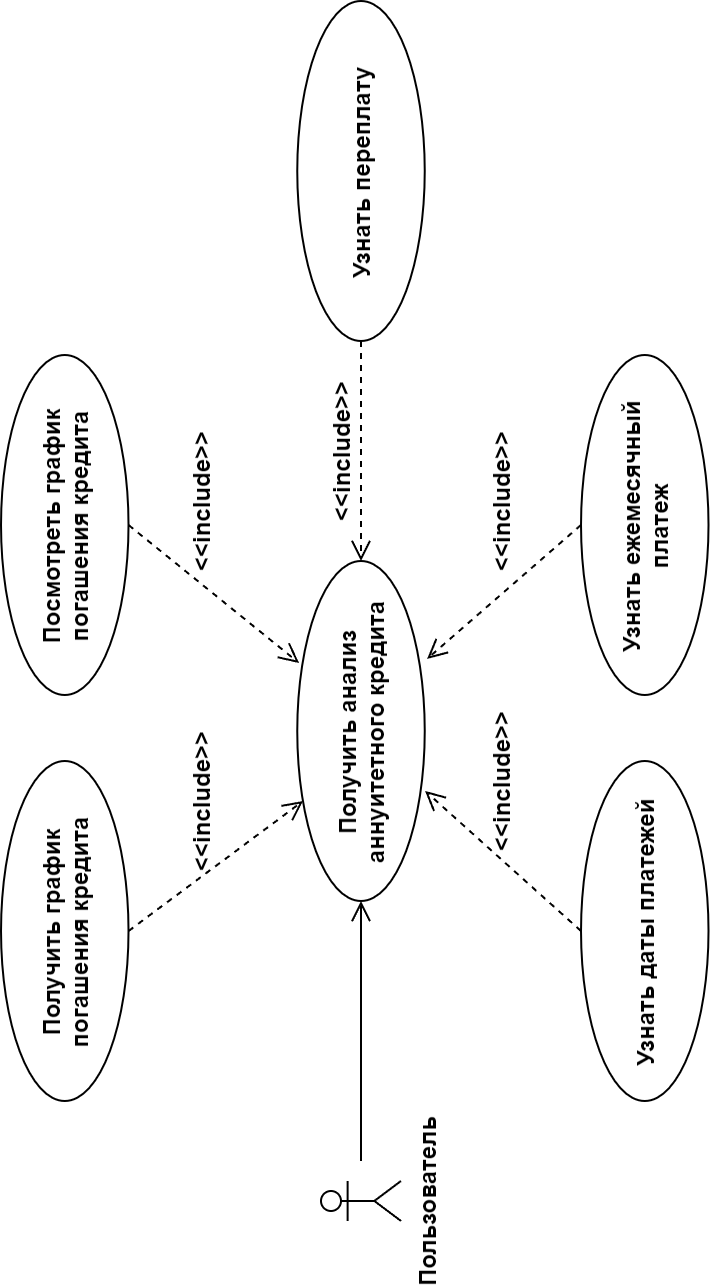
Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Лабораторная работа №8**

по курсу «Высокопроизводительные вычислительные комплексы»

Выполнил студент группы ИВТб-41 /Категов А. Д./ Проверил преподаватель /Мельцов В. Ю./

Киров 2024

1. Цель работы

Получение практических навыков написания параллельных программ для кластера и работы с клатером.

1. Ход работы
2. Задание

В качестве задачи был выбран алгоритм параллельного умножения матриц SUMMA.

1. Словесное описание алгоритма

Алгоритм SUMMA (Scalable Universal Matrix Multiplication Algorithm) параллельный метод умножения матриц. Исходные матрицы A и B, а также результирующая матрица C делятся на блоки, которые распределяются между р процессами.

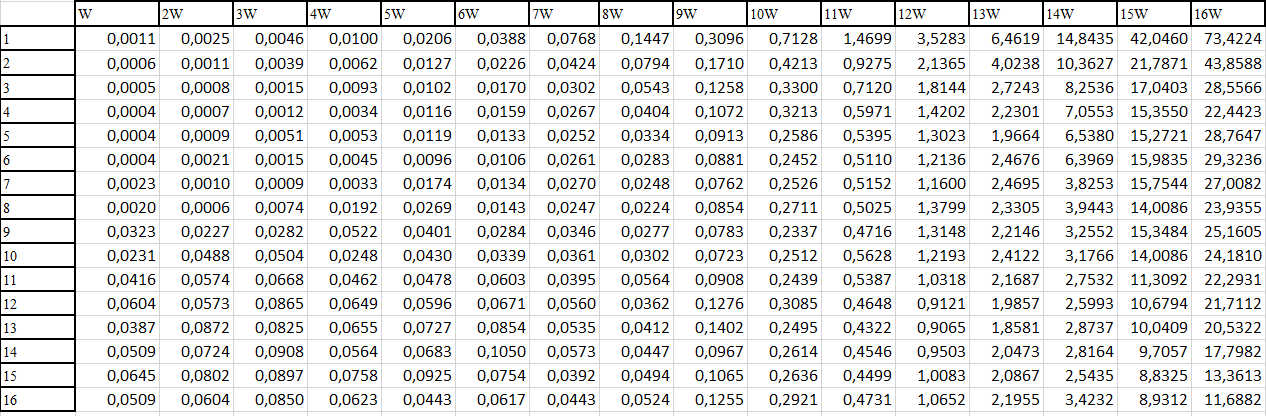
Сложность алгоритма: O(n3)

