21.11.2020

Архитектура вычислительных систем.

Микропроект №1.

НИУ ВШЭ, ФКН ПИ

Албогачиев Алим

БПИ196

1. Описание задания.

Вычислить векторное произведение квадратных матриц A и B. Входные данные: произвольные квадратные матрицы A и B одинаковой размерности. Размер матриц задается входным параметром. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков.

2. Описание реализации.

Для реализации данной задачи используется встроенная библиотека ***threads*** языка ***C++.***

За основу взят так называемый ***Итеративный параллеллизм*** описанный в приложении к лабораторной работе. Так же были использованы другие источники, указанные в списке литературы по данному заданию на сайте:

1.Википедия.Многопточность
2. Википедия.Поток выполнения
3.Википедия.Процесс
4.Уроки по многопотчному программированию
5.Введение в язык C++
6.Документация по C++

Размеры матрицы и количество потоков является входными данными, т.е. наша задача при любых значениях размера матрицы и колчества потоков во-первых, обеспечить корректное создание потоков, во-вторых, обеспечить максимально равномерное разделение вычислительно нагрузки между потоками.

В таком случае, суть многопоточности в данной программе заключается в том, чтобы разбить умножение матриц на N-ое количество потоков. Умножение матриц – это последовательное скалярное умножение векторов, из которых состоят наши матрицы(для первой матрицы горизонатальных, для второй – вертикальных). В таком случае, возьмем за единицу вычисления скалярное произведение этих векторов.

Каждый элемент искомой матрицы произведения, с индексами i,j – скалярное произведение i-строки и j-го столбца исходных первой и второй матриц соотвественно.

Для равномерного разделения нагрузки, между заданным N числом потоков, мы разделим между потоками общее количество элементов искомой матрицы. Представим элементы искомой матрицы в виде одномерного массива. В таком случае, при размерности квадратной матрицы k длина такого массива составит k*k. Каждый поток будет вычислять k*k/N элементов искомой матрицы. Весь остаток(если он есть) k*k%N будет вычисляться в отдельном потоке.

В таком случае достигается оптимальное распределение вычислительной мощности между потоками.

Далее рассмотрим отдельные функции программы:

`int VectorMul(std::vector<std::vector<int>> matrix1, std::vector<std::vector<int>> matrix2, int i, int j); -` фукнция принимает на вход две перемножаемые матрицы и высчитывает скалярное произведение между i-ым и j-ым столбцами.

`int ReadNumber(std::string message, int left, int right); -` считывает целое число после вывода сообщения message.

`void ThreadsFunction(int curr, int last, std::vector<std::vector<int>> matr2, std::vector<std::vector<int>> matr1, std::vector<int>* result, int num); -` фукнция занимается рассчетом задачи отдельного потока, а именно: рассчитывает элементы матрицы умножения result(который в данной программе представлен в виде одномерного массива) в своих границах – с индекса last до индекса curr.

`std::vector<std::vector<int>> GenerateMatrix(int n); -` метод генерирует квадратную матрицу заданного размера со случайными элементами от 0 до 10.

`void ShowMatrix(std::vector<std::vector<int>> matr) –` метод выводит в консоль матрицу.

Функция `main()` создает исходные матрицы для умножения, создает массив-результат умножения и затем, вычислив шаг по описанному раннее шаблону(k*k/N) разпределяет в цикле вычислительные единицы между потоками и запускает те самые потоки с помощью функции `ThreadsFunction.`

После этого цикла следует запуск функции `this_thread::sleep_for` на 100 миллисекунд для того, чтобы все потоки успели запуститься и выпонить вычисления. На конечное время исполнения это влияет очень мало.

Скриншоты с тестами.

Скриншоты с тестами будут приложены отедельно в репозитории.