МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Параллельные алгоритмы»

Тема: Разработка потокобезопасных структур данных

Студент гр. 3341 _____ Романов А. К.

Преподаватель _____ Сергеева Е. И.

Санкт-Петербург

2025

Цель работы.

Реализовать алгоритмы из предложенных. При реализации допустимо использование фреймворков, предоставляющих шаблоны для параллельных алгоритмов (в этом случае выбор фреймворка должен быть обоснован на защите).

Задание.

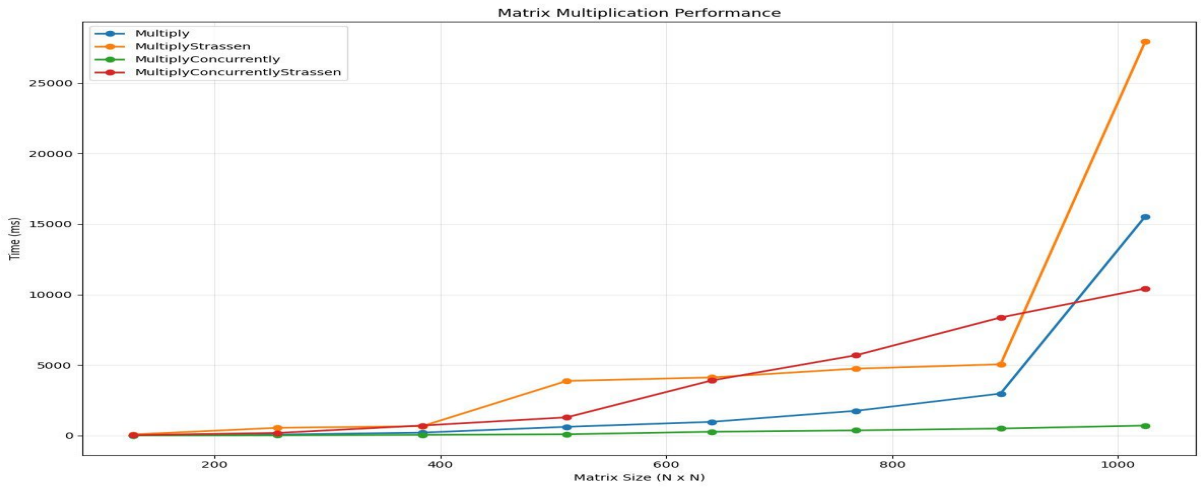
Параллельное "быстрое" умножение матриц (алгоритм Штрассена). Выполнить тестирование корректности вычислений и сравнение производительности с лр1.

Параллельная реализация алгоритма сортировки. Алгоритм сортировки выбрать самостоятельно, сложность алгоритма не хуже O(n log n). Должно быть проведено тестирование корректности результатов сортировки и сравнение с однопоточной реализацией того же алгоритма.

Выполнение работы.

Параллельное умножение (алгоритм Штрассена)

Для реализации используется ThreadPool из лабораторной работы №1. В основе также лежит алгоритм блочного умножения матриц. Вычисление самих блоков заменим на алгоритм Штрассена. По графмику (рис. 1) видно, что алгоритм работает медленнее, это возникает из-за накладных расходов на выделение памяти и рекурсию.



Параллельная сортировка слиянием

1 подход:

Воспользуемся ThreadPool аналогично предыдущей задаче. Возникает следующая проблема: т.к. алгоритм рекурсивный, то просто добавлять задачи в пул не получится, потому что нужно установить зависимость между ними. Для этого воспользуемся DAG.

Введем следующий инвариант:

1) если задача пораждает новую, то она является вызовом mergeSort
2) если задача задача является исполняемой, то она является вызовом merge или вызовом mergeSort от подмассива длинны 1.

