****МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

**Практикум по курсу**

**"Суперкомпьютеры и параллельная обработка данных"**

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

**ОТЧЕТ**

**о выполненном задании**

студента 325 учебной группы факультета ВМК МГУ

Ивачева Федора Владиславовича

Москва, 2017 г.

Оглавление

[1 Постановка задачи - 2 -](#_Toc469570567)

[2 Описание алгоритма Фокса умножения матриц - 2 -](#_Toc469570568)

[2.1 Основа: последовательный алгоритм - 2 -](#_Toc469570569)

[2.2 Параллельный алгоритм - 3 -](#_Toc469570570)

[3 Результаты замеров времени выполнения - 3 -](#_Toc469570571)

[3.1 Таблицы - 3 -](#_Toc469570572)

[3.2 3D-графики - 8 -](#_Toc469570573)

[4 Анализ результатов - 10 -](#_Toc469570579)

[5 Выводы - 10 -](#_Toc469570580)

# 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 технологий OpenMP и 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.

5. Сравнить эффективность OpenMP и MPI-версий параллельной программы.

# 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 n^3 операций умножения и столько же операций сложения элементов исходных матриц. Количество выполненных операций имеет порядок O(n^3).

## Параллельный алгоритм

В алгоритме Фокса базовые подзадачи отвечают за вычисления отдельных блоков матрицы С и при этом на каждой итерации подзадачи располагаются только необходимые для этого блоки матриц А и В – т.е. по одному блоку от каждой матрицы.

Пронумеруем подзадачи исходя из расположения результирующих блоков матрицы С, то есть пара (i,j) будет соответствовать подзадаче, отвечающей за вычисление блока .

На каждой подзадаче (i,j) будут находиться четыре блока: 1) блок 2) блок 3) блоки и , получаемые подзадачей в процессе вычислений.

Этапы алгоритма:

1. Инициализация: каждой подзадаче передаются соответствующие блоки и обнуляются
2. Проводятся n 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).

Рассматривается наиболее простой случай, когда число процессоров является полным квадратом от количества блоков по горизонтали и вертикали. Таким образом нагрузка между процессорами будет одинаковой.

Коды программ можно найти в github-репозитории: <https://github.com/Teodorissimo/FoxAlgorithm>

Ниже приведены результаты замеров времени программ на суперкомпьютерах: непосредственно в табличной форме и наглядно на 3D-графиках.

Программа была запущена в конфигурациях:

* на Regatta - 1,4,16 ядер для OpenMP-программы;
* на Bluegene - 1,4 для OpenMP.
* На Regatta – 1,4,16 ядер для MPI-реализации;
* На rsc – 1, 4 и 16 ядер для MPI-реализации

Каждая конфигурация была запущена 3 раза для OpenMP. Так как относительная разница во времени выполнения для трех запусков минимальна, то было решено произвести по одному запуску для MPI-реализации.

## Таблицы

OpenMP:

На Regatta был запущен алгоритм на размерах матриц до 2048 на 2048, но при использовании одного потока время выполнения программы превысило максимальные 15 минут (поэтому такая конфигурация не отображена на графике). На Bluegene алгоритм уже был запущен для матриц размером до 1024, чтобы не превышать критическое время работы алгоритма.

MPI показал себя с лучшей стороны: получилось провести расчеты на бОльших размерах матриц.

OpenMP на Regatta:

Time: время работы на компьютере, ProcNum – количество потоков, mSize – размер матрицы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Time: | 6.5784023630 | ||||||| | ProcNum: | 16 | mSize: | 1024 |
| Time: | 6.6194283440 | ||||||| | ProcNum: | 16 | mSize: | 1024 |
| Time: | 7.5763089820 | ||||||| | ProcNum: | 16 | mSize: | 1024 |
| Time: | 0.0408303400 | ||||||| | ProcNum: | 16 | mSize: | 128 |
| Time: | 0.0541550861 | ||||||| | ProcNum: | 16 | mSize: | 128 |
| Time: | 0.0472249080 | ||||||| | ProcNum: | 16 | mSize: | 128 |
| Time: | 0.0012958440 | ||||||| | ProcNum: | 16 | mSize: | 16 |
| Time: | 0.0012841460 | ||||||| | ProcNum: | 16 | mSize: | 16 |
| Time: | 0.0012700780 | ||||||| | ProcNum: | 16 | mSize: | 16 |
| Time: | 85.7909919030 | ||||||| | ProcNum: | 16 | mSize: | 2048 |
| Time: | 85.8028811560 | ||||||| | ProcNum: | 16 | mSize: | 2048 |
| Time: | 85.6750405960 | ||||||| | ProcNum: | 16 | mSize: | 2048 |
| Time: | 0.1271177900 | ||||||| | ProcNum: | 16 | mSize: | 256 |
| Time: | 0.1415040290 | ||||||| | ProcNum: | 16 | mSize: | 256 |
| Time: | 0.1562151030 | ||||||| | ProcNum: | 16 | mSize: | 256 |
| Time: | 0.0027930280 | ||||||| | ProcNum: | 16 | mSize: | 32 |
| Time: | 0.0028297230 | ||||||| | ProcNum: | 16 | mSize: | 32 |
| Time: | 0.0028006520 | ||||||| | ProcNum: | 16 | mSize: | 32 |
| Time: | 0.8277926030 | ||||||| | ProcNum: | 16 | mSize: | 512 |
| Time: | 0.5670156280 | ||||||| | ProcNum: | 16 | mSize: | 512 |
| Time: | 0.6911229900 | ||||||| | ProcNum: | 16 | mSize: | 512 |
| Time: | 0.0100089260 | ||||||| | ProcNum: | 16 | mSize: | 64 |
| Time: | 0.0141682220 | ||||||| | ProcNum: | 16 | mSize: | 64 |
| Time: | 0.0140133740 | ||||||| | ProcNum: | 16 | mSize: | 64 |
| Time: | 158.2899041710 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 158.7339290300 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 158.4148839150 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 0.1029303400 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 0.1023085590 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 0.1037036410 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 0.0002375870 | ||||||| | ProcNum: | 1 | mSize: | 16 |
| Time: | 0.0002348110 | ||||||| | ProcNum: | 1 | mSize: | 16 |
| Time: | 0.0002364970 | ||||||| | ProcNum: | 1 | mSize: | 16 |
| Time: | 0.8256592310 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 0.8287470620 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 0.8250791660 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 0.0016570530 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 0.0016608140 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 0.0016745490 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 8.1224482741 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 8.0856177230 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 8.0051871610 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 0.0129247630 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 0.0136507320 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 0.0128888800 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 39.1748762120 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 37.0674204530 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 35.3792220050 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 0.0720227270 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.0652327720 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.0603217830 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.0004408830 | ||||||| | ProcNum: | 4 | mSize: | 16 |
| Time: | 0.0004397630 | ||||||| | ProcNum: | 4 | mSize: | 16 |
| Time: | 0.0004525810 | ||||||| | ProcNum: | 4 | mSize: | 16 |
| Time: | 359.7314773990 | ||||||| | ProcNum: | 4 | mSize: | 2048 |
| Time: | 359.9895335840 | ||||||| | ProcNum: | 4 | mSize: | 2048 |
| Time: | 360.4479481780 | ||||||| | ProcNum: | 4 | mSize: | 2048 |
| Time: | 0.6196236710 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.4881290401 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.5258770680 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.0015733560 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 0.0015707460 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 0.0016025870 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 2.1838203200 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 2.5181763820 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 1.8722037890 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 0.0083420560 | ||||||| | ProcNum: | 4 | mSize: | 64 |
| Time: | 0.0101751330 | ||||||| | ProcNum: | 4 | mSize: | 64 |
| Time: | 0.0101772260 | ||||||| | ProcNum: | 4 | mSize: | 64 |

OpenMP на Bluegen:

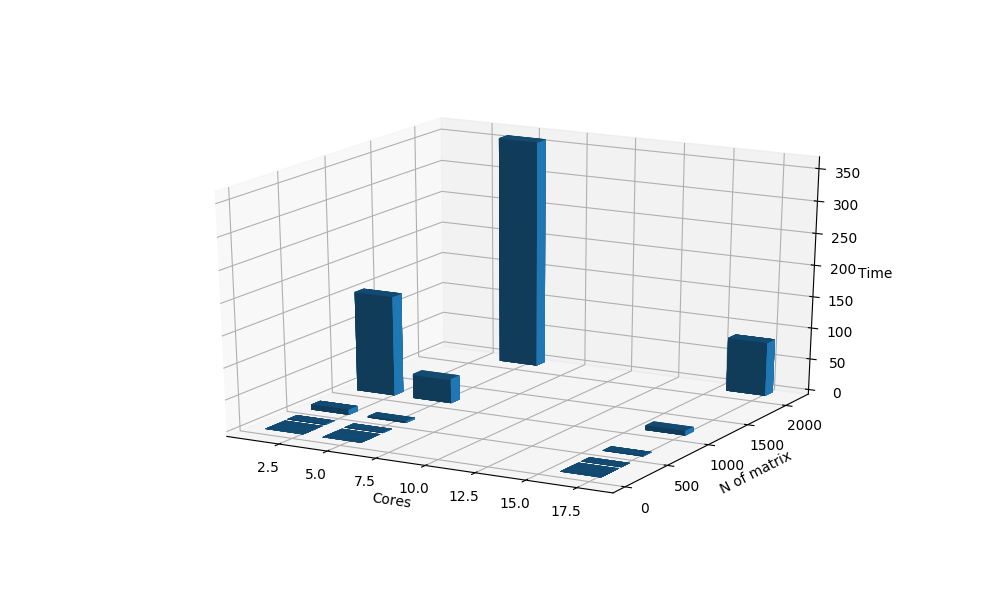
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Time: | 69.8710839748 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 69.8710358143 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 69.8710470200 | ||||||| | ProcNum: | 1 | mSize: | 1024 |
| Time: | 0.1374161243 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 0.1374440193 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 0.1374418736 | ||||||| | ProcNum: | 1 | mSize: | 128 |
| Time: | 1.0885188580 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 1.0885190964 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 1.0885200500 | ||||||| | ProcNum: | 1 | mSize: | 256 |
| Time: | 0.0018868446 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 0.0018570423 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 0.0018849373 | ||||||| | ProcNum: | 1 | mSize: | 32 |
| Time: | 8.7369418144 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 8.7369351387 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 8.7369420528 | ||||||| | ProcNum: | 1 | mSize: | 512 |
| Time: | 0.0046641827 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 0.0046639442 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 0.0046641827 | ||||||| | ProcNum: | 1 | mSize: | 64 |
| Time: | 17.5834381580 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 17.6578750610 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 17.6251900196 | ||||||| | ProcNum: | 4 | mSize: | 1024 |
| Time: | 0.0168728828 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.0168819427 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.0168480873 | ||||||| | ProcNum: | 4 | mSize: | 128 |
| Time: | 0.2760000229 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.2738280296 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.2739291191 | ||||||| | ProcNum: | 4 | mSize: | 256 |
| Time: | 0.0016369820 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 0.0016369820 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 0.0016369820 | ||||||| | ProcNum: | 4 | mSize: | 32 |
| Time: | 2.2061159611 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 2.2060608864 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 2.2061388493 | ||||||| | ProcNum: | 4 | mSize: | 512 |
| Time: | 0.0023140907 | ||||||| | ProcNum: | 4 | mSize: | 64 |
| Time: | 0.0023398399 | ||||||| | ProcNum: | 4 | mSize: | 64 |
| Time: | 0.0023419857 | ||||||| | ProcNum: | 4 | mSize: | 64 |

MPI на Regatta:

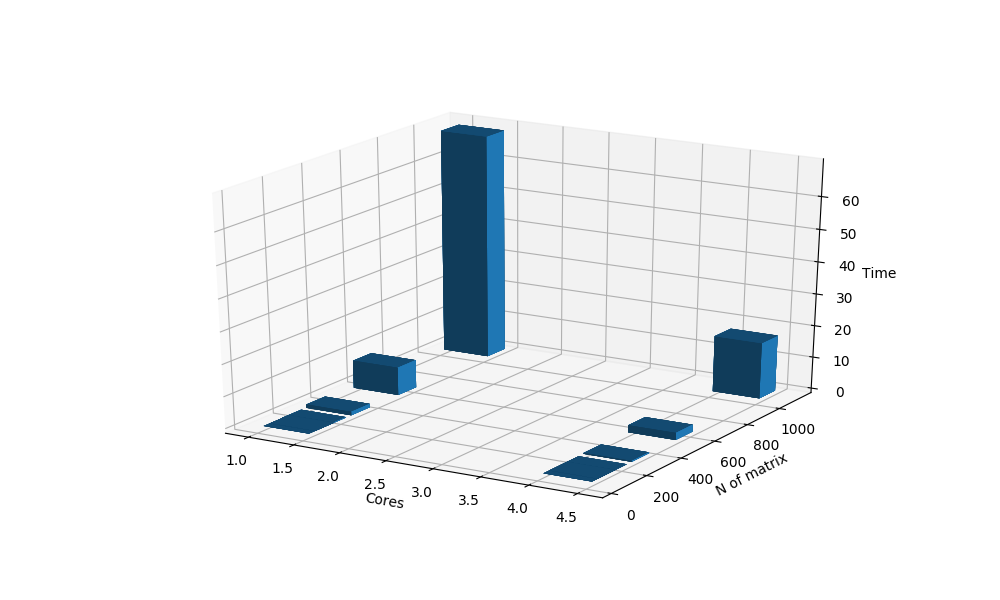
|  |  |  |
| --- | --- | --- |
| ProcNum | MatrixSize | Time |
| 1 | 16 | 0.000659227 |
| 1 | 512 | 4.66383 |
| 1 | 1024 | 39.2572 |
| 1 | 2048 | 542.515 |
| 4 | 16 | 0.000434875 |
| 4 | 512 | 1.06294 |
| 4 | 1024 | 8.88665 |
| 4 | 2048 | 80.6529 |
| 16 | 16 | 0.002195 |
| 16 | 512 | 0.273459 |
| 16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

