МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

«Параллельная реализация решения системы линейных алгебраических уравнений с помощью OpenMP»

студентки 2 курса, группы 21207

Черновской Яны Тихоновны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А.Ю. Власенко

Новосибирск 2023

СОДЕРЖАНИЕ

ЦЕЛЬ

Изучить стандарт для распараллеливания программ на языке Си openMP

ЗАДАНИЕ

1. Последовательную программу из предыдущей практической работы, реализующую итерационный алгоритм решения системы линейных алгебраических уравнений вида Ax=b, распараллелить с помощью OpenMP.

   ОБЯЗАТЕЛЬНОЕ УСЛОВИЕ: создается одна параллельная секция #pragma omp parallel, охватывающая весь итерационный алгоритм.

2. Замерить время работы программы на кластере НГУ на 1, 2, 4, 8, 12, 16 потоках. Построить графики зависимости времени работы программы, ускорения и эффективности распараллеливания от числа используемых ядер. Исходные данные и параметры задачи подобрать таким образом, чтобы решение задачи на одном ядре занимало не менее 30 секунд.

3. Провести исследование на определение оптимальных параметров #pragma omp for schedule(...) при некотором фиксированном размере задачи и количестве потоков.

Вариант задания:

Метод простой итерации

В методе простой итерации преобразование решения на каждом шаге задается формулой:

$$x^{n+1} = x^n - \tau(Ax^n - b).$$

Здесь τ – константа, параметр метода. В зависимости от значения параметра τ последовательность $\{x^n\}$ может сходиться к решению быстрее или медленнее, или вообще расходиться. В качестве подходящего значения τ

[1]Общая формула для итерационных методов выглядит следующим образом: $x^{n+1} = f(x^{n+1}, x^n, x^{n-1}, \ldots, x^0)$, но для целей лабораторных работ достаточно будет формулы, представленной в тексте.

можно взять 0.01 или -0.01. Знак параметра τ зависит от задачи. Если с некоторым знаком решение начинает расходиться, то следует сменить его на противоположный. Критерий завершения счета:

$$\frac{\|Ax^n - b\|_2}{\|b\|_2} < \varepsilon,$$

где $\|u\|_2 = \sqrt{\sum_{i=0}^{N-1} u_i^2}$. Для тестирования метода значение ε можно взять равным 10^{-5}.

ОПИСАНИЕ РАБОТЫ

1. Была написана параллельная реализация решения системы линейных алгебраических уравнений с помощью openMP
   Запуск программы на 1, 2, 4, 8, 12, 16 потоках

```
Total time is 53.360254 seconds
Total time is 27.838755 seconds
Total time is 13.053114 seconds
Total time is 7.595445 seconds
Total time is 4.871618 seconds
Total time is 7.177618 seconds
```

3. Были построены графики времени, ускорения и эффективности, где

а) Ускорение: Sp = T1 / Tp, где T1 - время работы на 1 потоке, Tp - время работы параллельной программы на p потоках

б) Эффективность Ep = Sp / p * 100%.

