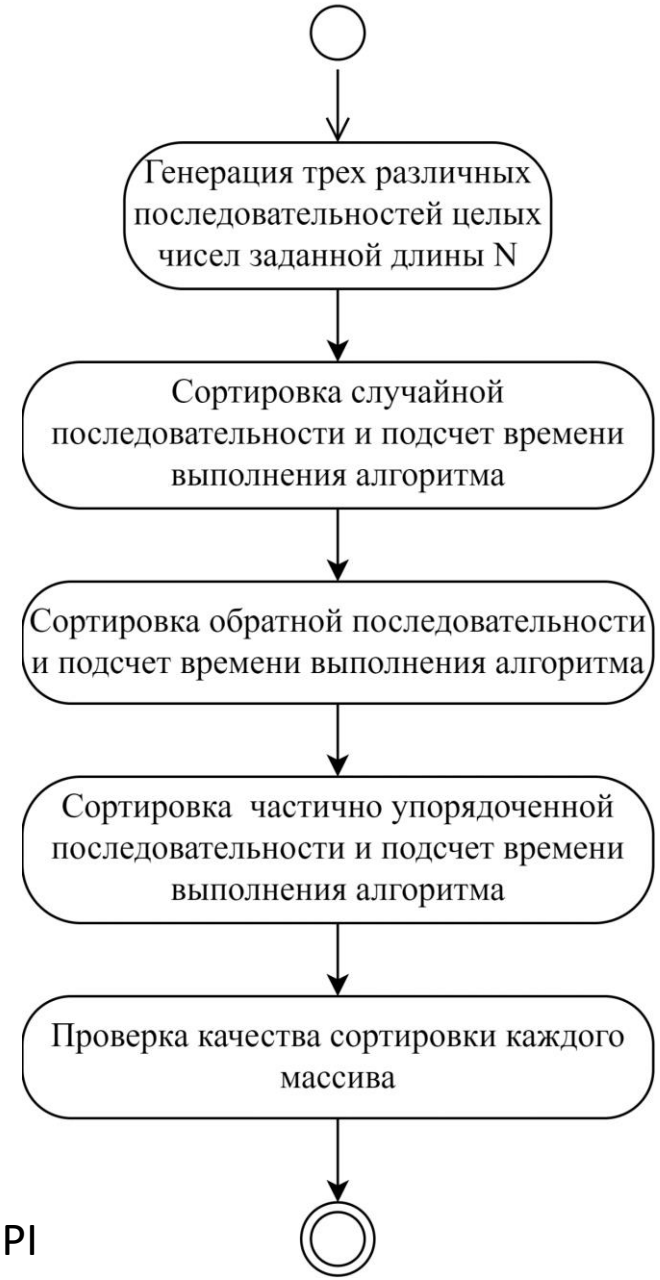
https://github.com/Uto4kAqq/ShellSortOpenMPI

Класс ShellSortOpenMP

ShellSortOpenMP
void shellSort(std::vector<int>& arr) void generateRandomArray(vector<int>& arr, int size) void generateReversedArray(vector<int>& arr, int size) void generatePartiallySortedArray(vector<int>& arr, int size) void checkArray(int array[], int length) int main()

https://github.com/Uto4kAqq/ShellSortOpenMPI

Диаграмма конечных автоматов программы для общей памяти



https://github.com/Uto4kAqq/ShellSortOpenMPI

Класс ShellSortMPI

ShellSortMPI
void shellSort(std::vector<int>& arr) void generateRandomArray(vector<int>& arr, int size) void generateReversedArray(vector<int>& arr, int size) void generatePartiallySortedArray(vector<int>& arr, int size) void checkArray(int array[], int length) void merge(std::vector<int>& left, std::vector<int>& right, std::vector<int>& bars) void mergeBlocks(std::vector<int>& data, int block_size, int num_blocks) int main()

