Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Исследование динамических топологий

Отчет по лабораторной работе №8 дисциплины
«Высокопроизводительные вычислительные комплексы»

Выполнил студент группы ИВТ-41_____/Крючков И. С./
Проверил_____/Мельцов В. Ю./

Киров 2023

1. Цель работы

Получение практических навыков выполнения параллельных программ на кластере.

2. Ход работы

1) Задание

В качестве вычислительной задачи для распараллеливания был выбран алгоритм блочного умножения матриц методом Кэннона.

2) Словесное описание алгоритма

При блочном способе разделения данных исходные матрицы A, B и результирующая матрица C представляются в виде набора блоков. Исходные матрицы должны квадратными и иметь размерность, кратную $\sqrt{p}$, где p – количество процессоров

При таком разбиении данных для определения базовых подзадач за основу берутся вычисления, выполняемые над блоками. А базовой подзадачей является процедура вычисления всех элементов одного из блоков результирующей матрицы. Начальное расположение блоков в алгоритме Кэннона подбирается таким образом, чтобы блоки в подзадачах могли бы быть перемножены без каких-либо дополнительных передач данных.

Сам алгоритм состоит из двух больших операций — это инициализация матриц, то есть подготовка к умножению, и основной цикл алгоритма - умножение блоков и получение матрицы результата.

Этап инициализации алгоритма Кэннона включает выполнение следующих операций передач данных:

- в каждую подзадачу (i, j) передаются блоки A_{ij}, B_{ij};
- для каждой строки I решетки подзадач блоки матрицы A сдвигаются на (i-1) позицию влево.
- Для каждого столбца j решетки подзадач блоки матрицы B сдвигаются на (j-1) позиций вверх.

В результате такого начального распределения в каждой базовой подзадаче будут располагаться блоки, которые могут быть перемножены без дополнительных операций передачи данных. После выполнения операции блочного умножения каждый блок матрицы A должен быть передан левому соседу, а каждый блок матрицы B – верхнему соседу в двумерной решетке процессоров. После выполнения $\sqrt{p}$ итераций получается матрица Cij на каждом процессорном элементе.

Сложность алгоритма – $O(n^3)$

Степень параллелизма n^2

3. Результаты выполнения

В ходе выполнения лабораторной работы, написанная программа запускалась на различном числе ядер и для различных размерностей матриц.

Результаты экспериментов представлены в таблице 1.

Количество ядер, шт	Размер исходных матриц			
	960	1920	2880	3840
1	1,56	30,00	193,00	560,00
4	0,42	3,36	14,50	64,20
9	0,26	2,09	8,70	56,50
16	0,24	1,50	4,80	16,20

На рисунках 1-3 представлены графики зависимостей:

- рисунок 1 – график зависимости времени выполнения от количества ядер;
- рисунок 2 – график зависимости времени выполнения от объема данных;
- рисунок 3 – график зависимости времени выполнения при равномерном увеличении числа ядер и объема данных.

