Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Дисциплина: Параллельное программирование

Лабораторная работа №1

Выполнил: Го Цзыхань

Группа: P33201

Санкт-Петербург

2023г.

1. Описание решаемой задачи

1) На языке Си написать консольную программу lab1.c, решающую задачу, указанную в (см. ниже).
2) Скомпилировать написанную программу без использования автоматического распараллеливания с помощью следующей команды: /home/user/gcc -O3 -Wall -Werror -o lab1-seq lab1.c
3) Скомпилировать написанную программу, используя встроенное в gcc средство автоматического распараллеливания Graphite с помощью следующей команды “/home/user/gcc -O3 -Wall -Werror -floopparallelize-all -ftree-parallelize-loops=K lab1.c -o lab1-par-K” (переменной K поочерёдно присвоить хотя бы 4 значения: 1, меньше числа физических ядер, равное числу физических ядер и больше числа физических ядер). В результате получится одна нераспараллеленная программа и четыре или более распараллеленных.
4) Запускать файл lab1-seq из командной строки, увеличивая значения N до значения N1, при котором время выполнения превысит 0.01 с. Подобным образом найти значение N=N2, при котором время выполнения превысит 5 с.
5) Используя найденные значения N1 и N2, выполнить следующие эксперименты (для автоматизации проведения экспериментов рекомендуется написать скрипт):
   • запускать lab1-seq для значений N = N1, N1 + Δ, N1 + 2Δ, N1 + 3Δ, . . . , N2 и записывать получающиеся значения времени delta_ms(N) в функцию seq(N);
   • запускать lab1-par-K для значений N = N1, N1 + Δ, N1 + 2Δ, N1 + 3Δ, . . . , N2 и записывать получающиеся значения времени delta_ms(N) в функцию par − K(N);
   • значение Δ выбрать так: Δ = (N2 − N1)/10.
6) Провести верификацию значения X. Добавить в конец цикла вывод значения X и изменить количество экспериментов на 5. Сравнить значения X для распараллеленной программы и не распараллеленной.
7) Найти вычислительную сложность алгоритма до и после распараллеливания, сравнить полученные результаты.

указанный вопрос:

A = 2 × 7 = 14

1) Этап Generate. M1(1-14), M2(1-140)
2) Этап Map.
   M1 Экспонента квадратного корня
   M2 Десятичный логарифм, возведенный в степень e
3) Этап Merge. Деление (т.е. M2[i] = M1[i]/M2[i])
4) Этап Sort. Сортировка расчёской (Comb sort).
5) Этап Reduce.

2. лабораторная среда

   Intel(R) Core(TM) i7-9750H CPU @ 2.6GHz

   Virtualbox Ubuntu 20.04

   gcc version 9.4.0

3. lab1.c

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

#define A 14

void comb_sort(float *data, int n)
{
  const float shrink = 1.25;
  int i, delta = n, noswap = 0;
  while (!noswap)
  {
    for (noswap = 1, i = 0; i + delta < n; i++)
      if (data[i] > data[i + delta])
      {
        float swap;
        swap = data[i];
        data[i] = data[i + delta];
        data[i + delta] = swap;
        noswap = 0;
      }
    if (delta > 1)
    {
      delta /= shrink;
      noswap = 0;
    }
  }
}

int main(int argc, char *argv[])
{
  int i, N;
  struct timeval T1, T2;
  long delta_ms;
  N = atoi(argv[1]);        /* N равен первому параметру командной строки
*/gettimeofday(&T1, NULL);  /* запомнить текущее время T1 */
  for (i = 0; i < 100; i++) /* 100 экспериментов*/
  {
    srand(i);
    /* инициализировать начальное значение ГСЧ */
    unsigned int seed = time(NULL);
    /* Заполнить массив исходных данных размером N */
    /* Решить поставленную задачу, заполнить массив с результатами */
    float M1[N];
    float M2[N / 2];
    float M2CP[N / 2];
    // M1
    for (int j = 0; j < N; j++)
    {
      M1[j] = rand_r(&seed) % A + 1;
      M1[j] = exp(sqrt(M1[j]));
    }
```

```c
    // M2
    for (int j = 0; j < N / 2; j++)
    {
      M2[j] = rand_r(&seed) % (9 * A + 1) + A;
      M2CP[j] = M2[j];
      if (j != 0)
      {
        M2[j] = M2[j] + M2CP[j - 1];
      }
      M2[j] = log10(M2[j]);
    }