https://github.com/Uto4kAqq/ShellSortOpenMPI

Диаграмма конечных автоматов программы для распределенной памяти

https://github.com/Uto4kAqq/ShellSortOpenMPI

Реализация

Программа для общей памяти

- Язык программирования C++
- Технология OpenMP

Программа для распределенной памяти

- Язык программирования C++
- Технологии OpenMP + MPI

https://github.com/Uto4kAqq/ShellSortOpenMPI

Функциональное тестирование

Количество процессорных ядер	Тип массивов		
	Случайный	Обратный	Частично упорядоченный
	Тест пройден?		
2	Да	Да	Да
3	Да	Да	Да
4	Да	Да	Да
5	Да	Да	Да
6	Да	Да	Да
7	Да	Да	Да
8	Да	Да	Да
9	Да	Да	Да
10	Да	Да	Да
11	Да	Да	Да
12	Да	Да	Да
24	Да	Да	Да
36	Да	Да	Да
48	Да	Да	Да
60	Да	Да	Да
72	Да	Да	Да
84	Да	Да	Да
96	Да	Да	Да
120	Да	Да	Да
132	Да	Да	Да
156	Да	Да	Да
168	Да	Да	Да
180	Да	Да	Да
192	Да	Да	Да
204	Да	Да	Да
216	Да	Да	Да
228	Да	Да	Да
240	Да	Да	Да
252	Да	Да	Да
264	Да	Да	Да
276	Да	Да	Да
288	Да	Да	Да
312	Да	Да	Да
324	Да	Да	Да
336	Да	Да	Да
348	Да	Да	Да
360	Да	Да	Да
372	Да	Да	Да
384	Да	Да	Да

Вычислительные эксперименты

Проводились на суперкомпьютере «Торнадо ЮУрГУ».
Изменяемыми параметрами являются:

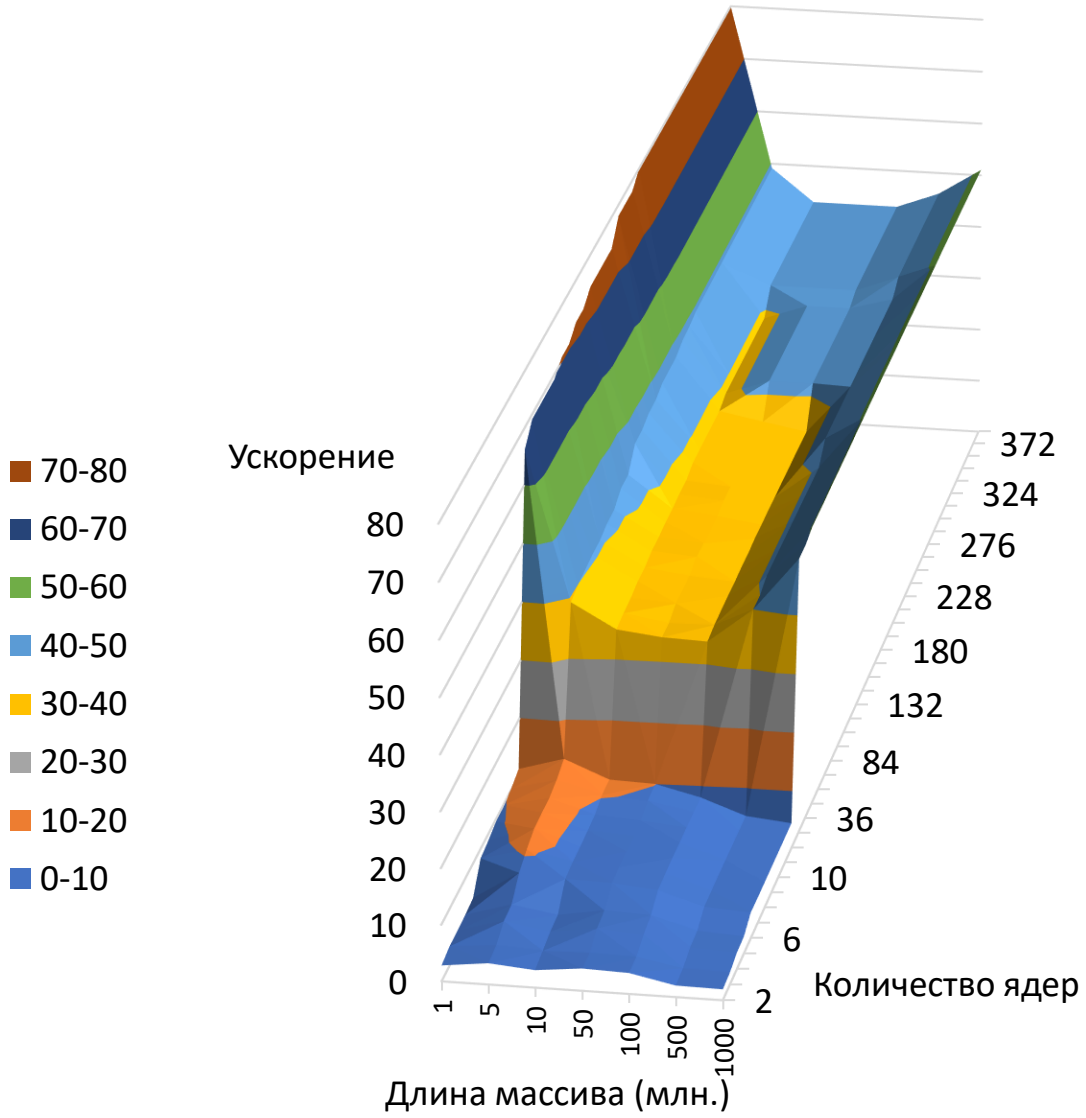
- Размерность сортируемых массивов.
- Для общей памяти количество процессорных ядер узла (от 2 до 12).
- Для распределенной памяти количество узлов (от 2 до максимально доступных пользователям 32).