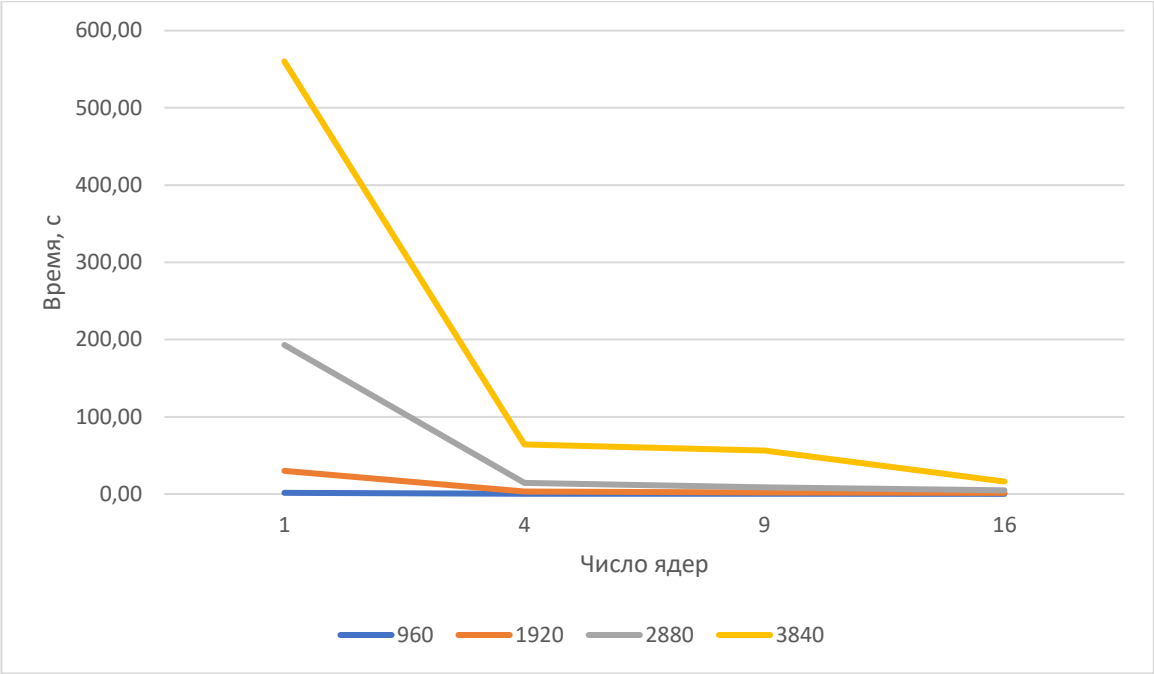

Рисунок 1 – Зависимость времени вычисления от числа ядер

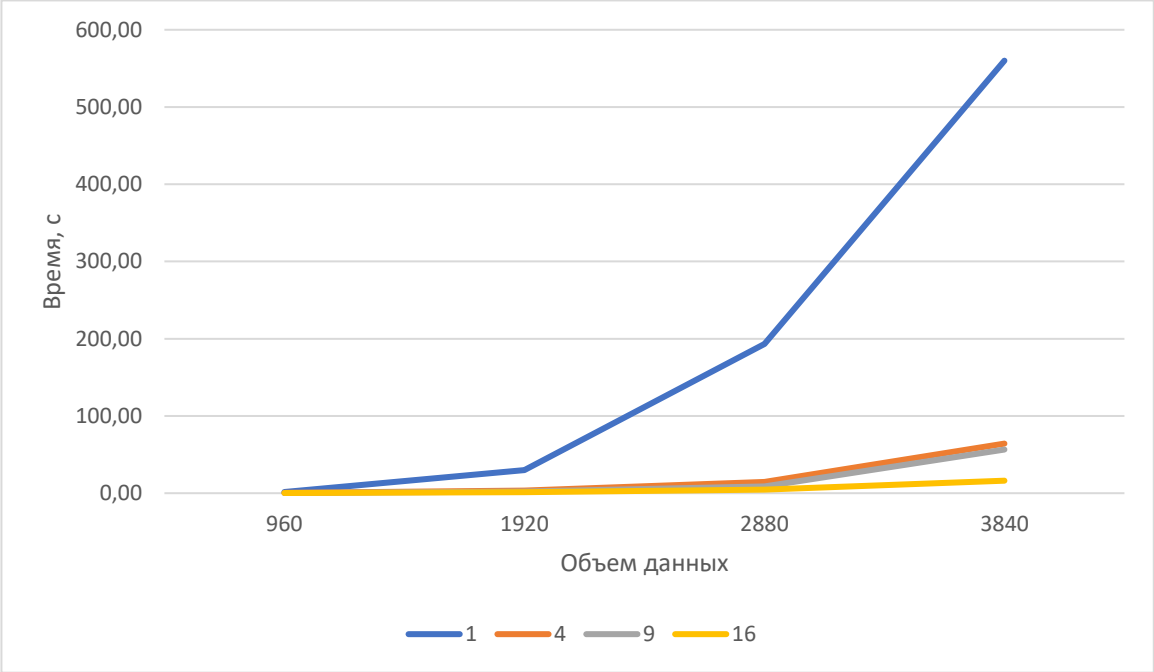

Рисунок 2 – Зависимость времени вычисления от объема данных

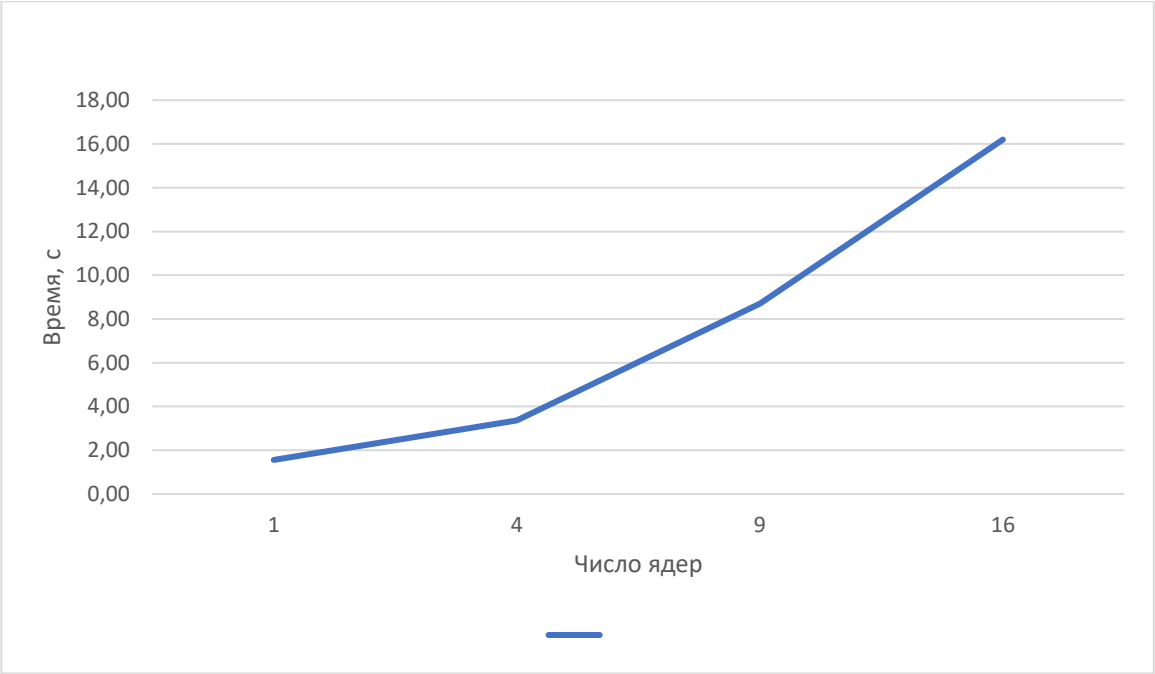

Рисунок 3 – Зависимость времени при одновременном увеличении числа ядер и объема данных

Вывод

При увеличении числа ядер и одинаковом объеме данных время решения задачи уменьшается. Наиболее значительное ускорения произошло при увеличении количества ядер с 1 до 4, при количестве ядер больше 4 эффективность параллельных вычислений снижается из-за накладных расходов на обмен данными.

При увеличении количества данных и неизменном количестве ядер время решения увеличивается сильнее при малом количестве ядер и слабо при большом количестве ядер. Разница в интенсивности изменения времени связана с тем, что при малом количестве ядер время расчёта в большей степени состоит из времени, затрачиваемом на решение задачи (операцию умножения) и в меньшей – на пересылку данных, поэтому оно так сильно зависит от размерности данных. В случае же большого количества ядер ситуация противоположная, и время меньше зависит от размерности данных, нежели от времени, затрачиваемого на их пересылку.

При пропорциональном увеличении количества ядер и объема исходных данных время решения задачи на практике увеличивается. Это можно объяснить тем, что время, затрачиваемое на пересылку данных между ядрами одного блейда значительно меньше, чем время, которое затрачивается на пересылку данных между блейдами.