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 ленточного алгоритма.

ОТЧЕТ

о выполненном задании

студента 321 учебной группы факультета ВМК МГУ

Аграновского Михаила Леонидовича

Москва, 2016 г.

Оглавление

1 Постановка задачи

Ставится задача перемножения двух квадратных матриц при помощи т.н. ленточного алгоритма:

$$C = \mathrm{A} \times \mathrm{B}, \quad \mathrm{A}, \mathrm{B}, \mathrm{C} \in Z^{n \times n}$$

Результатом перемножения матриц A и B является матрица C, каждый элемент которой есть скалярное произведение соответствующих строк матрицы A и столбцов матрицы B.

Требуется:

1. Реализовать параллельные алгоритмы ленточного перемножения матриц с помощью технологий параллельного программирования OpenMP и MPI.
2. Сравнить их эффективность.
3. Исследовать масштабируемость полученных программ и построить графики зависимости времени выполнения программ от числа используемых ядер и объёма входных данных.

2 Описание алгоритма ленточного умножения матриц

2.1 Основа: последовательный алгоритм

Простейшая форма алгоритма ленточного умножения матриц имеет следующий вид:

```
void ribbonMult(int *a, int *b, int *c, int n)
{
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         auto pElem = &(c[i * n + j]);
         *pElem = 0;
         for (int k = 0; k < n; k++) {
            *pElem += a[i * n + k] * b[k * n + j];
         }
      }
   }
}
```

Этот алгоритм является итеративным и ориентирован на последовательное вычисление строк матрицы C. Предполагается выполнение n*n*n операций умножения и столько же операций сложения элементов исходных матриц. Количество выполненных операций имеет порядок O(n**3).

2.2 Параллельный алгоритм

Разбиваем задачу вычисления конечной матрицы на подзадачи по вычислению строк и распределяем их по потокам (OpenMP) / процессам (MPI). Разбиение на вычисление отдельных полей не производим, т.к. размеры матриц при вычислениях и так будут на порядки превышать число потоков (процессов).

Ниже приведены краткие заметки по реализациям параллельных алгоритмах на OpenMP и MPI. Коды программ можно найти в github-репозитории: https://github.com/agrml/ribbonMultiplicationSummary

В OpenMP-версии вся модификация когда сводится к добавлению клаузы ***omp parallel for.***

В MPI-версии производится широковещательная рассылка заполненных матриц ***a, b***, а в конце работы – ***reduce*** результатов. Для синхронизации используются команды ***MPI_Barrier.***

Реализованные алгоритмы проверялись на корректность и совпадение по результатам с последовательным алгоритмом (соответствующий код закомментирован).

3 Результаты замеров времени выполнения

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

Программа была запущена в конфигурациях:

- на Regatta - 1,2,4,8,16 ядер для MPI и OpenMP-программы;
- на Bluegene - 1,2,4 для OpenMP; 1,2,4,8,16,32,64,128,256 для MPI.

Также для сравнения OpenMP-версия программы была запущена на ноутбуке (Core i5-6300HQ 2.30GHz × 4, 24GB RAM, Ubuntu 16.04)

3.1 Таблицы

*** OpenMP on Regatta ***

# cores \ size	512	1024	1546	2048	2660	3072
1	0.0340704560	1.5013673610	2.3241722385	TL	TL	TL
2	0.0176534441	0.6935598996	1.1644718223	8.6897240281	13.1839327892	TL
4	0.0098530385	0.3782198442	0.6460601939	4.6249823266	4.1428378052	12.6873633676
8	0.0099463238	0.1945567767	0.3219709224	2.1271543503	3.0596747663	6.6800018933
16	0.0078350504	0.1153473430	0.1570382436	1.1371890836	1.2857059280	3.5861414009

*** MPI on Regatta ***

# cores \ size	512	1024	1546	2048	2660	3072
1	0.0912712222	2.2039444444	3.4378555556	TL	TL	TL
2	0.0523225556	1.0324500000	2.1455277778	12.8050666667	12.4375333333	TL
4	0.0298995556	0.5296877778	0.7725750000	4.8668944444	7.5279388889	TL
8	0.0159858000	0.2666544444	0.4407155556	2.7794055556	4.1788277778	8.5385444444
16	0.0083096111	0.1258391667	0.2372777778	1.2826244444	2.2486388889	4.3690555556