    /* Этап Merge */
    for (int j = 0; j < N / 2; j++)
    {
      M2[j] = M1[j] / M2[j];
    }

    /* Отсортировать массив с результатами указанным методом*/
    comb_sort(M2, N / 2);

    /* Этап Reduce */
    float min = 0;
    int k = 0;
    int temp = 0;
    float X = 0;

    while (min == 0)
    {
      min = (fabs(M2[k]) < 0.00001) ? M2[k] : 0;
      k++;
    }
    for (int j = 0; j < N / 2; j++)
    {
      temp = (int)(M2[j] / min);
      if (temp % 2 == 0)
      {
        X = X + sin(M2[j]);
      }
    }
    // printf("X: %fd\n", X); /* T2 -T1 */
  }

  gettimeofday(&T2, NULL); /* запомнить текущее время T2 */
  delta_ms = 1000 * (T2.tv_sec - T1.tv_sec) + (T2.tv_usec - T1.tv_usec) /
1000;printf("\n N=%d. Milliseconds passed: %ld\n", N, delta_ms); /* T2 - T1 */
  return 0;
}
```

4. Содержание отчета

N1 = 1500
N2 = 570000
Δ = (570000 - 1500) / 10 = 56850

```bash
#!/bin/bash
echo -e "It is lab1-seq \v"
for ((var = 1500; var <= 570000; var += 56850));
do  ./lab1-seq $var
done

echo -e "It is lab1-par-1 \v"
for ((var = 1500; var <= 570000; var += 56850));
do  ./lab1-par-1 $var
done

echo -e "It is lab1-par-2 \v"
for ((var = 1500; var <= 570000; var += 56850));
do  ./lab1-par-2 $var
done

echo -e "It is lab1-par-4 \v"
for ((var = 1500; var <= 570000; var += 56850));
do  ./lab1-par-4 $var
done