## 3D-графики

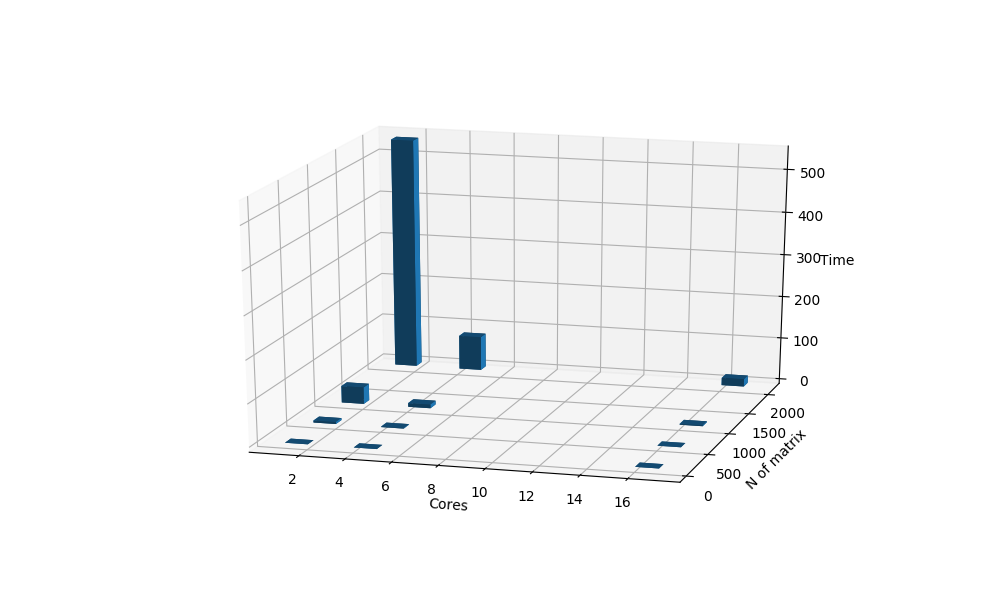
### OpenMP на Regatta



### OpenMP на Bluegene



### MPI на Regatta



# Анализ результатов

Благодаря использованию техногии OpenMP и 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 OpenMP-реализации на Regatta алгоритм наиболее быстро работает:

Для матриц размера 16 при количестве процессоров, равном 1;

Для матриц размера 32 при 4 процессорах;

Для матриц размера 64 и больше при 16 процессорах

Для MPI-реализации алгоритм быстрее всего выполняется:

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 больших размерах матриц MPI работает ощутимо быстрее OpenMP реализации.

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).

PS Также было выполнено тестирование MPI-реализации на собственном двухъядерном компьютере. Ввиду того, что количество ядер в нем не равняется квадрату натурального числа, то было решено запустить программу при размере матрицы n=16 на одном и четырех потоках. Очевидно, что в последнем случае говорить о четырехкратном приросте скорости работы по сравнению с одноядерным режимом не приходится. Мало того, теоритически даже не должно быть двукратного прироста, так как физическое количество ядер вдвое меньше задаваемого.

В любом случае, при размере матрицы в 16 на 16 клеток время работы алгоритма при nProc = 1 составляет 0.00004 секунды, при nProc = 4: 0.03 секунды. Колоссальная разница во времени работы обусловлена теми же причинами, которые были указаны выше.

При линейном размере матрицы уже в 1024 клетки время работы при одном ядре уже 31 секунда, что примерно соответствует времени работы при банном размере матрицы на Regatta при одном ядре, при четырех ядрах – 4 секунды. (быстрее Regatta).