*** OpenMP on Bluegene ***

# cpus \ size	512	1024	1546	2048	2660	3072
1	0.1425155242	1.1412363781	3.9497226238	TL	TL	TL
2	0.0712737388	0.5706499563	1.9758077727	TL	TL	TL
4	0.0356570893	0.2853540407	0.9927679830	TL	TL	TL
8	TL	TL	TL	TL	TL	TL
16	TL	TL	TL	TL	TL	TL
32	TL	TL	TL	TL	TL	TL
64	TL	TL	TL	TL	TL	TL
128	TL	TL	TL	TL	TL	TL
256	TL	TL	TL	TL	TL	TL

*** MPI on Bluegene ***

# cpus \ size	512	1024	1546	2048	2660	3072
1	0.2167766667	1.7355833333	5.9898833333	TL	TL	TL
2	0.1083780000	0.8677888889	2.9929222222	11.3540166667	TL	TL
4	0.0541890000	0.4338816667	1.4983066667	5.6770500000	7.6518722222	TL
8	0.0271379444	0.2169444444	0.7510988889	2.8385666667	3.8286555556	10.0985000000
16	0.0136139167	0.1086423889	0.3755627778	1.4193422222	1.9195833333	5.0530000000
32	0.0068293667	0.0544948889	0.1956677778	0.7277800000	0.9777333333	2.5269916667
64	0.0034248778	0.0273389444	0.0967936667	0.3558733333	0.4827883333	1.2670422222
128	0.0017153111	0.0137103889	0.0503345556	0.1783272222	0.2414044444	0.6372911111
256	0.0008579628	0.0068695278	0.0271044444	0.0893645556	0.1264602778	0.3205294444

