Параллельное и распределённое программирование

Отчет

Лабораторная работа № 1

Автор:
Поташев Н. А.

Руководитель:
Рудалев А. В.

28 мая 2018 г.

1 Постановка задачи

Целью данной лабораторной работы было сравнение параллельной и последовательной реализации алгоритма умножения матриц.

2 Выполнение работы

В ходе выполнения данной работы были получены:

- программа, реализующая параллельное и последовательное умножение матриц;
- класс Matrix, который описывает 2D матрицы;
- python-скрипт, реализующий построение графиков на основе полученных данных;
- bash-скрипт для запуска программы на персональном компьютере;
- sbatch-скрипт для запуска программы на кластере;
- результаты расчетов программы на кластере и персональном компьютере(в формате csv-файлов).

2.1 Алгоритмы

Был реализован сначала последовательный ленточный алгоритм умножения матриц, а потом он был распараллелен см. Листинг 1 и Листинг 2.

```
  ...
if (A.cols() == B.rows()) {
  for (i = 0; i< C.cols(); ++i){
    for (j = 0; j < C.cols(); ++j){
      for (k = 0; k < C.rows(); ++k){

        C(i,j)+= A(i,k) * B(k,j);
      }
    }
  }
}
else
  throw std::invalid_argument("wrong dims!");
  ...
```

Листинг 1: реализация последовательного алгоритма

```
  ...
size_t i,j,k;
double localResult=0;
if (A.cols() == B.rows()) {
  #pragma omp parallel for private(i,j,k,localResult)
  for (i = 0; i< C.cols(); ++i){
    for (j = 0; j < C.cols(); ++j){
      for (k = 0; k < C.rows(); ++k){
        localResult+= A(i,k) * B(k,j);
      }
      C(i,j)=localResult;
    }
  }
}
else
  throw std::invalid_argument("wrong dims!");
  ...
```

Листинг 2: реализация параллельного алгоритма

2.2 Результаты расчетов

Результатом выполнения программы был текстовой csv-файл с большим количеством опытов. Были выполнены расчеты на двух ПК и кластере, также были нарисованы графики времени работы программы, эффективности, ускорения при распараллеливании. Также для оптимизации вычислений были изменены вложенности циклов при умножении матриц.

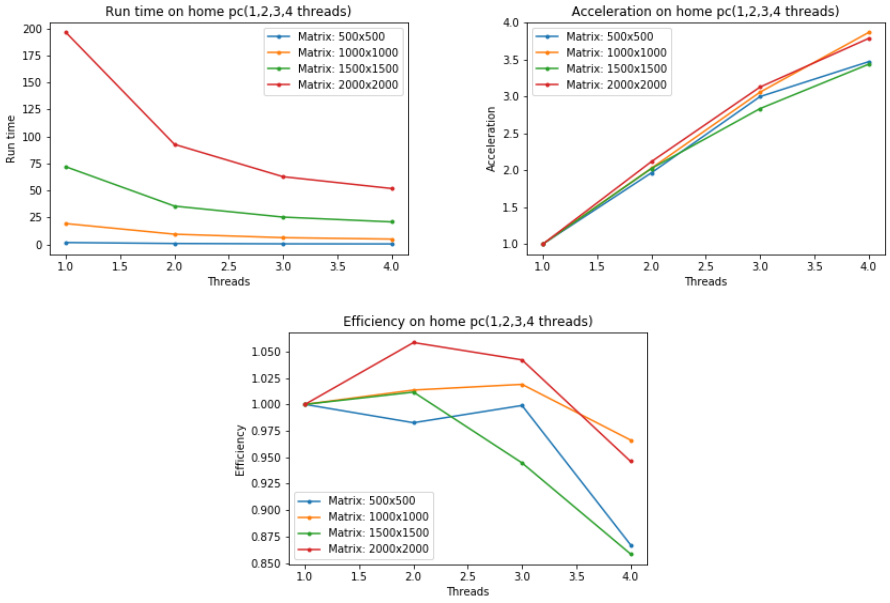

Рис. 1: графики результатов работы на домашнем ПК

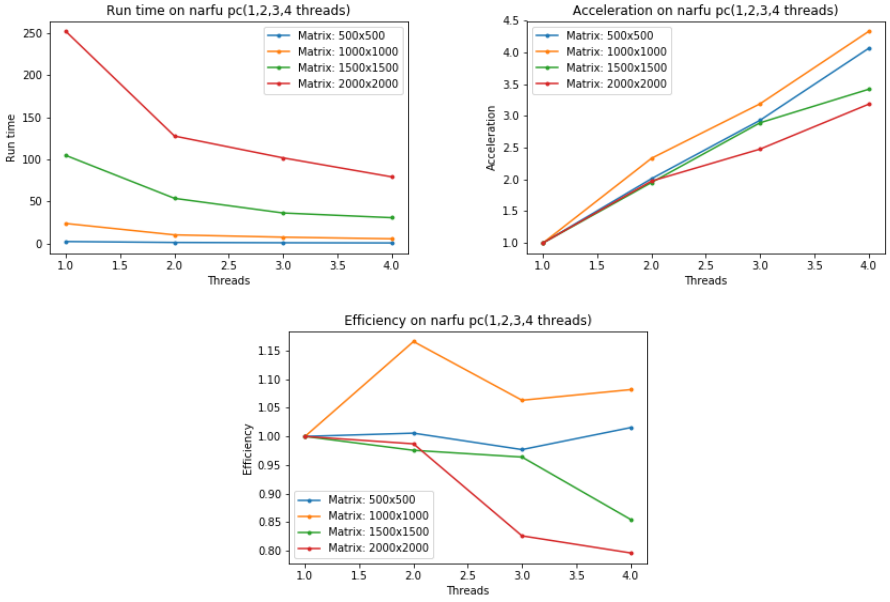

Рис. 2: графики результатов работы ПК(ауд.301)

