Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4 по курсу "Анализ алгоритмов"

Тема Параллельное умножение матриц

Студент Богаченко А.Е.

Группа ИУ7-56Б

Оценка (баллы) ______________________

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2020 г.

Оглавление

Введение

Умножение матриц является основным инструментом линейной алгебры и имеет многочисленные применения в математике, физике, программировании.

В данной лабораторной работе ставятся следующие задачи:

- изучение распараллеливания вычислений и работа с потоками;
- реализация распараллеленных вычислений;
- экспериментальное сравнение работы алгоритма на разном количестве потоков.

1 Аналитическая часть

Умножение матриц активно применяется в областях физики, математики и программирования [1]. Рассмотрим как можно решить эту задачу.

1.1 Описание задачи

Пусть даны две прямоугольные матрицы A и B размерности $l \times m$ и $m \times n$ соответственно, указанные в формуле 1.1.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix} \tag{1.1}$$

Тогда матрица C будет размерностью $l \times n$ в формуле 1.2, в которой каждый элемент равен выражению из формулы 1.3 [2].

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{l1} & c_{l2} & \cdots & c_{ln} \end{bmatrix} \tag{1.2}$$

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}, i = \overline{1; l}, j = \overline{1; n} \tag{1.3}$$

Операция умножения двух матриц выполнима только в том случае, если число столбцов в первом сомножителе равно числу строк во втором; в этом случае говорят, что матрицы согласованы. В частности, умножение всегда выполнимо, если оба сомножителя – квадратные матрицы одного и того же порядка [2].

Таким образом, из существования произведения $A \times B$ вовсе не следует существование произведения $B \times A$ [2].

Вывод

Умножение матриц необходимый инструмент, для которого есть пути ускорения вычислений за счет уменьшения доли умножения и распараллеливания вычислений.

2 Конструкторская часть

2.1 Схемы алгоритмов

Рассмотрим алгоритм Винограда и способы его расспаралеливания.

2.2 Функциональная модель

На рисунке 2.1 представлена функциональная модель IDEF0 уровня 1.

Рисунок 2.1 – Функциональная модель IDEF0 уровня 1

2.3 Схемы алгоритмов

На рисунке 2.2 изображена схема классического алгоритма.

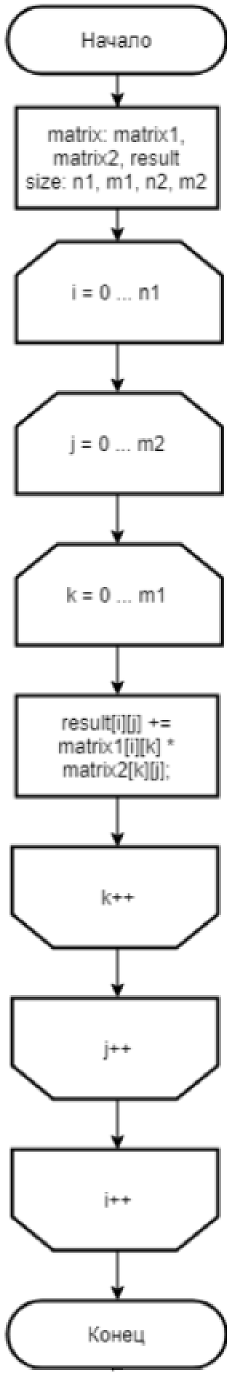

Рисунок 2.2 – Схема классического алгоритма

На рисунке **??** изображена схема алгоритма классического умножения с возможность распараллеливания вычислений.

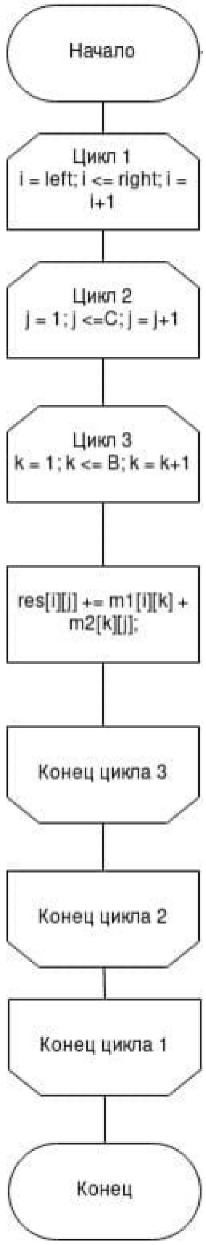

Рисунок 2.3 – Схема классического алгоритма с возможностью распараллеливания

Распараллеливание вычислений реализовано благодаря добавлению двух новых переменных `left` и `right`, которые указывают на диапазон строк, которые необходимо рассчитать.