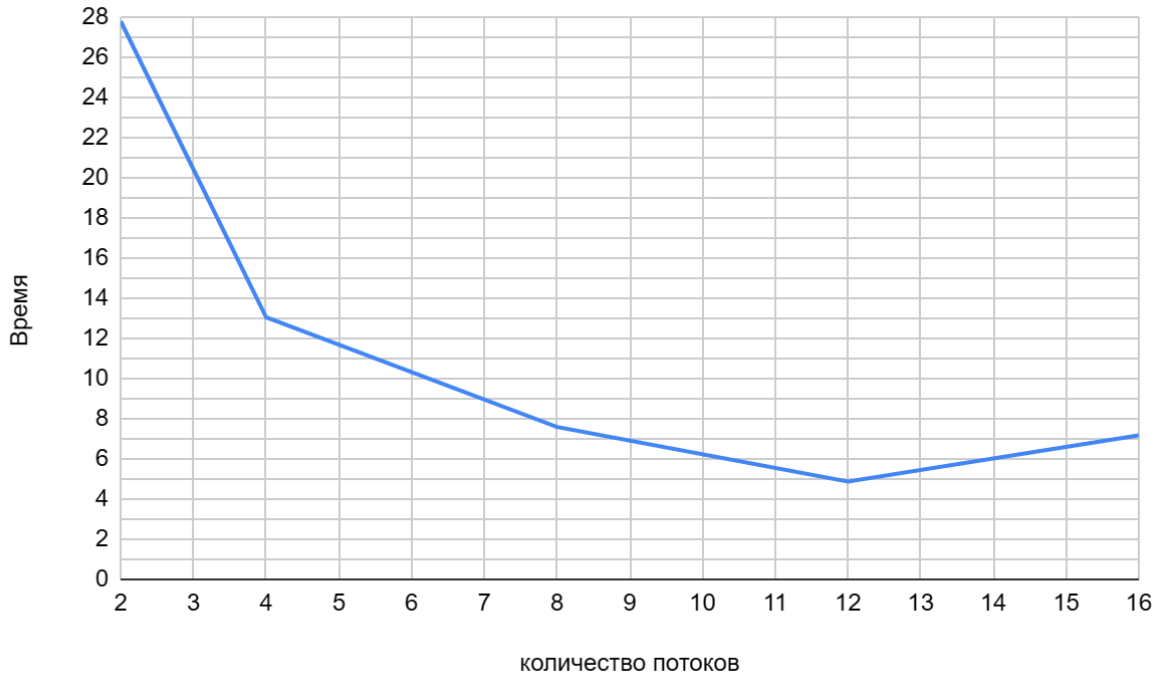

Рис 1. время работы программы

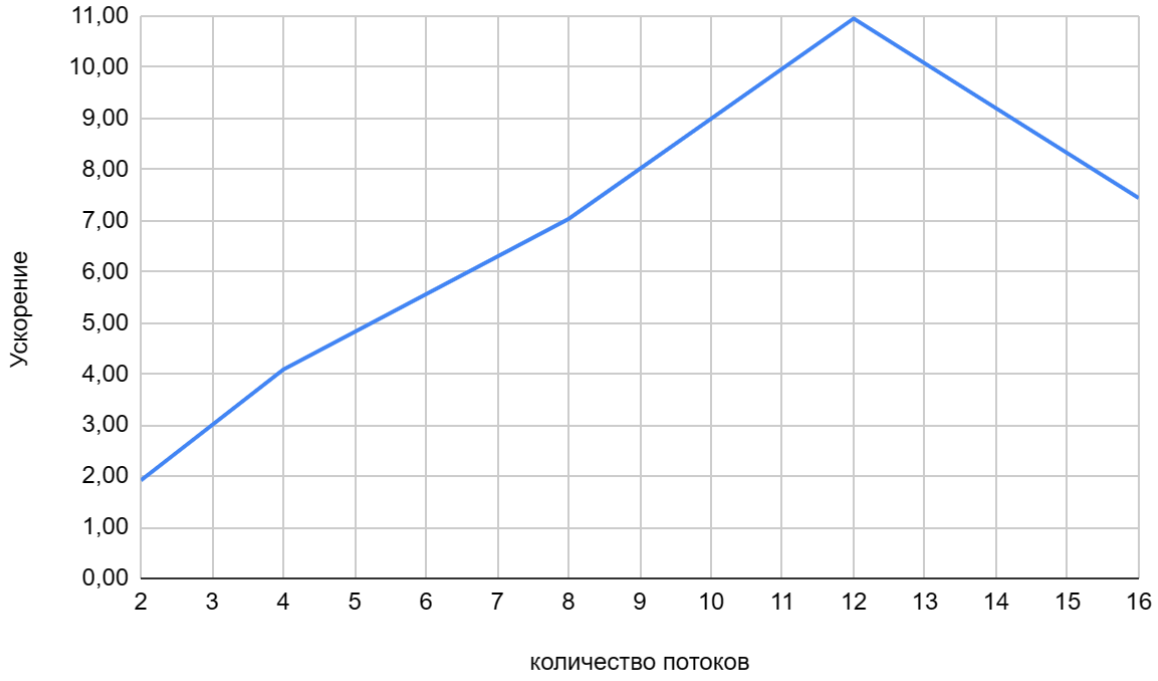

Рис2. ускорение

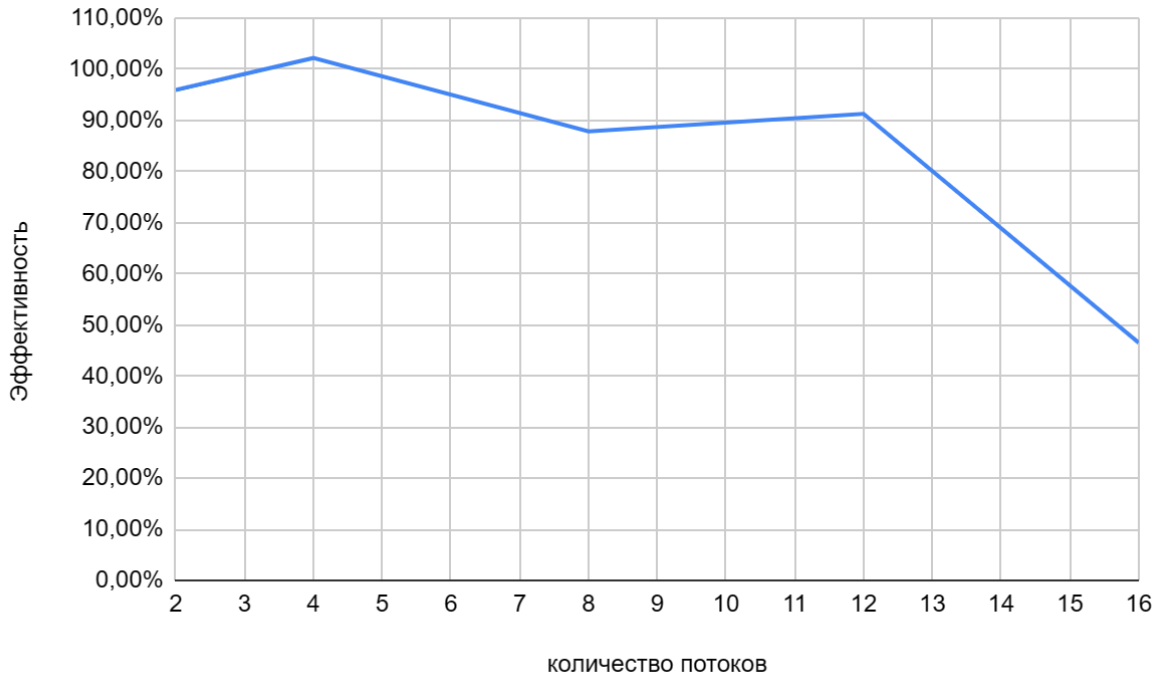

Рис3. эффективность

4. Было проведено исследование на определение оптимальных параметров #pragma omp for schedule(...) при фиксированном размере задачи (N = 1500) и количестве потоков(n = 4)

Chunk	Static	Dynamic	Guided
1	8,61762	12,50098	6,271872
2	7,87122	11,95616	6,274066
3	7,84444	10,92093	6,267952
4	8,567485	9,987578	6,268303
5	7,278471	9,89245	6,21343
6	7,217279	8,590184	6,236471
7	6,471187	8,152327	6,271755
8	6,600594	7,75487	6,263557
9	6,540779	7,011116	6,270854
10	6,419139	6,504414	6,332172
50	6,302023	6,779115	6,310566
100	6,301397	6,454415	8,013385
500	8,00914	7,927176	17,59693
1000	15,60158	17,28352	23,45679
1500	23,12951	28,33795	27,12399

ЗАКЛЮЧЕНИЕ

По полученным данным в таблице можно сделать вывод, что static и quided работает примерно одинаково на большинстве размеров chunk, dynamic требует же ручного подбора размера. Наиболее быстро и на наибольшем количестве процессов сработала программа с параметром quided.