По результатам вычислительных экспериментов выполнена оценка динамических характеристик:

- Ускорение — отношение времени выполнения параллельной программы ко времени выполнения параллельной программы, выполняемой на одном узле на одном ядре процессора.
- Эффективность — отношение ускорения к числу ядер, необходимых для получения данного ускорения.

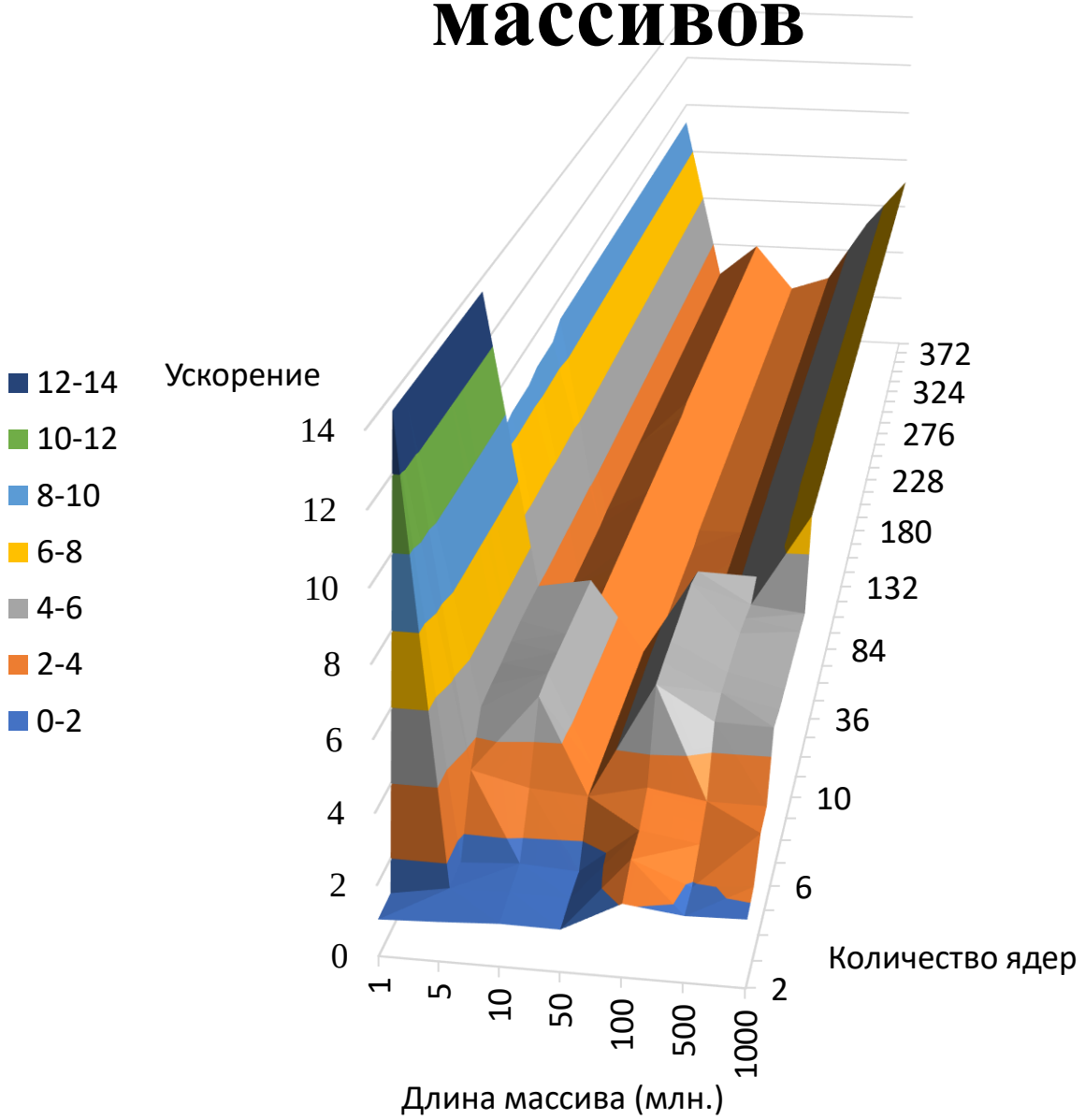
https://github.com/Uto4kAqq/ShellSortOpenMPI