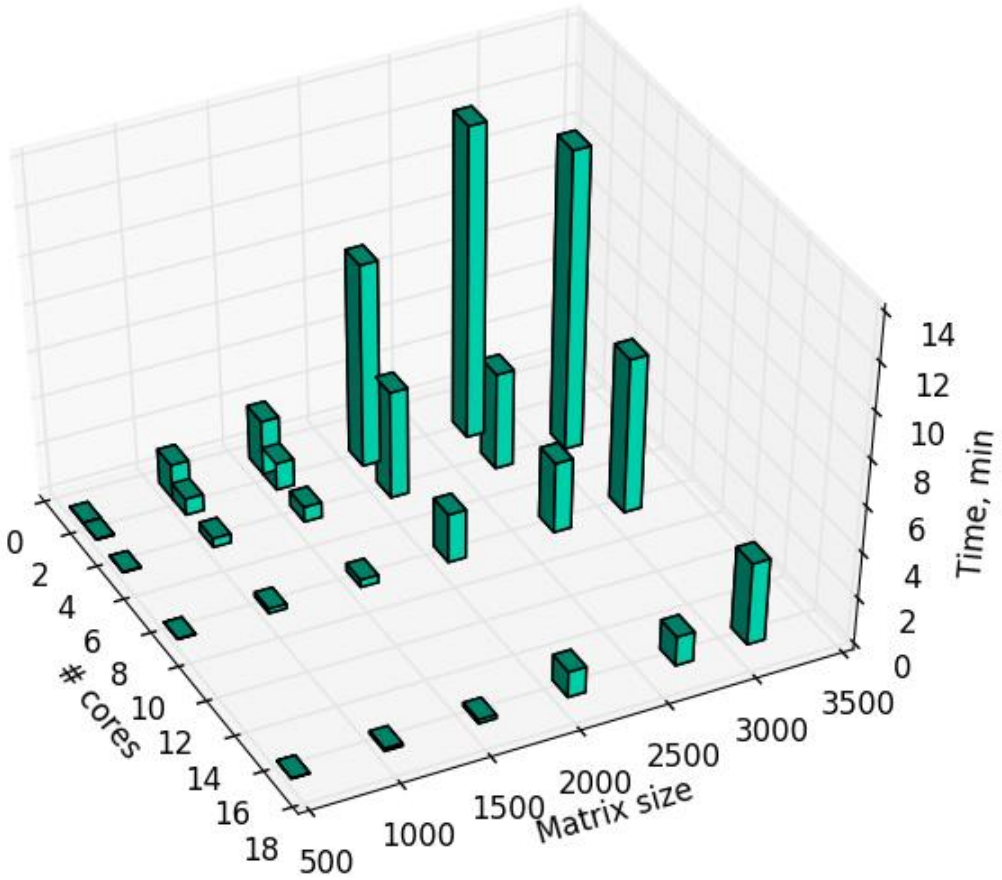
--

*** OpenMP on laptop ***

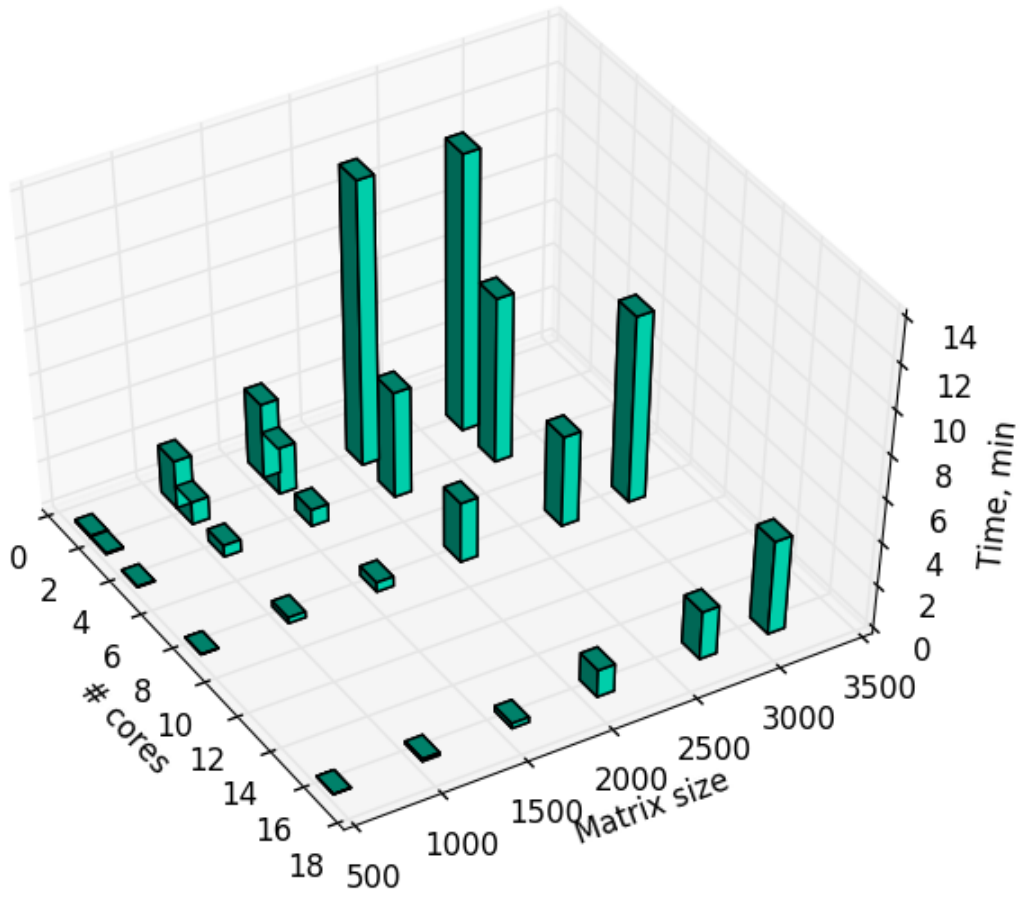
# cores \ size	512	1024	1536	2048	2560
1	0.0050625841	0.0993038082	0.1539202531	1.6225423028	1.9629396455
2	0.0024762013	0.0256568672	0.0670999968	0.9166809965	0.7502034421
4	0.0019306760	0.0221467963	0.0454794165	0.7291778088	0.4920051545
8	0.0022465396	0.0247184336	0.0427601747	0.7329968622	0.6516332081
256	0.0017316641	0.0130488024	0.0369737692	0.6317382681	0.6192644079
512	0.0022095709	0.0132159678	0.0365475313	0.5801013719	0.4832582133