echo -e "It is lab1-par-6 \v"
for ((var = 1500; var <= 570000; var += 56850));
do  ./lab1-par-6 $var
done
```

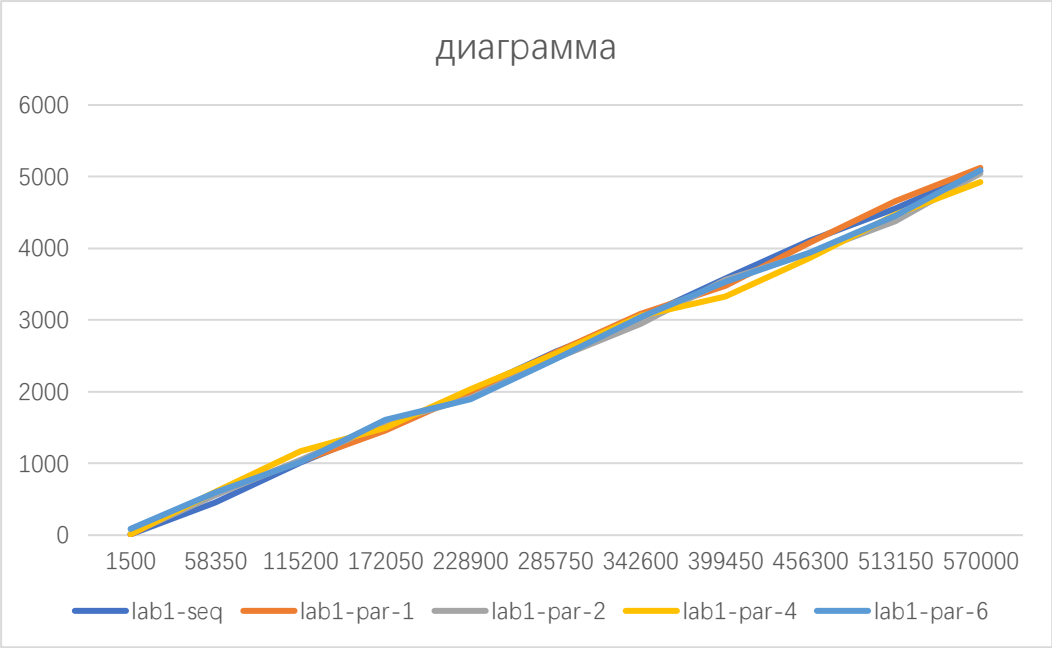
Таблица 1. Время выполнения программы в зависимости от N и количества потоков

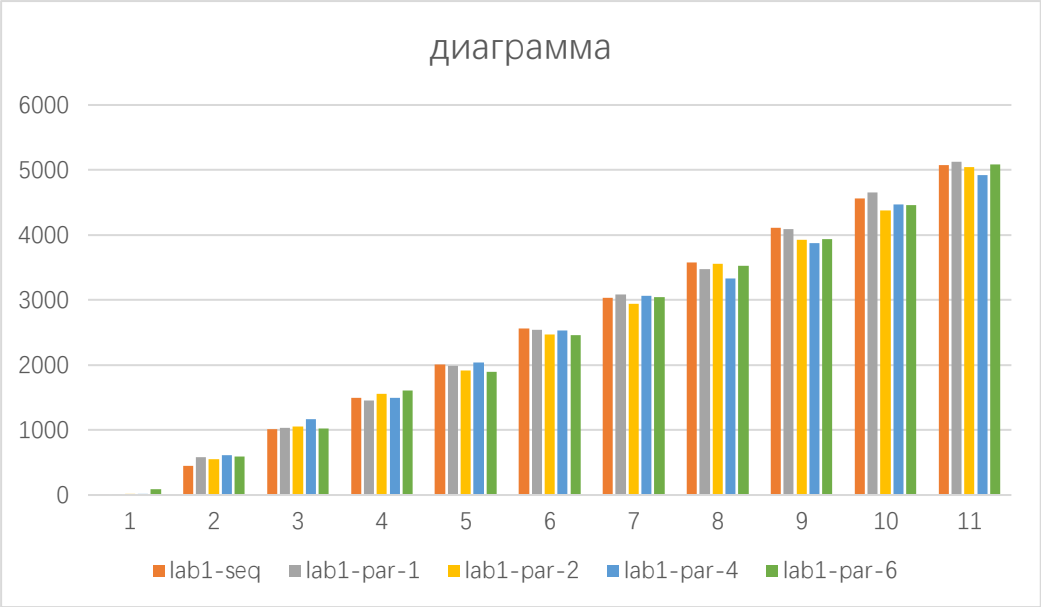
N	lab1-seq	lab1-par-1	lab1-par-2	lab1-par-4	lab1-par-6
1500	11	10	21	13	88
58350	451	586	556	608	590
115200	1014	1032	1049	1169	1020
172050	1493	1455	1558	1493	1608
228900	2010	1991	1919	2035	1899
285750	2561	2545	2465	2530	2456
342600	3029	3084	2945	3063	3039
399450	3578	3480	3558	3332	3530
456300	4110	4086	3929	3873	3940
513150	4558	4656	4378	4475	4458
570000	5077	5123	5047	4925	5090

Таблица 2. Значения X при различных N

N	lab1-seq	lab1-par-1	lab1-par-2	lab1-par-4	lab1-par-6
1500	355.124023	322.069061	190.621567	402.178314	310.694763
58350	11699.4688	12149.9229	12488.8555	11627.2695	11852.3301
115200	23743.0762	24303.9199	24053.3047	23941.3262	23574.0176
172050	36513.8359	37040.1328	33887.5664	22503.998	36184.4414
228900	48799.5508	48450.0781	45170.0977	48421.3711	47736.9063
285750	61171.0195	60210.4375	55926.2148	58409	57305.4219
342600	71982.0469	72132.3125	68486.9531	70435.6719	70245.375
399450	83022.0156	80945.6641	79502.125	83051.8047	55090.7422
456300	97298.7422	96426.4766	92311.0156	94247.2188	93224.3125
513150	109038.75	105255.57	107059.281	108919.953	105587.07
570000	120992.844	118004.281	119228.391	118141	116103.258

Gcc функций seq(N), par-K(N) диаграмма





5. Выводы

В ходе выполнения лабораторной работы мы получили несколько программ – одну последовательную и 4 параллельных, выполняющихся на 1, 2, 4, 6 потоках.

В результате выполнения этих программ были получены результаты, указанные в таблицах, на основе которых можно сделать следующие выводы:

1) Чем больше размерность массива обрабатываемых данных, тем меньше пользы от распараллеливания программ.
2) Результат выполнения программ может незначительно отличаться в зависимости от компилятора за счет различных принципов выполнения некоторый действий (например, округления и приведения типов).