Ускорение Q-эффективных программ для случайных массивов



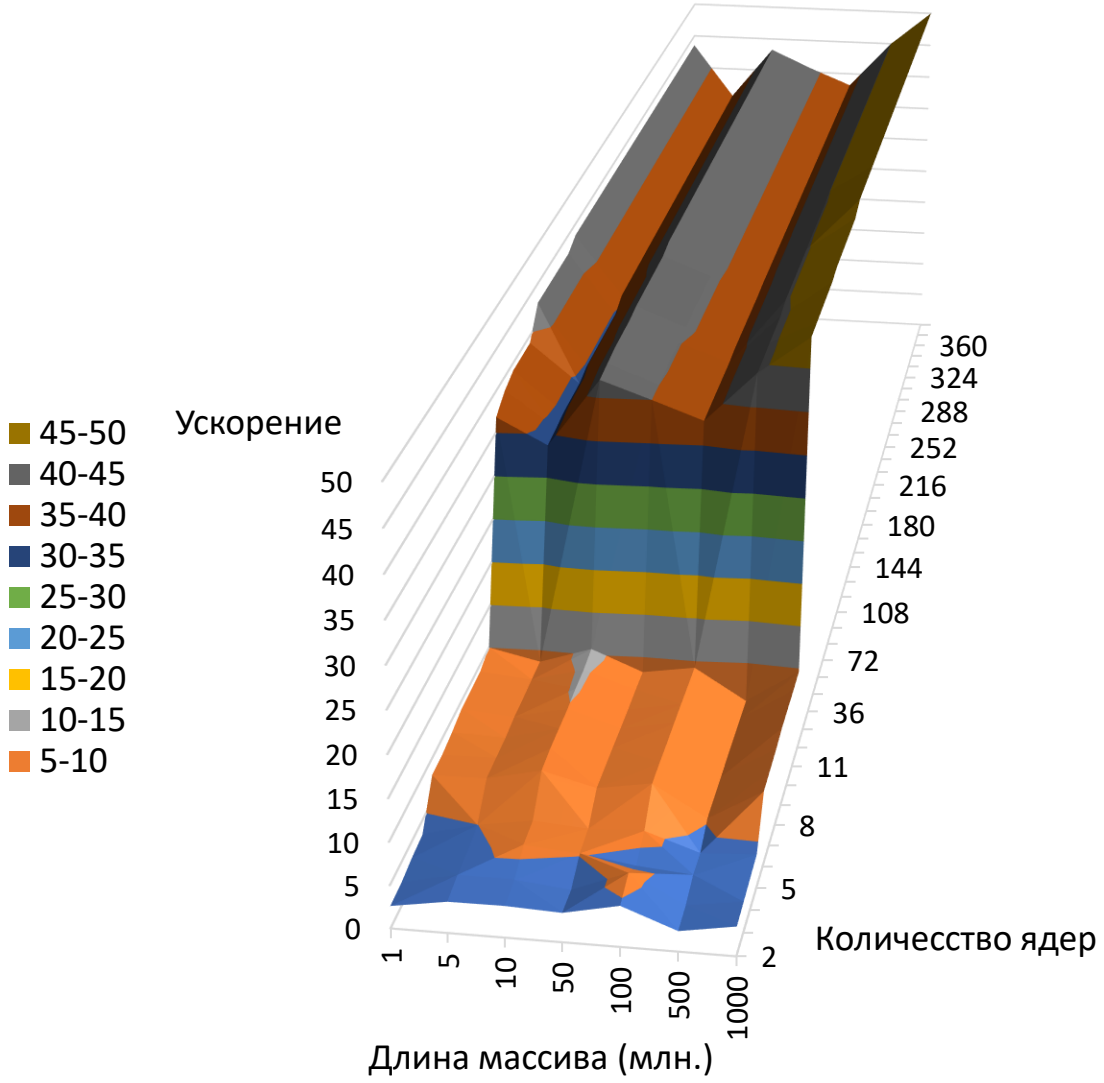
https://github.com/Uto4kAqq/ShellSortOpenMPI

