Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе**

Дисциплина «Параллельные вычисления»

«Разработка программ с использованием библиотек pthread и OpenMP»

Выполнил

студент гр. 13541/2

Никитенко А.П.

Преподаватель

Стручков И.В.

«\_\_»\_\_\_\_\_\_\_\_\_2017г.

Санкт-Петербург

2017

Цель работы

Научиться создавать программы с использованием многопоточных технологий. Познакомиться с работой библиотек pthread и OpenMP для языка C++. Проанализировать прирост производительности при использовании многопоточных библиотек.

Постановка задачи

Задание №7 «Умножение двух матриц».

Решить следующую задачу тремя способами:

1. Однопоточной программой.

2. Программой с использованием библиотеки pthread.

3. Программой с использованием библиотеки OpenMP.

Ход работы

Задание выполнялось на компьютере со следующим оборудованием:

Процессор Intel Core i-7 4510U – двух ядерный процессор с поддержкой hyper-threading (4 логических потока).

Операция умножения двух матриц вводится только для случая, когда число столбцов первой матрицы равно числу строк второй матрицы. Произведением матрицы Аm×n на матрицу Вn×p, называется матрица Сm×p такая, что сik = ai1 × b1k + ai2 × b2k + ... + ain × bnk, т. е. находиться сумма произведений элементов i - ой строки матрицы А на соответствующие элементы j - ого столбца матрицы В. Если матрицы А и В квадратные одного размера, то произведения АВ и ВА всегда существуют. Поэтому, при выполнении данного задания подразумевается, что перемножаемые матрицы квадратные.

Для умножения матриц в последовательной программе была написана функция *mult*, на вход которой подаются размер матриц *size* одним числом, перемножаемые матрицы и матрицы, которая будет хранить результат операции. Код функции представлен в листинге 1.

**Листинг 1. Однопоточное решение задачи**

|  |
| --- |
| void mult(int \*\*M1, int \*\*M2, int \*\*M3, int size){ //Умножение матриц  for (int i=0; i<size; i++)  for (int j=0; j<size; j++)  {  int res=0;  for (int k=0; k<size; k++)  {  res+=M1[i][k]\*M2[k][j];  }  M3[i][j]=res;  }  } |

Многопоточная реализация задачи с использованием библиотеки pthread

Для реализации многопоточного выполнения матриц функция *mult* была изменена таким образом, чтобы обеспечить параллельную работу нескольких потоков. Для этого была создана структура *args\_t*, подаваемая на вход функции. В этой структуре содержатся, как и раньше, размер матриц, перемножаемые матрицы и матрица результата, а также добавились переменные *start* и *end*, которые указывают каждому отдельному потоку диапазон строк матрицы А, с которыми будет работать поток. Код измененной функции представлен в листинге 2.

Листинг 2. Многопоточная реализация с использованием библиотеки pthread.

|  |
| --- |
| void\* mult(void \*argg){ //Умножение матриц  args\_t \*arg=(args\_t\*) argg;  int start=arg->start;  int end=arg->end;  int size=arg->size;    for (int i=start; i<end+1; i++)  for (int j=0; j<size; j++)  {  int res=0;  for (int k=0; k<size; k++)  {  res+=arg->A[i][k]\*arg->B[k][j];  }  arg->C[i][j]=res;  }    return NULL;  } |

Многопоточная реализация с использованием библиотеки OpenMP

Код многопоточной реализации с использованием библиотеки OpenMP представлен в листинге 3.

Листинг 3. Многопоточная реализация с использованием библиотеки OpenMP.

|  |
| --- |
| void mult(int \*\*M1, int \*\*M2, int \*\*M3, int size, int num\_t){ //Умножение матриц  omp\_set\_num\_threads(num\_t);  #pragma omp parallel for shared(M1, M2, M3)  for (int i=0; i<size; i++)  for (int j=0; j<size; j++)  {  int res=0;  for (int k=0; k<size; k++)  {  res+=M1[i][k]\*M2[k][j];  }  M3[i][j]=res;  }  } |

Многократный запуск программ и подсчет вероятностных характеристик

Для расчета вероятностных характеристик каждая программа запускалась 50 раз на одинаковом наборе перемножаемых матриц 1000х1000. На основе полученных данных подсчитывается математическое ожидание, дисперсия и доверительных интервал. Данные представлены в таблице 1.

**Таблица 1. Таблица результатов для 50 запусков каждой реализации.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Реализация | Число потоков | Мат. ожидание | Дисперсия | Доверительный интервал 0.95% |
| Single Thread | 1 | 7023 | 968407 | [6746.89 – 7299.11] |
| pThread | 1 | 7468 | 545061 | [7260.85 – 7675.15] |
|  | 2 | 4805 | 447778 | [4617.25 – 4992.75] |
|  | 4 | 4195 | 506 | [4188.69– 4201.31] |
|  | 8 | 4354 | 30414 | [4305.07 – 4402.93] |
| OpenMP | 1 | 5890 | 67186 | [5817.27– 5962.73] |
|  | 2 | 4515 | 160679 | [4402.53 – 4627.47] |
|  | 4 | 4486 | 535606 | [4382.94 – 4589.06] |
|  | 8 | 4986 | 207481 | [4858.2 – 5113.8] |

Из таблицы 1 видно, что многопоточная реализация задания с помощью библиотек pThread и OpenMP позволяет существенно увеличить скорость выполнения программы, при этом наилучшие результат достигаются при 4 параллельно работающих потоках, сокращая время выполнения задачи примерно на 35-40%. При использовании 8 потоков прироста быстродействия не наблюдается, т.к. используемый процессор имеет только 4 логических потока.

Вывод

В ходе работы создано три программы на языке С++ для решения задачи умножения квадратных матриц. Предложенные решения позволяют умножать квадратные матрицы больших размеров. В ходе работы изучены и использованы библиотеки многопоточного программирования pThread и OpenMP. Реализованные на их основе решения, способны изменять количество исполняющих потоков. В результате экспериментов было установлено, что наибольший прирост быстродействия был получен при использовании 4 потоков в обоих случаях. Полученные реализации протестированы на матрице и проведен анализ вероятностных характеристик. В результате экспериментов установлено, что при использовании 1-2 потоков библиотека OpenMP показывает лучшие результаты, но при использовании явного задания 4 потоков наибольший эффект достигается с библиотекой pThread.