МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

Отчёт по лабораторной работе

**Умножение разреженных матриц. Элементы типа double. Формат хранения матрицы – строковый (CRS).**

**Выполнил:**

студент института ИТММ

гр. 381506-3

Окунев Б.П.

**Проверил:**

Доцент кафедры МОСТ

Сысоев. А.В.

Нижний Новгород

2018 г.

Содержание

[Введение 3](#_Toc514713284)

[Постановка задачи 4](#_Toc514713285)

[Метод решения 5](#_Toc514713286)

[Схема распараллеливания 6](#_Toc514713287)

[Описание программной реализации 7](#_Toc514713288)

[Руководство пользователя 7](#_Toc514713289)

[Руководство программиста 8](#_Toc514713290)

[Подтверждение корректности 9](#_Toc514713291)

[Результаты экспериментов по оценке масштабируемости 10](#_Toc514713292)

[Заключение 13](#_Toc514713293)

# Введение

**Разреженная матрица –** матрица с преимущественно нулевыми элементами. Обычно говорят, что матрица разрежена, если плотность матрицы меньше 50 % (или же количество нулевых элементов таково, что для данного алгоритма и вычислительной системы имеет смысл извлекать выгоду из наличия в ней нулей). Плотностью матрицы называется процентное содержание ненулевых элементов.

Основные способы хранения разреженных матрицы:

* Словарь ключей (DOK);
* Список координат (COO);
* Сжатый разреженный ряд (CSR, CRS);
* Список списков (LIL).

# Постановка задачи

Разработать и реализовать программу для умножения разреженные матриц, используя стандарт OpenMP и библиотеку TBB. В качестве способа хранения матриц должен быть использован способ сжатого хранения строк. Тип элементов матрицы – double.

Необходимо реализовать следующие модули:

* Генератор тестов (по входным параметрам);
* «Наивный» метод решения задачи;
* Последовательный метод решения задачи;
* Параллельный метод с использованием OpenMP;
* Параллельный метод с использованием TBB;
* Checker.

# Метод решения

Для реализации умножения матриц был выбран способ умножения поочерёдного умножение элементов строк матрицы A на элементы строк матрицы BT. Это позволит избежать перепаковок, а также получить ускорение, связанное с устройством кэша процессора.

Описание процесса **транспонирования** матрицы:

1. Формирование N (где N – количество столбцов матрицы) одномерных векторов для хранения номера столбца, а также N векторов для хранения чиста типа double.
2. Просмотр массива строк исходной матрицы и запись номера i данной строки в конец j-го массива.
3. Аналогичные изменения с массивом значений.
4. Сжатия и объединение новых массивов (приведение к CRS формату).

**Умножение** матриц происходит с использованием метода, основанном на использование CRS формата матриц. Пусть матрицы A и B – исходные матрицы, а матрица BT – транспонированная матрица B. Тогда алгоритм выглядит следующим образом:

1. В цикле по количеству строк:

* Формирование массива столбцов, содержащего номер столбца, если в исходном массиве содержится элемент, не равный 0, или -1 в противном случае.
* Последовательная проверка наличия элемента под номером (i,j) в матрицах A и BT. Суммирование произведения данных элементов, если они ненулевые в обеих матрицах. Это происходит с помощью сгенерированного выше массива.
* Добавление полученного значения в новую матрицу.

1. Формирование массива сжатых строк путём последовательного вычисления размера предыдущей строки.
2. Последовательное объединение массивов, полученных на разных потоках, в один.

# Схема распараллеливания

Распараллеливание данных происходит по строкам матрицы разбиением примерно на равные части, соответствующие количеству потоков. Происходит умножение векторов в каждом из потоков, после чего полученные вектора последовательно объединяются.