Ускорение Q-эффективных программ для обратных массивов



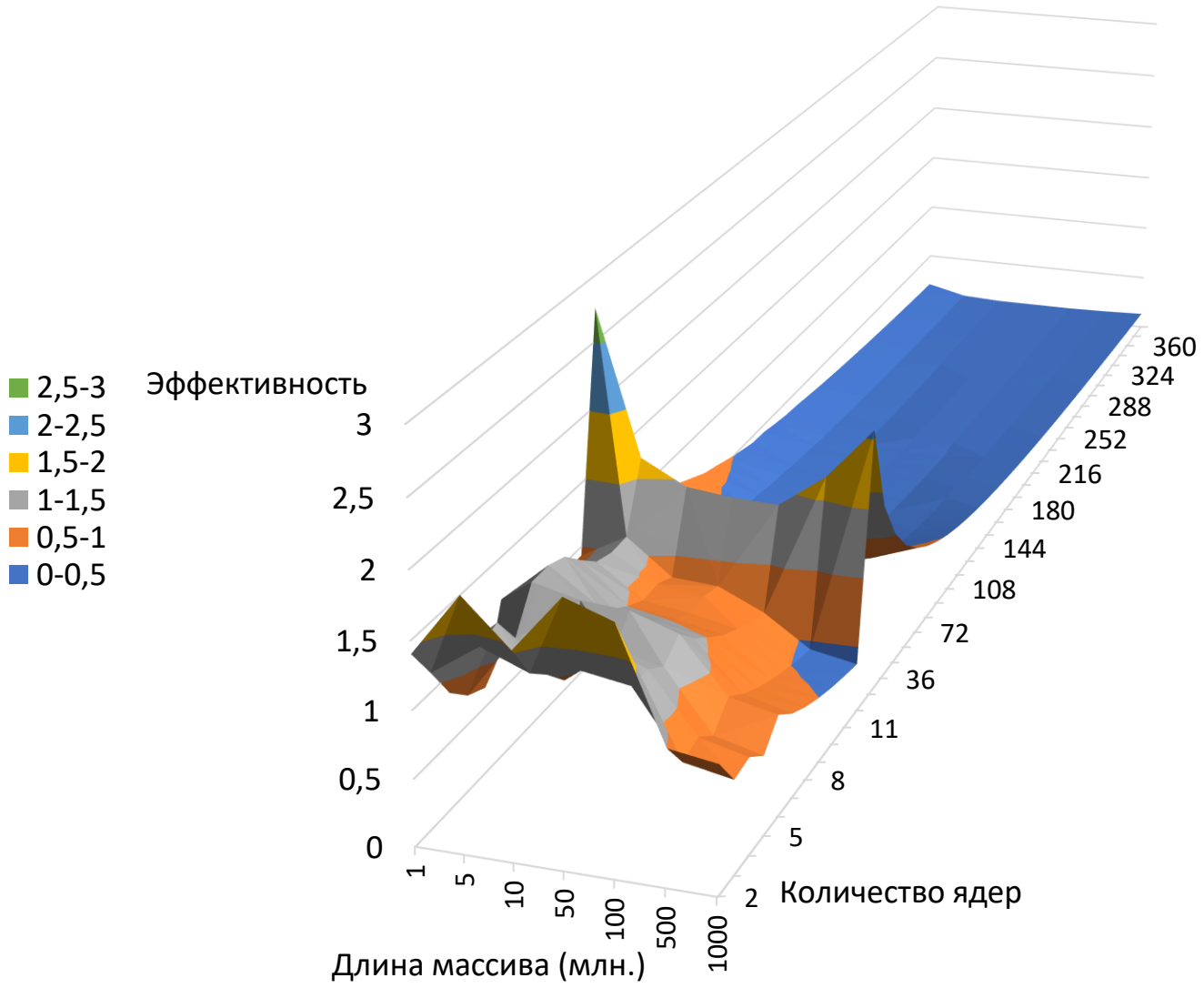
https://github.com/Uto4kAqq/ShellSortOpenMPI

