МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Отчет по лабораторной работе**

**«Умножение разреженных матриц. Элементы типа double. Формат хранения CRS.»**

**Выполнил:** студент группы 381506-1

Дегтярев Антон Юрьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:** ассистент каф. МОСТ ИИТММ

Кустикова Валентина Дмитриевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018

Содержание

[Введение 2](#_Toc469736548)

[Постановка задачи 3](#_Toc469736548)

[Последовательная реализация 4](#_Toc469736549)

[Схема распараллеливания 6](#_Toc469736550)

[Описание программной реализации](#_Toc469736551) 7

[Подтверждение корректности 9](#_Toc469736554)

[Результаты экспериментов по оценке масштабируемости 10](#_Toc469736555)

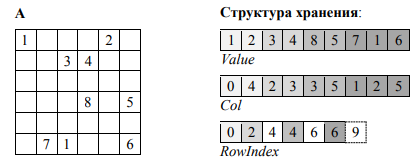
[Заключение 14](#_Toc469736556)

# Введение

Пусть A и B – квадратные матрицы размера N x N, в которых процент ненулевых элементов мал. Будем считать, что элементы матриц A и B – вещественные числа. Требуется найти матрицу C = A \* B. где символ \* соответствует матричному умножению. Формат хранения матриц - CRS

Представляет собой 3 массива, среди которых: первый массив хранит значения элементов построчно, второй – номера столбцов для каждого элемента, а третий заменяет номера строк, используемые в координатном формате, на индекс начала каждой строки. Заметим, что количество элементов массива RowIndex равно N + 1. При этом элементы строки i в массиве Value находятся по индексам от RowIndex[i] до RowIndex[i + 1] – 1 включительно. Исходя из этого обрабатывается случай пустых строк, а также добавляется «лишний» элемент в массив RowIndex – устраняется особенность при доступе к элементам последней строки. Данный элемент хранит номер последнего ненулевого элемента матрицы A плюс 1. что соответствует количеству ненулевых элементов NZ.

Пример:



**Постановка задачи**

Таким образом, требуется разработать программу умножения двух разреженных матриц. Для этого необходимо разработать последовательный и параллельный алгоритмы. Затем сравнить время выполнения на различном числе процессов (потоков) и определить ускорение.

Для распараллеливания использовать следующие инструменты:

* стандарт OpenMP;
* библиотеку Threading Building Blocks.

**Последовательная реализация**

Для выполнения умножения C = A \* B, необходимо сделать следующее:

1. Инициализация матриц A,B,C.

2. Транспонирование матрицы B.

2.1. Обнулим массив RowIndex матрицы АТ . Просмотрим массив Col матрицы A и сосчитаем количество элементов в каждом столбце матрицы A.сохраняя результаты в массиве RowIndex матрицы АТ . Пусть при этом AT.RowIndex[j] хранит количество элементов в столбце j–1 матрицы А (j меняется от 1 до N включительно).

2.2. Подсчитаем индексы начала каждой строки в матрице АТ так, что элемент AT.RowIndex[i] хранит индекс начала (i–1)-ой строки.

2.3. Зная, что в настоящий момент j-я строка матрицы A T начинается с элемента AT.RowIndex[j+1] , будем добавлять V и i в массивы AT.Value и AT.Col соответственно по адресу AT.RowIndex[j+1] , после чего увеличим AT.RowIndex[j+1] на единицу. В конце элемент AT.RowIndex[j+1] будет хранить индекс начала (j+1)-ой строки, что и требуется.

3. Последовательно перемножить каждую строку матрицы A на каждую из строк матрицы BT, записывая в C полученные результаты и формируя ее структуру.

3.1. Создадим дополнительный целочисленный массив X длины N. Инициализируем его числом -1. Обнулим вещественную переменную S.

3.2. Просмотрим в цикле все ненулевые элементы первого вектора V1. Пусть такой элемент с порядковым номером i расположен в столбце с номером j = V1.Col[i]. В этом случае запишем i в j-ю ячейку дополнительного массива.

3.3. Просмотрим в цикле все ненулевые элементы второго вектора V2. Пусть элемент с порядковым номером k расположен в столбце с номером z = V2.Col[k]. Проверим значение X[z]. Если оно равно -1, в первом векторе нет соответствующего элемента, т.е. умножение выполнять не нужно. Иначе умножаем V2.Value[k] и V1.Value[X[z]] и накапливаем в S.