Вывод

Благодаря возможности вычислять каждый элемент результативной матрицы отдельно друг от друга, удалось разделить вычисления по строчкам, выдавая каждому потоку диапазон строк, в которых необходимо считать. После выполнения всех потоков и соответственно расчета всех строк матрицы, получается правильный результат.

3 Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

К программе предъявляется ряд требований:

- корректное умножение матриц;
- при матрицах неправильных размеров программа не должна аварийно завершаться.

3.2 Средства реализации

В качестве языка программирования был выбран `Commnon LISP` с компилятором `SBCL`. В данном языке присутствуют нативные потоки.

Для распараллеливания вычислений был использован модуль `lparallel` из библиотеки `quick lisp` [3].

3.3 Листинг кода

Исходный код распараллеленного алгоритма приведен в листинге 3.1.

Листинг 3.1 – Параллельный классический алгоритм

```
(ql:quickload :array-operations)
(ql:quickload :lparallel)

(setf lparallel:*kernel*
    (lparallel:make-kernel 8))

(defun init-matrix (rows columns)
    (aops:generate (lambda () (random 10)) (list rows columns)))

(defun mult-row (m1-row-idx m2-col-idx)
    (setq sum 0)
    (loop for i from 0 below (length (aops:sub matrix1 m1-row-idx))
        do (setf sum
            (+ sum (* (aref (aops:sub matrix1 m1-row-idx) i)
                    (aref (aops:sub (list-to-2d-array
                                   (rotate (2d-array-to-list matrix2)))
                                    m2-col-idx) i)))))

    (setf (aref new-matrix m1-row-idx m2-col-idx) sum))

(defun mult-matrix (matrix1 matrix2)
    (destructuring-bind (n m) (array-dimensions matrix1)
        (setq m1-rows n)
        (setq m1-cols m))

    (destructuring-bind (n m) (array-dimensions matrix2)
        (setq m2-cols m)
        (setq m2-rows n))

    (defvar new-matrix-rows m1-rows)
    (defvar new-matrix-columns m2-cols)

    (if (/= m2-rows m1-cols)
        (return-from mult-matrix
            (PRINT "Wrong dimensions!")))

    (defvar new-matrix (aops:generate
        (lambda () ())
            (list new-matrix-rows new-matrix-columns)))

    (time (lparallel:pdotimes (i m1-rows)
        (dotimes (j m2-cols)
```

```
          (mult-row i j))))
    (PRINT new-matrix))

(defun rotate (list-of-lists)
  (apply #'mapcar #'list list-of-lists))

(defun 2d-array-to-list (array)
  (loop for i below (array-dimension array 0)
        collect (loop for j below (array-dimension array 1)
                      collect (aref array i j))))

(defun list-to-2d-array (list)
  (make-array (list (length list)
                    (length (first list)))
              :initial-contents list))
```

Для тестирования программы были заготовлены следующие тесты в таблице 3.1.

Таблица 3.1 – Тесты для алгоритмов

Первая матрица	Вторая матрица	Ожидаемый результат
1 2 3 4	1 2 3 4	7 10 15 22
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	30 36 42 66 81 96 102 126 150
1 2 3 4 5 6	1 2 3	14 32

Вывод

Был разработан и протестирован многопоточный алгоритм.

4 Исследовательская часть

Проведем тестирование и сравним алгоритмы по времени работы.

4.1 Пример работы

Демонстрация работы программы приведена на рисунке 4.1.