Ускорение Q-эффективных программ для частично упорядоченных массивов



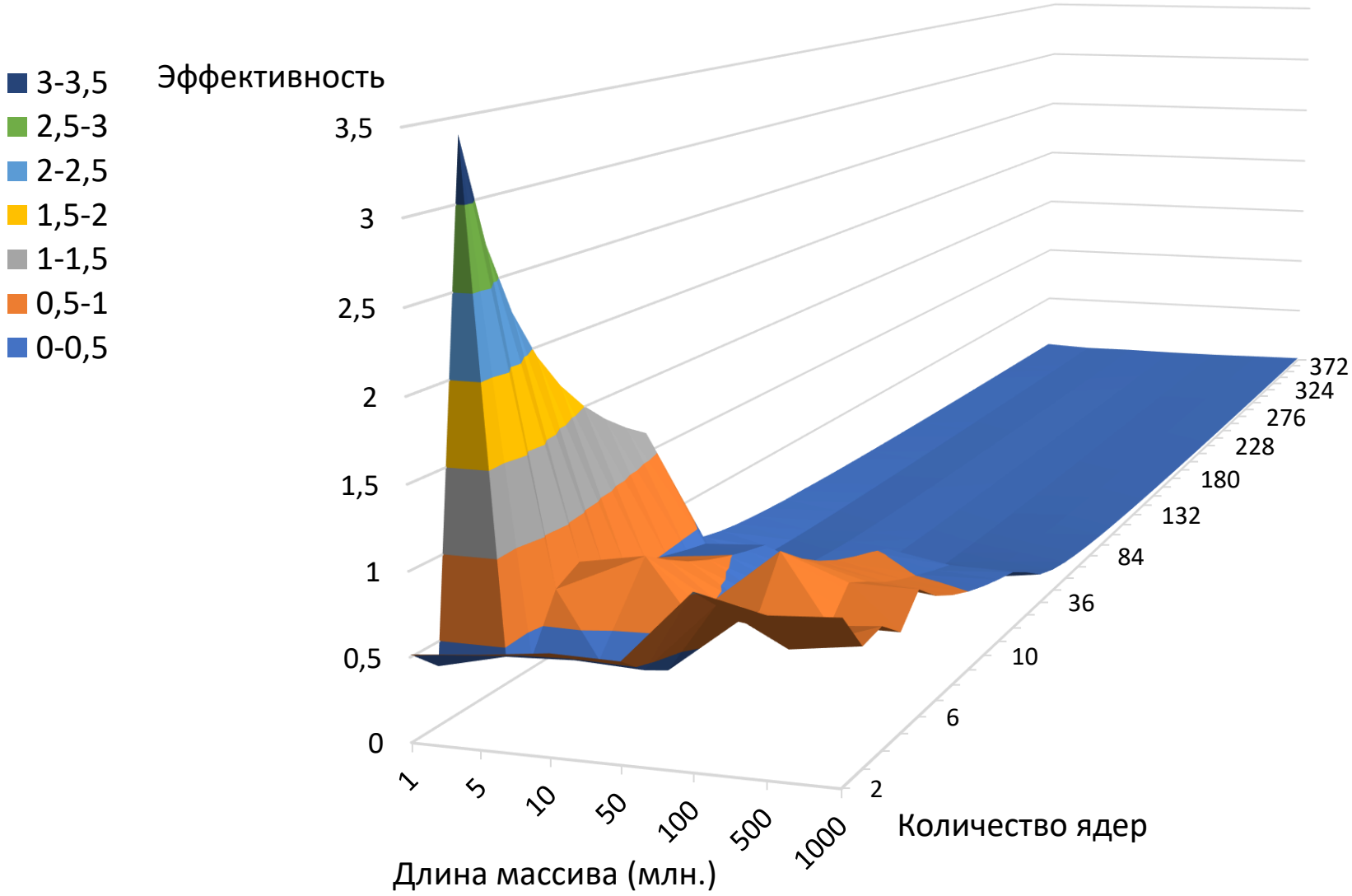
https://github.com/Uto4kAqq/ShellSortOpenMPI