**Схема распараллеливания**

Во внешнем цикле мы перебираем строки матрицы А. и далее работаем с ними независимо. Таким образом, естественный вариант параллелизма – распределение строк между потоками.

1. Дублируем все рабочие структуры по числу строк, собираем в них данные по каждой строке

2. В конце одним потоком все «просуммируем».

# Описание программной реализации

Generator - программа для генерации набора тестовых данных и их сохранение в бинарные файлы. Пользователь указывает имя бинарного файла, размер матрицы и количество элементов, случайно распределенных в каждой строке.



* Deserializer - считывание данных из бинарного файла в текстовый.

Пользователь указывает имена входного и выходного файлов.



Serializer - считывание данных из текстового файла в бинарный. Пользователь указывает имя входного файла.



* Sequence - решение задачи последовательным алгоритмом. Пользователь указывает имена входного и выходного файлов.



* Reference - решение задачи эталонным последовательным алгоритмом. Пользователь указывает имена входного и выходного файлов.



* Solver\_omp - решение задачи параллельным алгоритмом с помощью OpenMP. Пользователь указывает имена входного и выходного файлов, а также количество выполняемых потоков



* Solver \_tbb - решение задачи параллельным алгоритмом с помощью TBB. Пользователь указывает имена входного и выходного файлов, а также количество выполняемых потоков



**Подтверждение корректности**

Для подтверждения корректности, необходимо получить эталонный результат с помощью программы Reference. после запустить программу Checker. Этот результат сравнивается с результатом, полученным с помощью программы Sequence, Solver\_omp или Solver\_tbb. Результат сравнения записывается в файл result.txt.

**Результаты экспериментов по оценке масштабируемости**

Тесты по оценке масштабируемости выполнялись на следующей инфраструктуре:

* Процессор Core i5-3450 CPU @ 3.10 GHz 3.10 GHz
* ОЗУ 8GB
* 64-разрядная операционная система. процессор x64

Таблица 1. Время выполнения (сек)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Размер матрицы** | **Количество элементов в строке** | **1 процесс** | **2 процесса** | **4 процессов** | **8 процессов** |
| 1000 | 100 | 0.2841 | 0.1230 | 0.0650 | 0.0671 | **OMP** |
| 0.1280 | 0.0700 | 0.0683 | **TBB** |
| 10000 | 10 | 2.5061 | 1.1110 | 0.5770 | 0.6750 | **OMP** |
| 1.0960 | 0.5630 | 0.5710 | **TBB** |
| 5000 | 100 | 4.4082 | 2.3641 | 1.3050 | 1.3892 | **OMP** |
| 2.5481 | 1.4571 | 1.5738 | **TBB** |
| 7000 | 100 | 8.7375 | 4.9092 | 2.6821 | 2.8082 | **OMP** |
| 5.2333 | 3.1112 | 3.2891 | **TBB** |
| 9000 | 100 | 14.188 | 8.0834 | 5.3293 | 5.1169 | **OMP** |
| 9.3365 | 5.8282 | 5.6993 | **TBB** |

**OpenMP**

**TBB**

Таблица 2. Ускорение параллельной реализации в сравнении с последовательной

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер матрицы** | **Количество элементов в строке** | **2 процесса** | **4 процессов** | **8 процессов** |
| 1000 | 100 | 2.30 | 4.37 | 4.23 | **OMP** |
| 2.21 | 4.05 | 4.15 | **TBB** |
| 10000 | 10 | 2.25 | 4.34 | 3.71 | **OMP** |
| 2.28 | 4.45 | 4.38 | **TBB** |
| 5000 | 100 | 1.86 | 3.68 | 3.13 | **OMP** |
| 1.73 | 3.025 | 2.80 | **TBB** |
| 7000 | 100 | 1.78 | 3.26 | 3.11 | **OMP** |
| 1.67 | 2.81 | 3.2891 | **TBB** |
| 9000 | 100 | 1.75 | 2.66 | 2.77 | **OMP** |
| 1.52 | 2.43 | 2.49 | **TBB** |

**OpenMP**

**TBB**

**Заключение**

В результате выполнения лабораторной работы была разработана последовательная и две параллельные версии (OpenMP, TBB). Проведённые вычисления показали, при большом числе вершин, параллельная версия имеет выигрыш по времени в несколько раз. Причём, при увеличении числа вершин, ускорение параллельной версии так же увеличивается. Таким образом, можно утверждать, что распараллеливание умножения разреженных матриц является рациональным.