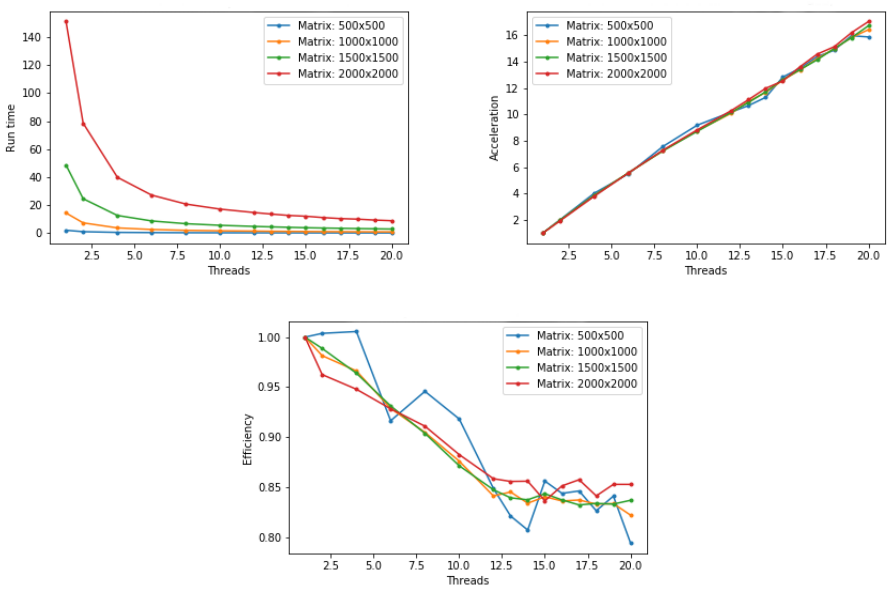

Рис. 3: графики результатов работы на кластере САФУ

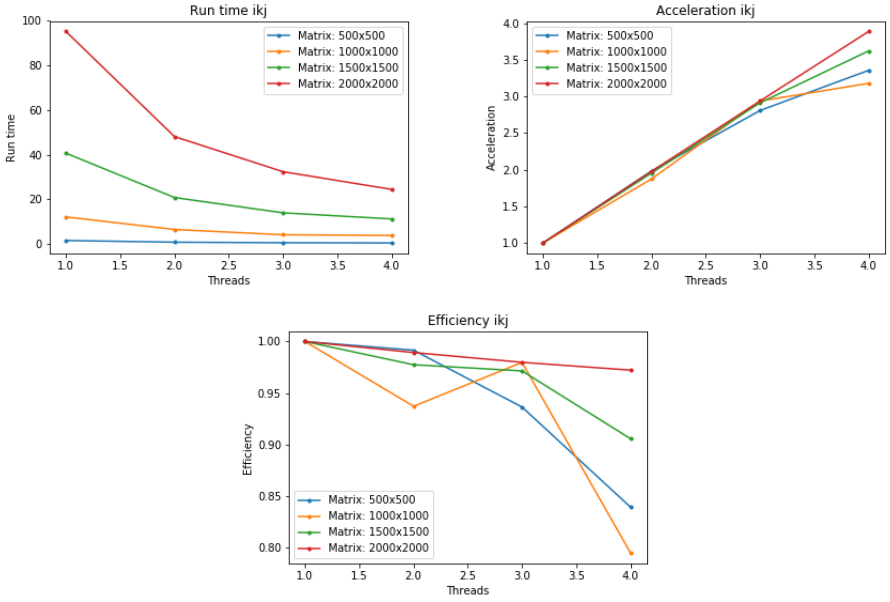

Рис. 4: графики результатов работы при смене переменных цикла на IKJ

3 Вывод

По рис. 1-3 можно сделать вывод, что параллельный алгоритм умножения матриц намного эффективнее и быстрее последовательного. Также были замечены "скачки"в графиках эффективности. Данные "скачки"демонстрируют принцип работы кэша процессора. Когда мы выполняем расчеты, в кэш заносится не один информационный элемент, а сразу целый блок, что ускоряет работу программы. При изменении организации циклов наиболее быстро будет выполняться умножение матриц с организацией внутреннего цикла по переменной j (IKJ, KIJ). По моему мнению, так происходит из-за особенности выделения памяти (память выделяется по строкам, и обращение к следующему элементу строки гораздо быстрее, чем обращение к следующему элементу столбца). Таким образом, зная, как выделяется память в C++, можно ускорить работу своей программы.