Лабораторные работы по курсу "Параллельное и распределённое программирование"

Игорь Комолых, Сергей Лущик

23 мая 2018 г.

1 Умножение матриц

Эта лабораторная работа заключалась в сравнении последовательной и параллельной реализации алгоритмов умножения матриц, а так же в сравнении времени работы программы при разных способах обхода массива.

В результате выполнения работы были получены:

- описанный класс Matrix
- bash и sbatch файлы запуска программы на персональных компьютерах и кластере САФУ
- python-скрипт для построения графиков из полученных данных.

Результатом выполнения программы является строка, в которой через запятую указаны количество используемых потоков, размерность квадратной матрицы, время работы. При запуске bash или sbatch сценариев, происходит формирование CSV-файла, по данным которого в дальнейшем можно строить графики ускорения, эффективности и времени работы (см. Рис. 1 и 2). Программы собирались и запускались на вычислительном кластере САФУ.

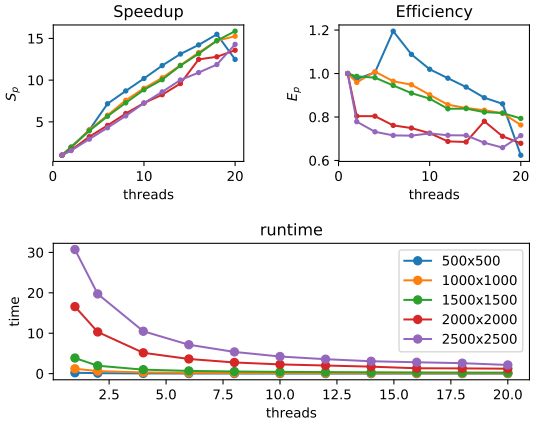

Рис. 1: графики до смены порядка обхода матриц.

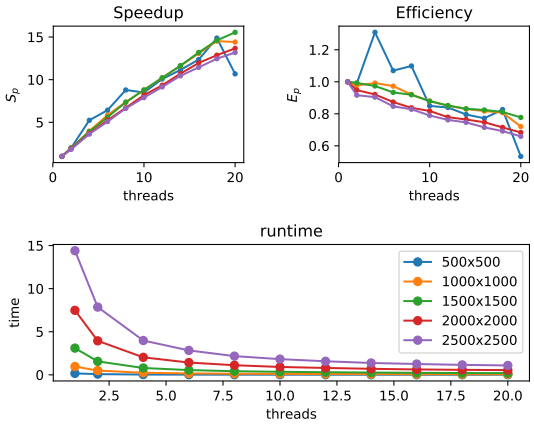

Рис. 2: графики после смены порядка обхода матриц.

По рис. 1 и 2 видно, что после изменения порядка обхода матрицы с привычного “строки-столбцы” на “столбцы-строки” (см. Листинг 1) время работы программы может сокращаться в 2 и более раза, в некоторых случаях удавалось достичь ускорения в 4-5 раз.

Данный пример демонстрирует особености устройства кэша процессора и оперативной памяти. При обращении к какой-либо ячейке памяти, в кэш вместе с ней загружаются и несколько соседних ячеек. При обращении в порядке “строки-столбцы” два элемента, над которыми производятся операции в смежных итерациях алгоритма, в памяти будут находиться на расстоянии, равном размеру строки матрицы. Если же обходить массивы в порядке “столбцы-строки”, смежные итерации будут опрерировать элеентами одной строки матрицы, элементы которой располагаются в памяти друг за другом.

```
//
// строки-столбцы
//

#include <iostream>
  Matrix mulParallel(const Matrix& first, const Matrix& second) {
    Matrix result(first.rows(), second.cols());
    if (first.cols() == second.rows())
      #pragma omp parallel for shared(result, first, second)
      for (size_t i = 0; i < result.rows(); ++i)
        for (size_t j = 0; j < result.cols(); ++j) {
          result(i, j) = 0;
          for (size_t k = 0; k < result.rows(); ++k)
            result(i, j) += first(i, k) * second(k, j);
        }
    else
      throw std::invalid_argument("Wrong dimensions");

    return result;
  }

  //
  // столбцы-строки
  //

  Matrix mulParallel2(const Matrix& first, const Matrix& second) {
    Matrix result(first.rows(), second.cols());
    if (first.cols() == second.rows()) {
      #pragma omp parallel for shared(result, first, second)
      for (size_t j = 0; j < result.cols(); ++j)
        for (size_t i = 0; i < result.rows(); ++i) {
          result(i, j) = 0;
          for (size_t k = 0; k < result.rows(); ++k)
            result(i, j) += first(i, k) * second(j, k);
        }
    }
    else
      throw std::invalid_argument("Wrong dimensions");

    return result;
  }
```

