**Практикум по курсу**

**"Параллельное программирование"**

**Разработка параллельной версии программы для перемножения матриц с использованием алгоритма Фокса.**

**ОТЧЕТ**

**о выполненном задании**

студента группы 11-804 ВШ ИТИС

Елина Артёма Ивановича

Казань, 2020 г.

Оглавление

[1 Постановка задачи 3](#__RefHeading__3055_2033249847)

[2 Описание алгоритма Фокса умножения матриц 4](#__RefHeading__3057_2033249847)

[2.1 Основа: последовательный алгоритм 4](#__RefHeading__3059_2033249847)

[2.2 Параллельный алгоритм 5](#__RefHeading__3061_2033249847)

[2.3 Таблицы 6](#__RefHeading__3063_2033249847)

[2.4 3D-график 7](#__RefHeading__3065_2033249847)

[2.4.1 MPI-алгоритм. 7](#__RefHeading__3071_2033249847)

[3 Анализ результатов 8](#__RefHeading__3073_2033249847)

[4 Выводы 9](#__RefHeading__3075_2033249847)

# Постановка задачи

Даны две квадратные матрицы. Необходимо найти их произведение при помощи алгоритма Фокса (используя блочное разделение матриц):

\* = ,

где каждый блок матрицы С определяется в соответствии с выражением

=

За основу берутся вычисления, выполняемые над матричными блоками.

Требуется:

1. Реализовать параллельную версию предложенного алгоритма с использованием технологии MPI.
2. Исследовать масштабируемость полученной параллельной программы: построить графики зависимости времени исполнения от числа процессоров для различного объёма входных данных.
3. Для каждого набора входных данных найти количество процессоров, при котором время выполнения задачи перестаёт уменьшаться.
4. Определить основные причины недостаточной масштабируемости программы при максимальном числе процессоров.

# Описание алгоритма Фокса умножения матриц

## Основа: последовательный алгоритм

Простейшая форма алгоритма Фокса умножения матриц имеет стандартный вид алгоритма умножения матриц.

Этот алгоритм является итеративным и ориентирован на последовательное вычисление единичных блоков матрицы С (т.е. ее клеток). Предполагается выполнение n3 операций умножения и столько же операций сложения элементов исходных матриц. Количество выполненных операций имеет порядок O(n^3).

## Параллельный алгоритм

Основная идея метода Фокса – разбиение умножения матриц на подзадачи умножения блоков матриц. Каждой подзадаче на конкретной итерации доступны только необходимые блоки матриц A и B, а результатом ее работы будет подсчет отдельного блока матрицы С.

Подзадачи нумеруются в зависимости от расположения результирующего блока в матрице С, то есть пара (i,j) будет соответствовать подзадаче, отвечающей за вычисление блока .

Этапы алгоритма:

1. Инициализация: каждой подзадаче передаются соответствующие блоки
2. Проводятся GridSize итераций, где на итерации stage для каждой строки блок пересылается на все подзадачи той же строки; индекс j вычисляется по формуле (i+stage)%n; к прибавляется произведение полученных в результате пересылок блоков; пересылаются подзадачам, являющимся соседями сверху данной подзадачи в сетке подзадач (для нижней строки это будет нулевая строка).

Рассматриваются квадратные матрицы такие, что число процессоров является полным квадратом от количества блоков на стороне. Это необходимо для равномерного распределения нагрузки между процессорами.

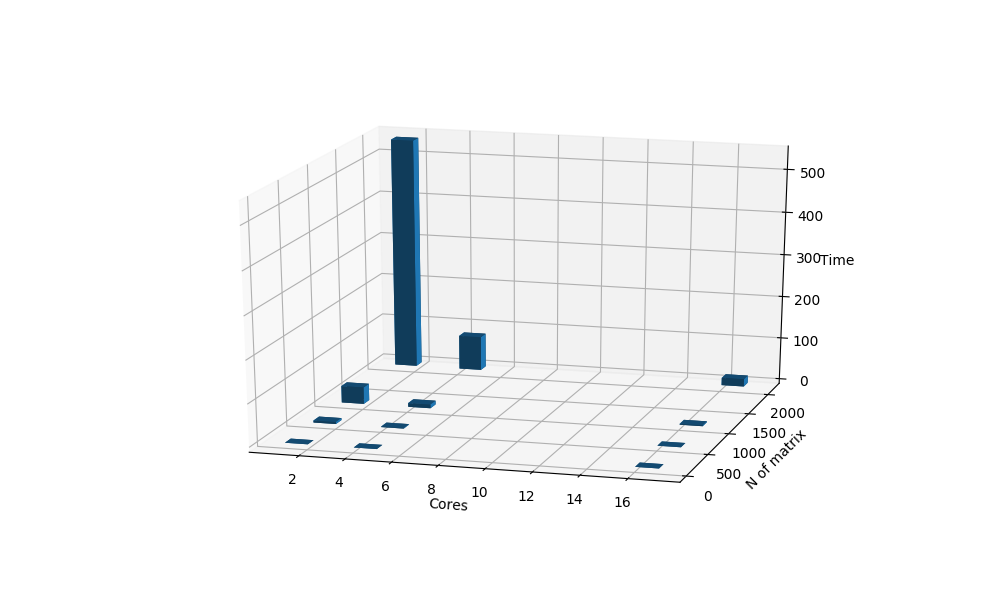
## Таблицы

MPI-алгоритм на различном количестве процессов и различном размере матриц:

|  |  |  |
| --- | --- | --- |
| ProcNum | MatrixSize | Time |
| 1 | 16 | 0.000659227 |
| 1 | 512 | 4.66383 |
| 1 | 1024 | 39.2572 |
| 1 | 2048 | 542.515 |
| 4 | 16 | 0.000434875 |
| 4 | 512 | 1.06294 |
| 4 | 1024 | 8.88665 |
| 4 | 2048 | 80.6529 |
| 16 | 16 | 0.002195 |
| 16 | 512 | 0.273459 |
| 16 | 1024 | 2.08103 |
| 16 | 2048 | 17.6535 |

## 3D-график

### MPI-алгоритм



# Анализ результатов

Благодаря использованию технологии MPI возможен практически линейный прирост производительности при увеличении числа потоков.

Если количество нитей (процессоров) превышает количество клеток матрицы, то улучшение выполнения алгоритма прекратится.

При этом, если отвечать на вопрос о целесообразности добавления процессоров, то для MPI-реализации алгоритм наиболее быстро работает:

Для матриц размера 16 при количестве процессоров ProcNum = 4;

Для матриц большего размера уже быстрее выполняется при 16 процессорах.

# Выводы

На небольших матрицах время работы алгоритмов наименьшее при использовании лишь одного процессора: распараллеливание в данном случае оказывает лишь негативный эффект на время работы программы.

При увеличении размеров матриц появляется положительный эффект от использования дополнительных процессоров: увеличивая их количество, линейно от их количества уменьшается время работы программы.

При этом MPI реализацию можно ускорить, если хранить матрицы не как двумерные массивы, а одномерные (для того, чтобы не переводить блоки при пересылке в одномерные массивы и обратно).