Эффективность Q-эффективных программ для случайных массивов



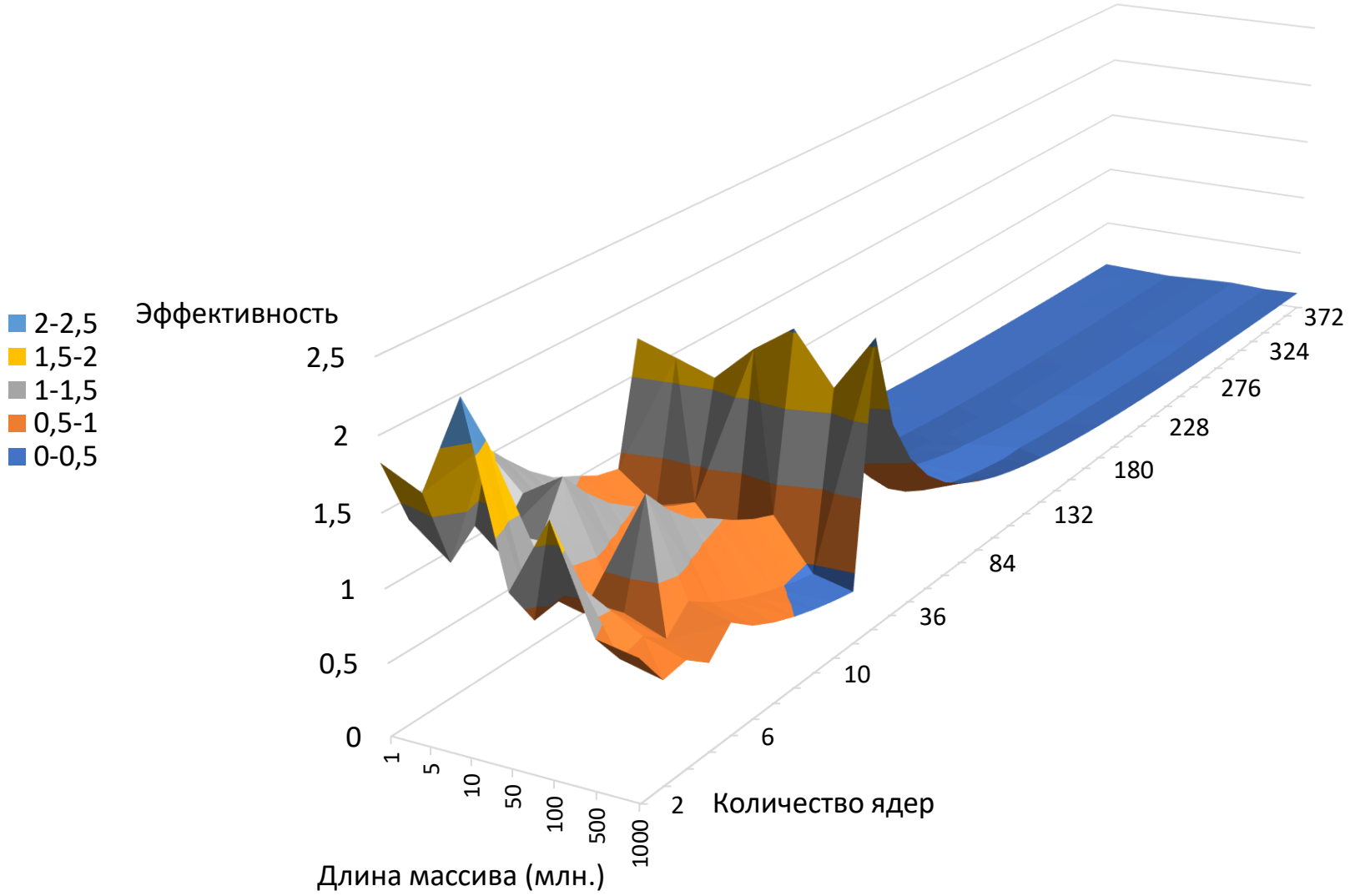
https://github.com/Uto4kAqq/ShellSortOpenMPI

Эффективность Q-эффективных программ для обратных массивов



https://github.com/Uto4kAqq/ShellSortOpenMPI

Эффективность Q-эффективных программ для частично упорядоченных массивов



https://github.com/Uto4kAqq/ShellSortOpenMPI

Основные результаты

1. Разработаны Q-эффективные программы для реализации алгоритма сортировки Шелла на общей и распределенной памяти.
2. Проведено функциональное тестирование разработанных программ.
3. Проведен анализ динамических характеристик разработанных программ.

https://github.com/Uto4kAqq/ShellSortOpenMPI