Листинг 1: Два способа обхода матрицы

2 Задача Дирихле для уравнения Пуассона

С использованием класса матриц, полученного в ходе выполения первой лабораторной были реализованны последовательный и параллельный алгоритмы решения задачи Дирихле для уравнения Пуассона, написаны python-скрипты для построения графиков поверхностей по полученным данным. Результат тестового запуска алгоритма для уравнения $f(x,y) = 4$ с краевыми условиями $g(x,y) = (x-0.5)^2 + (x-0.5)^2$ представлен на рис. 3.

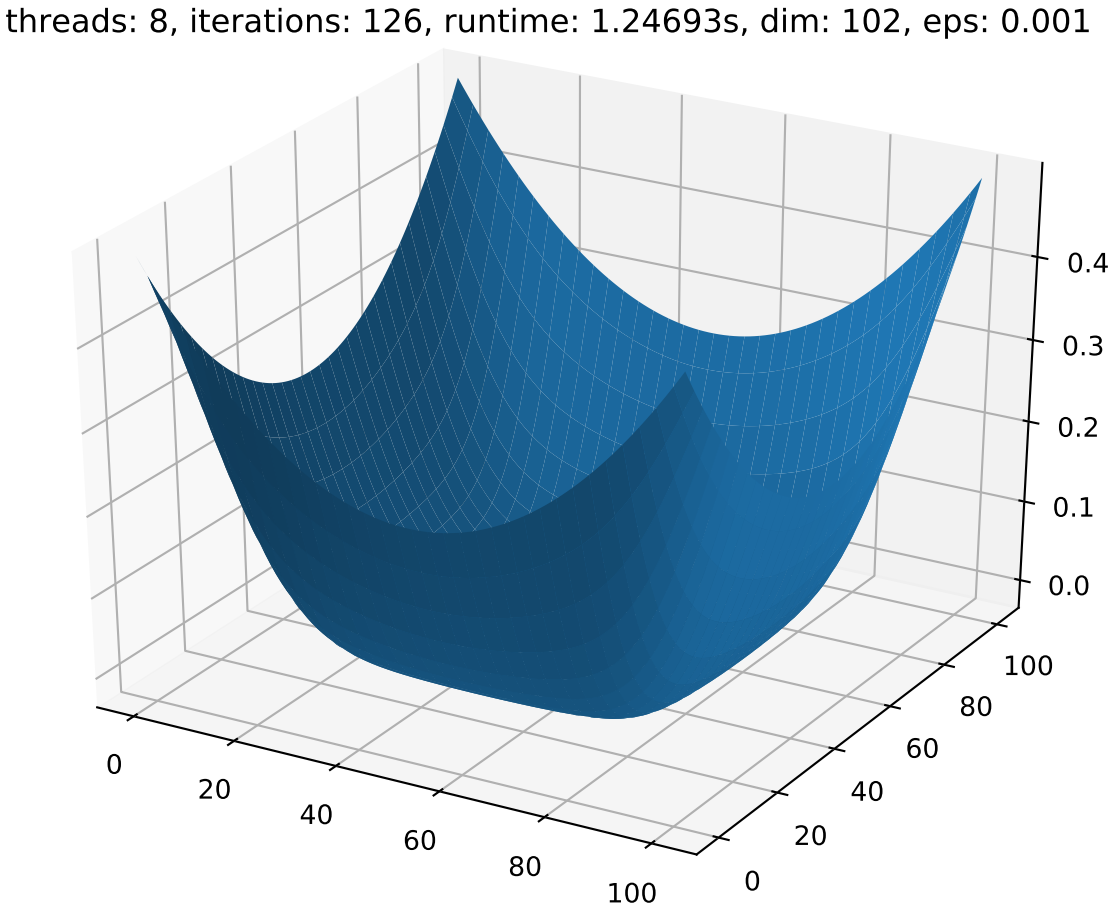

Рис. 3: График решения задачи Дирихле для уравнения Пуассона.