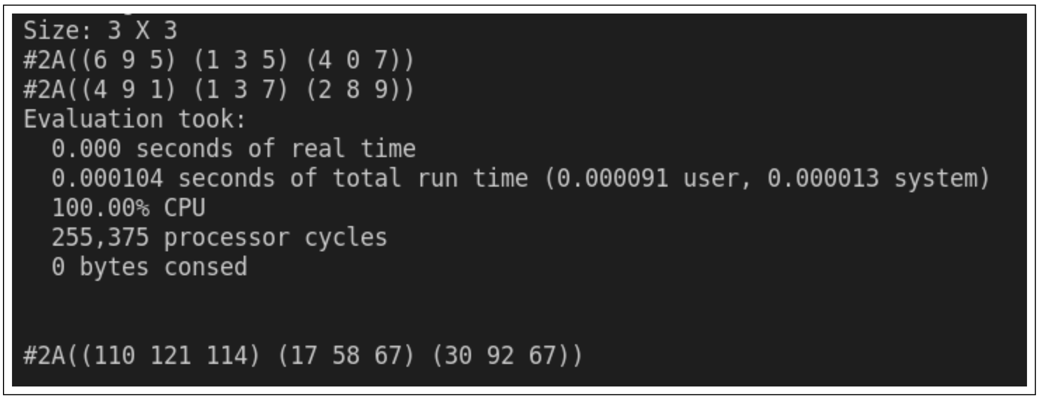

Рисунок 4.1 – Размерность 3x3, 2 потока

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Kali [4] Linux [5] 5.8.10-1kali1 64-bit.
- Память: 8 GB.
- Процессор: Intel® Core™ i5-8250U [6] CPU @ 1.60GHz
- Количество логических потоков: 8

Тестирование проводилось на ноутбуке при включённом режиме производительности. Во время тестирования ноутбук был нагружен только системными процессами.

4.3 Результаты тестирования

Результаты продемонстрированы в таблицах 4.1 и 4.2.

Таблица 4.1 – Результаты однопоточного алгоритма

Первая матрица	Вторая матрица	Результат
1 2 3 4	1 2 3 4	7 10 15 22
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	30 36 42 66 81 96 102 126 150
1 2 3 4 5 6	1 2 3	14 32

Таблица 4.2 – Результаты многопоточного алгоритма

Первая матрица	Вторая матрица	Результат
1 2 3 4	1 2 3 4	7 10 15 22
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	30 36 42 66 81 96 102 126 150
1 2 3 4 5 6	1 2 3	14 32

4.4 Замеры времени

Время замерялось с помощью макроса `time` [7].

На рисунке 4.2 представлены результаты замера времени алгоритма.