3.2 3D-графики

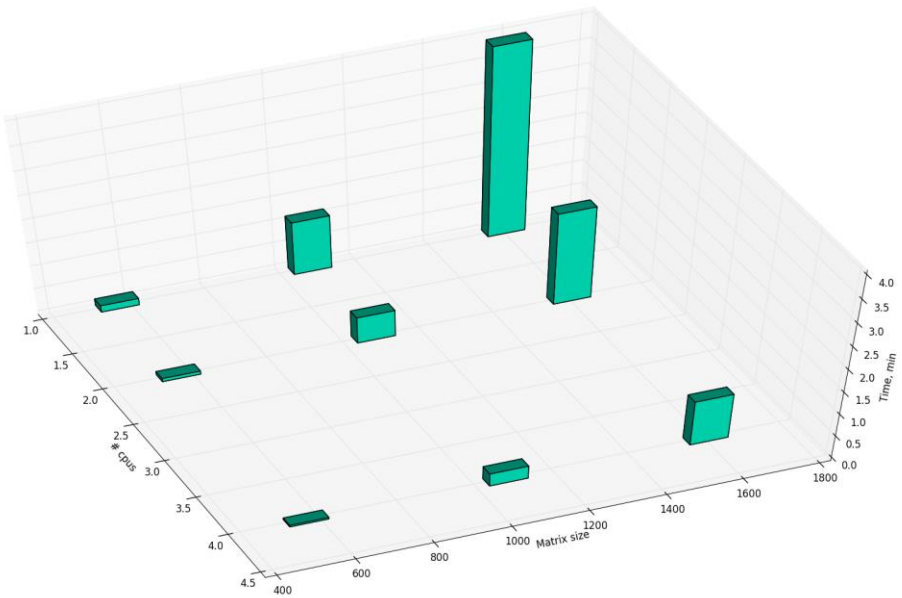
3.2.1 OpenMP на Regatta



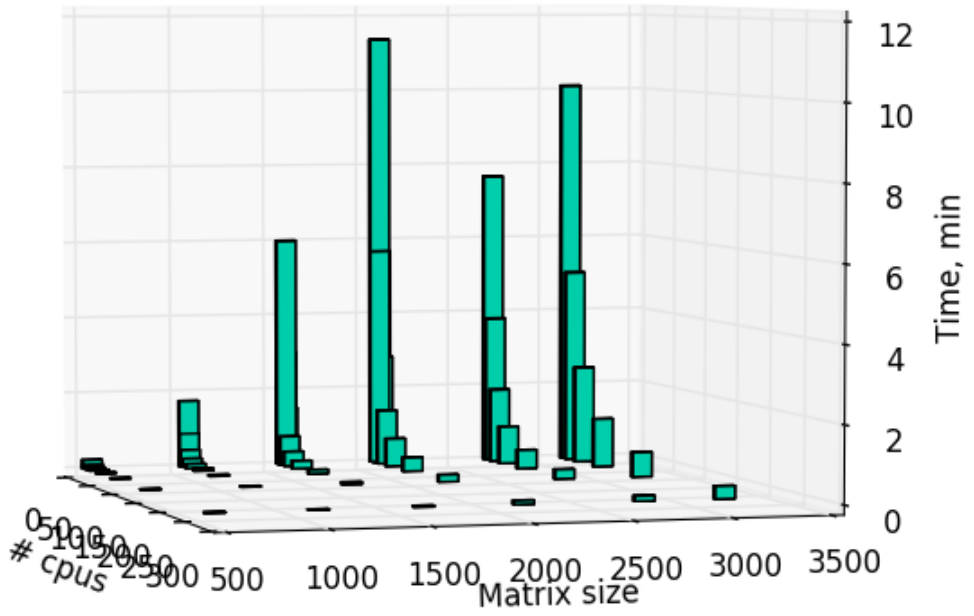
3.2.2 MPI на Regatta



3.2.3 OpenMP на Bluegene

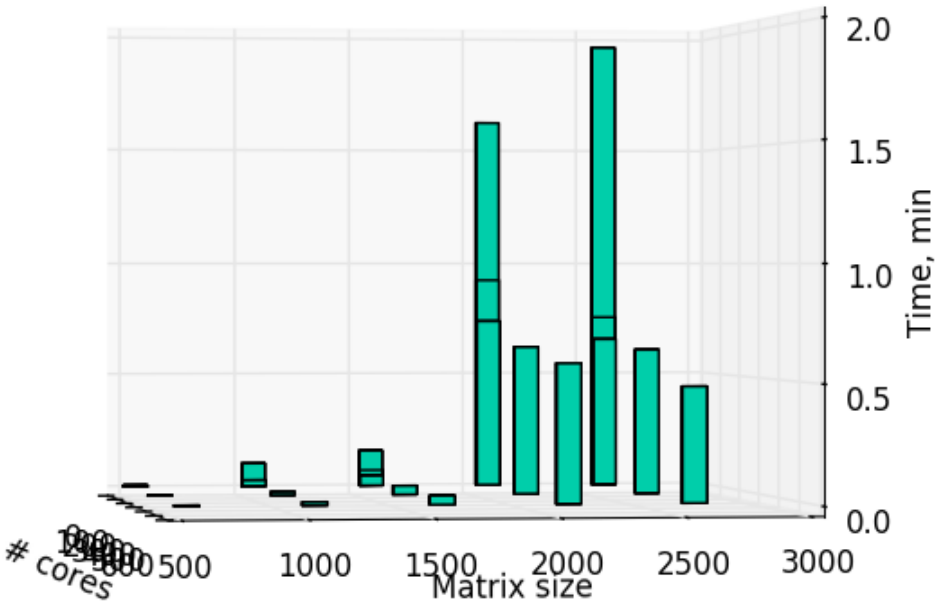


3.2.4 MPI на Bluegene



Замечание к графику: может показаться, что на размере матрицы в 2560 строк скорость выполнения выросла. Это не так: на графике нету данных по вычислению на 1 процессоре по причине превышения ограничения по времени (см. таблицу).

3.2.5 OpenMP на ноутбуке



4 Анализ результатов

Bluegene показал на MPI-версии лучшую производительность. Так как система заточена под прогопроцессорные вычисления (множество относительно слабых процессоров), результаты запуска на Bluegene OpenMP-версии не впечатляют. В практических целях OpenMP (да и любые другие многопоточные (multithread) программы) следует выполнять на Regatta.

Заметим, что задача прекрасна поддалась распараллеливанию и зависимость скорости работы от числа вычислителей близка к линейной. Также обратим внимание, что программа запускалась только с числом потоков / процессов, равным степеням 2. Для n != 2^m ожидается спад производительности.

В сравнении с временем работы на ноутбуке принципиальный выигрыш дает выполнение на 256 вычислителях Bluegene. Заметим, что на ПК уже зависимость времени работы от числа потоков не линейная, а переход от 4 потоков (число ядер процессора, Intel Hiperthreading отсутствует) к 8 сопровождается спадом производительности. На суперкомпьютерах данную ситуацию поймать не удалось.

5 Выводы

Выполнена работа по разработке параллельной версии алгоритма ленточного умножения матриц. Изучены технологии написания параллельных алгоритмов OpenMP и MPI. Проанализировано время выполнения алгоритмов на различных вычислительных системах.

Технология OpenMP крайне удобна в использовании, причем дает колоссальный прирост производительности на рассчитанных на многопоточные вычисления системах, в том числе и на персональных компьютерах.

MPI можно назвать более низкоуровневой технологией: разработка MPI-программы знакомит с основами взаимодействия вычислительных узлов суперкомпьютера. При этом MPI заточена именно на многопроцессорные системы и наибольшую скорость работы показала именно MPI-реализация, запущенная на наибольшем числе вычислителей суперкомпьютера Bluegene.