# Описание программной реализации

## Руководство пользователя

* generator.ccp [file] [matrixSize] [density] - генератор тестов.
  + file - имя выходного файла;
  + matrixSize - размер генерируемых матриц;
  + density - плотность матриц.
* before\_code.cpp [num\_threads] [file1] [file2] - чтение файла теста, вызов решения, запись результата в файл. Применимо как для последовательной версии, так и для параллельных.
  + num\_threads - количество потоков;
  + file1 - имя входного файла;
  + file2 - имя выходного файла.
* answer.cpp [file1] [file2] - "наивная" последовательная версия, результат сохраняется в виде плотной матрицы.
  + file1 - входной файл;
  + file2 - выходной файл.
* checker.cpp [file1] [file2] - программа проверки корректности. Выходной файл result.txt.
  + file1 - входной файл с матрицей участника;
  + file2 - входной файл с матрицей жюри (в стандартной форме).

## Руководство программиста

Для хранения и работы с разреженными матрицами реализован класс CRSmatrix, реализованный в файле CRSmatrix.h.

***Основные методы класса*** CRSmatrix***:***

|  |  |
| --- | --- |
| CRSmatrix getTransposed() const; | Метод, позволяющий получить транспонированную матрицу. |
| CRSmatrix operator\*(const CRSmatrix & crsmatrix); | Метод, производящий умножение матриц. Использует транспонирование матрицы |
| int getSize() | Данные метода позволяют получить значения матрицы. |
| int getNotZero() |
| double\* getValue() |
| int\* getCol() |
| int\* getSparseRow() |

***Генератор тестов:***

Генерация тестов происходит по входным параметрам. Для каждой строки длины N вычисляется количество ненулевых элементов по формуле . Позиция элемента в строке, а также его значение вычисляется случайным образом.

N – размер генерируемой матрицы;

density – плотность матрицы (в процентах).

***Чекер:***

Для проверки корректности производятся следующие действия:

1. Считывание матрицы участника (CRS формат) и матрицы жюри (стандартный формат);
2. Перевод матрицы участника из CRS формат в стандартный;
3. Вычисление суммарной погрешности для разности элементов матрицы участника и матрицы жюри.
4. В зависимости от размера погрешности выставляется вердикт:

* AC = Accepted = Решение выдаёт корректный результат на данном тесте
* WA = Wrong Answer = Решение выдаёт некорректный результат на данном тесте

Результат записывается в формате “VERDICT MESSAGE TIME”.

# Подтверждение корректности

Для подтверждения корректности работы программы был написан модуль “checker”. В качестве входных данных он принимает строку [file1] [file2], где:

file1 - - входной файл с матрицей участника;

file2 - входной файл с матрицей жюри (в стандартной форме).

Результатом является файл result.txt, содержащий корректность решения, а также время участника.

# Результаты экспериментов по оценке масштабируемости

**Характеристики тестового стенда:**

*процессор i5-4570 3.2GHz*

*оперативная память: 8 GB 1866GHz*

*OS: windows 10 PRO*.

**Принцип подсчетов:**

Ускорение высчитывается как отношение tнач/tтек

**Наборы тестов**

|  |  |  |
| --- | --- | --- |
| Номер теста | Размер матрицы. Количество строк и столбцов | Плотность, % |
| 1 | 20 | 10 |
| 2 | 40 | 10 |
| 3 | 80 | 10 |
| 4 | 160 | 10 |
| 5 | 320 | 10 |
| 6 | 640 | 10 |
| 7 | 1000 | 10 |
| 8 | 2000 | 10 |
| 9 | 1000 | 50 |
| 10 | 1000 | 25 |
| 11 | 1000 | 12,5 |
| 12 | 1000 | 6,25 |
| 13 | 1000 | 3,125 |
| 14 | 1000 | 10 |
| 15 | 1000 | 10 |
| 16 | 1000 | 10 |
| 17 | 1000 | 10 |
| 18 | 1000 | 10 |
| 19 | 1000 | 10 |
| 20 | 1000 | 10 |