Алгоритм SUMMA отличается от других методов, таких как алгоритм Кэннона, своей универсальностью и возможностью масштабирования на произвольное количество процессоров.

Основные этапы алгоритма SUMMA:

1. Инициализация:
   * Каждый процесс получает блоки матриц A и B.
   * Процессы подготавливают необходимые буферы для хранения промежуточных данных.
2. Вычисление:
   * Шаг передачи данных:
     + В каждой итерации блоки из строки текущей подматрицы A передаются всем процессам в той же строке решетки.
     + Блоки из столбца текущей подматрицы B передаются всем процессам в том же столбце решетки.
   * Шаг вычислений:
     + Каждый процесс выполняет умножение переданных блоков A и B и добавляет результат в локальный блок матрицы C.
3. Сбор результатов:  
   После завершения всех итераций локальные блоки результирующей матрицы C объединяются для формирования итоговой матрицы.
4. Результаты выполнения

В ходе выполнения лабораторной работы, написанная программа запускалась на различном числе ядер и для различных размерностей матриц. Количество ядер увеличивалось от 1 до 16. Количество элементов в матрицах равномерно увеличивалось от 10000 до 9000000.

Таблица 1 - Результаты экспериментов.

На рисунках 1-5 представлены графики зависимостей:

* рисунок 1 – график зависимости времени выполнения от количества ядер;
* рисунок 2 – график зависимости времени выполнения от количества ядер, логарифмическая шкала;
* рисунок 3 – график зависимости времени выполнения от объема данных;
* рисунок 4 – график зависимости времени выполнения от объема данных, логарифмическая шкала;
* рисунок 5 – график зависимости времени выполнения при равномерном увеличении числа ядер и объема данных.

1. Выводы

Вывод 1

На рисунке 1 видно, что зависимость времени вычисления от числа ядер не является линейной. Чем больше ядер, тем меньше прирост ускорения. Это объясняется тем, что n ядер, на практике, никогда не дадут ускорение в n раз. При увеличении числа ядер увеличиваются затраты на межъядерную коммуникацию и организацию параллельных вычислений.

При переходе от 4 к 5 ядрам наблюдается падение производительности. Особенно заметно на большом объеме входных данных. Это объясняется подключением второго процессора и необходимостью обмена сообщениями между 2 разными процессорами.

При переходе от 8 к 9 ядрам также наблюдается снижение производительности. Особенно заметно при небольшом объеме входных данных. Это объясняется подключением второго блэйда и необходимостью обмена сообщениями между 2 разными блэйдами. Отсюда можно сделать вывод о нерациональности использования 2 блэйдов при небольшом объеме входных данных.

Рисунок 1 – Зависимость времени вычисления от числа ядер

Рисунок 2 – Зависимость времени вычисления от числа ядер, логарифмическая шкала

Рисунок 3 – Зависимость времени вычисления от объема данных

Рисунок 4 – Зависимость времени вычисления от объема данных, логарифмическая шкала

Вывод 2

На рисунке 3 видно, что зависимость времени вычисления от объема данных не является линейной, графики похожи на функцию х3. Это объясняется сложностью алгоритма O(n3). При увеличении объема данных в n раз количество вычислений увеличивается в n3 раз.

На рисунке 4 видно, что при при небольшом объеме входных данных использование большого количества ядер неэффективно. Это объясняется возрастанием расходов на межъядерное взаимодействие при увеличении количества ядер.

Рисунок 5 – Зависимость времени при одновременном увеличении числа ядер и объема данных

Вывод 3

На рисунке 5 видно, что зависимость времени вычисления при одновременном увеличении числа ядер и объема данных. Это объясняется сложностью алгоритма O(n3). При увеличении объема данных в n раз количество вычислений увеличивается в n3 раз. Также это объясняется возрастанием расходов на межъядерное взаимодействие при увеличении количества ядер.