Для поддержания структуры DAG скажем, что у каждой задачи есть родитель, очевидно, что DAG будет представлять собой бинарное дерево. Введем у каждой счётчик её потомков, которые выполнили свою работу. Изначально он равен 2, если задача является пораждающей. Как только потомки порождены, задача родитель удаляется из пула и считается выполненной с точки зрения пула. Когда потомок исполняется вызывается callback, в котором он понижает счётчик родителя на 1. Если счётчик доходит до 0, то задача родитель порождается заново своим же потомком.

К сожалению в таком подходе время затраченное на накладные расходы слишком большое, поэтому он не является практичным.

2 подход:

Обычный merge sort с генерацией нового потока на каждом шаге рекурсии. Добавим ограничение на кол-во генерируемых потоков, чтобы избежать гонки за ресурсы.

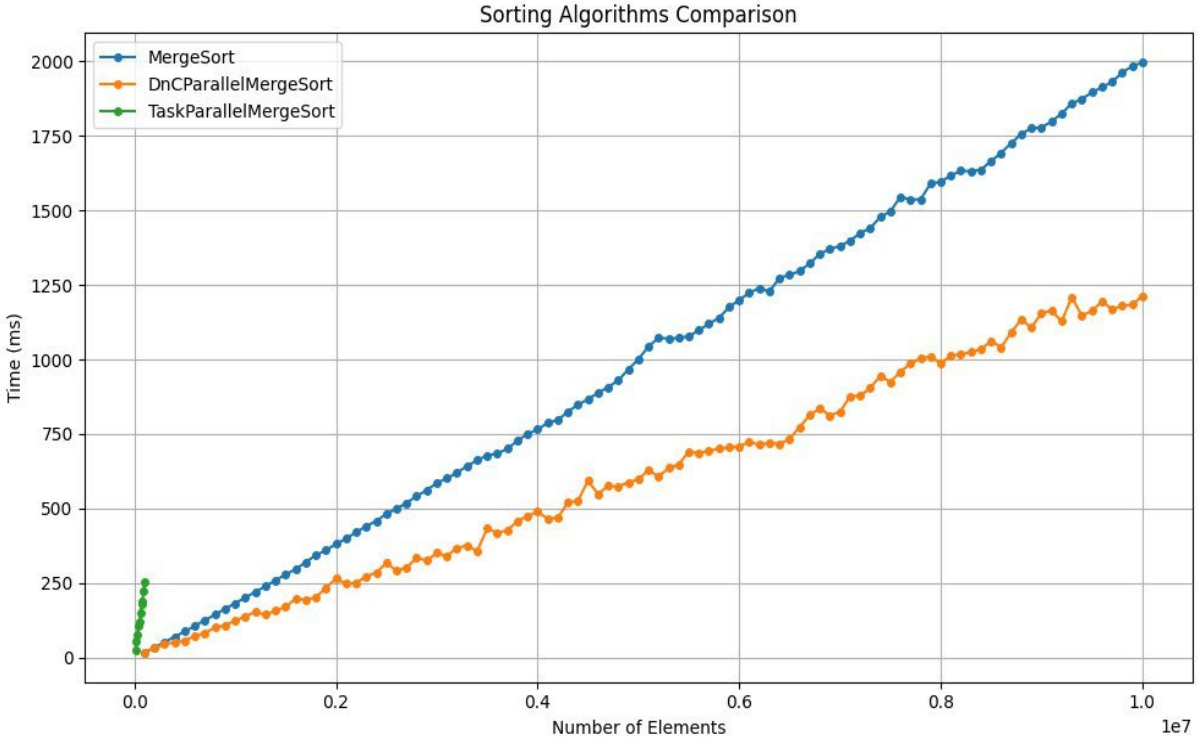

Sorting Algorithms Comparison
MergeSort
DnCParallelMergeSort
TaskParallelMergeSort
Time (ms)
2000
1750
1500
1250
1000
750
500
250
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7
Number of Elements

Рис. 2

Выводы.

В результате выполнения данной лабораторной работы был разработан и протестирован параллельный алгоритм сортировки слиянием.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