Табл. 1. Описание тестов

**OpenMP версия. Тестирование на подготовленных наборах тестов.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер теста | Наивный алгоритм, us | OMP 2 потока, us | OMP 4 потока, us | Ускорение 2 потока | Ускорение 4 потока |
| 1 | 32 | 93 | 365 | 0,344086 | 0,087671 |
| 2 | 120 | 246 | 469 | 0,487805 | 0,255864 |
| 3 | 331 | 538 | 510 | 0,615242 | 0,64902 |
| 4 | 2062 | 3093 | 6231 | 0,666667 | 0,330926 |
| 5 | 9194 | 7741 | 5980 | 1,187702 | 1,537458 |
| 6 | 56742 | 35401 | 30552 | 1,602836 | 1,857227 |
| 7 | 195402 | 95725 | 65583 | 2,041285 | 2,979461 |
| 8 | 1483335 | 702142 | 440113 | 2,112585 | 3,37035 |
| 9 | 36104 | 24691 | 24603 | 1,462233 | 1,467463 |
| 10 | 71773 | 49652 | 34117 | 1,445521 | 2,103731 |
| 11 | 152226 | 76084 | 50147 | 2,000762 | 3,035595 |
| 12 | 389554 | 179166 | 107741 | 2,174263 | 3,615652 |
| 13 | 1192377 | 577121 | 345689 | 2,066078 | 3,449277 |
| 14 | 194394 | 95004 | 61471 | 2,046166 | 3,162369 |
| 15 | 195959 | 94962 | 64845 | 2,063552 | 3,02196 |
| 16 | 195298 | 96329 | 74029 | 2,027406 | 2,638128 |
| 17 | 195622 | 96185 | 64195 | 2,03381 | 3,047309 |
| 18 | 198728 | 96063 | 60470 | 2,068726 | 3,28639 |
| 19 | 207934 | 103741 | 60374 | 2,004357 | 3,444098 |
| 20 | 199132 | 95883 | 69339 | 2,076823 | 2,871861 |

Табл. 2. Тестирование OpenMP версии

**TBB версия. Тестирование на подготовленных наборах тестов.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер теста | Наивный алгоритм, us | OMP 2 потока, us | OMP 4 потока, us | Ускорение 2 потока | Ускорение 4 потока |
| 1 | 32 | 385 | 453 | 0,083117 | 0,07064 |
| 2 | 120 | 2293 | 532 | 0,052333 | 0,225564 |
| 3 | 331 | 943 | 1105 | 0,351007 | 0,299548 |
| 4 | 2062 | 2626 | 2942 | 0,785225 | 0,700884 |
| 5 | 9194 | 7865 | 7074 | 1,168976 | 1,299689 |
| 6 | 56742 | 36052 | 25533 | 1,573893 | 2,222301 |
| 7 | 195402 | 100439 | 61886 | 1,945479 | 3,157451 |
| 8 | 1483335 | 687582 | 392031 | 2,157321 | 3,783719 |
| 9 | 36104 | 27177 | 19711 | 1,328476 | 1,831668 |
| 10 | 71773 | 45927 | 31239 | 1,562763 | 2,297545 |
| 11 | 152226 | 75372 | 48635 | 2,019662 | 3,129968 |
| 12 | 389554 | 185013 | 107421 | 2,105549 | 3,626423 |
| 13 | 1192377 | 599310 | 344842 | 1,989583 | 3,457749 |
| 14 | 194394 | 95617 | 59679 | 2,033049 | 3,257327 |
| 15 | 195959 | 95565 | 66205 | 2,050531 | 2,959882 |
| 16 | 195298 | 100419 | 59347 | 1,944831 | 3,290781 |
| 17 | 195622 | 100242 | 60021 | 1,951497 | 3,259226 |
| 18 | 198728 | 98048 | 59412 | 2,026844 | 3,344913 |
| 19 | 207934 | 112441 | 65932 | 1,849272 | 3,153764 |
| 20 | 199132 | 98052 | 60082 | 2,030882 | 3,314337 |

Табл. 3. Тестирование TBB версии

# Заключение

В результате лабораторной работы была реализована программа для работы с разреженными матрицами в CRS формате. Был реализован алгоритм параллельного умножения матриц с элементами типа double с использованием стандарта OpenMP и библиотеки TBB. Для каждой из реализаций проведены оценки масштабируемости, показывающие ускорение. Можно заметить, что максимальное ускорение достигается при наибольшем количестве ненулевых элементов. На малых размерах наблюдается обратная ситуация.