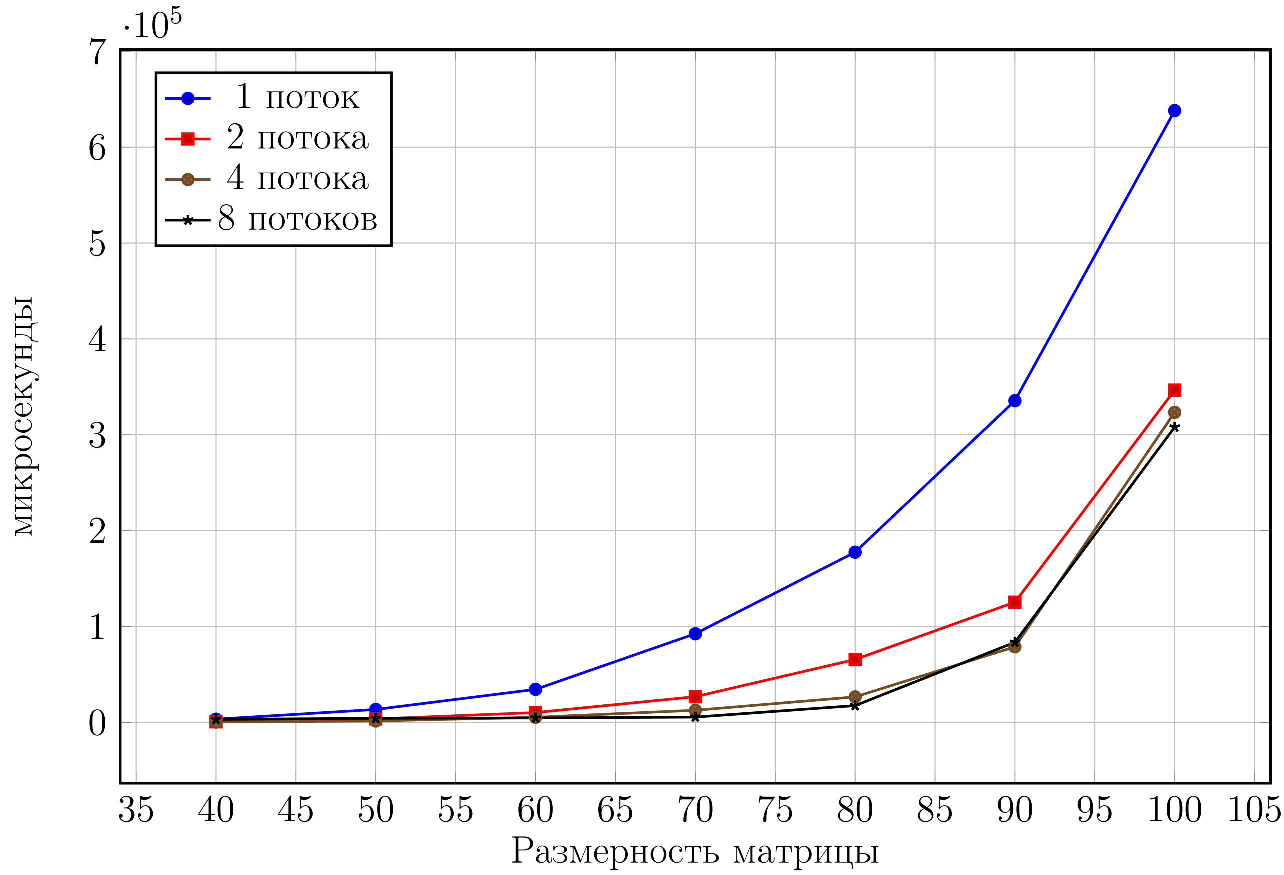

Рисунок 4.2 – Результаты замеров

Выводы

Из графиков отношения размерности матрицы ко времени вычисления видно, что рассчет на одном потоке работает в 2 раза медленнее вычислений на нескольких потоках. Среди распараллеленных вычислений видно, что увеличение числа потоков дает небольшой прирост к скорости примерно на 5%. Но чем больше потоков используется, тем этот прирост меньше.

Заключение

В ходе данной работы было проведено сравнение расчета произведения матриц на нескольких потоках, а именно на 1, 2, 4 и 8, были сделаны следующие выводы:

- распараллеленные вычисления эффективней в 2 раза;
- использование числа потоков больше, чем число потоков процессора не дает выйгрыша по времени и может даже работать медленнее.

Все цели, поставленные на эту работы, были выполнены:

1. изучено распараллеливания вычислений и работа с потоками;
2. реализация распараллеленных вычислений;
3. экспериментальное сравнение работы алгоритма на разном количестве потоков.

Литература

[1] Анисимов Н.С. Строганов Ю.В. Реализация алгоритма умножения матриц по Винограду на языке Haskell. – Новые информационные технологии в автоматизированных системах, 2018.

[2] Корн Г А. Алгебра матриц и матричное исчисление. – McGraw-Hill Book Company, 1968.

[3] Документация по lparallel [Электронный ресурс]. Режим доступа: https://lparallel.org/api/kernel/ (дата обращения: 16.10.2020).

[4] Our Most Advanced Penetration Testing Distribution, Ever. [Электронный ресурс]. Режим доступа: https://kali.org/ (дата обращения: 12.09.2020).

[5] Linux – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Linux (дата обращения: 12.09.2020).

[6] Intel Processors [Электронный ресурс]. Режим доступа: https://www.intel.com/content/www/us/en/products/processors/core/i5-processors.html (дата обращения: 12.09.2020).

[7] Документация по time [Электронный ресурс]. Режим доступа: http://www.lispworks.com/documentation/lw60/LW/html/lw-628.htm (дата обращения: 16.10.2020).