ПРИЛОЖЕНИЕ

Приложение 1. Листинг файла makefile

```
1.  omp_slae.out: openMP_slae_schedule.c
2.          gcc -fopenmp -o $@ openMP_slae_sch.c -lm -std=c99hpcuser221@clu:~/lab2>
```

Приложение 2. Скрипт для запуска параллельной программы

```
#!/bin/sh
#PBS -l walltime=00:02:00
#PBS -l select=1:ncpus=16:ompthreads=16

cd $PBS_O_WORKDIR

OMP_NUM_THREADS="1" ./omp_slae_schedule.out
OMP_NUM_THREADS="2" ./omp_slae_schedule.out
OMP_NUM_THREADS="4" ./omp_slae_schedule.out
OMP_NUM_THREADS="8" ./omp_slae_schedule.out
OMP_NUM_THREADS="12" ./omp_slae_schedule.out
OMP_NUM_THREADS="16" ./omp_slae_schedule.out
```

Приложение 3. Полный листинг параллельной программы на C

```
1.  ##include <math.h>
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.  #include <omp.h>
5.
6.  #define N 3500
7.  #define EPSILON 1e-6
8.  #define MAX_ITERATION_COUNT 50000
9.  #define TAU 1e-5
10.
11. double calc_square_norm(const double* vector)
12. {
13.     double norm = 0;
14.     for (int i = 0; i < N; i++)
15.     {
16.         norm += vector[i] * vector[i];
17.     }
18.     return norm;
19. }
20. void generate_vector(double* vector)
21. {
22.     for (int i = 0; i < N; i++)
23.     {
24.         vector[i] = (double)rand() / RAND_MAX * 10.0 - 5.0;
25.     }
26. }
27.
28. void generate_matrix(double* matrix)
29. {
30.     for(int i = 0; i < N; i++)
31.     {
32.         for(int j = 0; j < i; j++)
33.         {
34.             matrix[i * N + j] = matrix[j * N + i];
35.         }
36.
37.         for(int j = i; j < N; j++)
38.         {
39.             matrix[i * N + j] = (double)rand() / RAND_MAX * 2.0 - 1.0; //float in range -1 to 1
40.             if(i == j) matrix[i * N + j] = matrix[i * N + j] + N;
41.
42.         }
43.     }
44. }
45.
46. void check(double* A, double*x , double* b){
47.     double * result = malloc(sizeof(double) * N);
48.     for(int i = 0; i < N; i++) {
49.         result[i] = 0;
50.         for(int j = 0; j < N; j++) {
51.             result[i] += A[i * N + j] * x[j];
52.         }
```

```
53.     }
54.     for (int i = 0; i < N; i++){
55.         if (result[i] - b[i] < EPSILON || b[i] - result[i] < EPSILON){}
56.         else{
57.             printf("not okay\n");
58.             return;
59.         }
60.     }
61.     printf("correct\n");
62. }
63.
64. int main(int argc, char *argv[]) {
65.
66.     double* A = malloc(sizeof(double) * N * N);
67.     double* x = malloc(sizeof(double) * N);
68.     double* b = malloc(sizeof(double) * N);
69.
70.     double *buffer = malloc(sizeof(double) * N);
71.
72.     generate_matrix(A);
73.     generate_vector(x);
74.     generate_vector(b);
75.
76.     double norm_b = sqrt(calc_square_norm(b));
77.     double res = 1;
78.     int iterationCount = 1;
79.
80.     double start, end;
81.     start = omp_get_wtime();
82.     int i, j;
83.     double norm;
84.
85. #pragma omp parallel private(i, j)
86.     {
87.         while (res > EPSILON && iterationCount < MAX_ITERATION_COUNT) {
88.
89. #pragma omp  for private (j) schedule(runtime)
90.             for(i = 0; i < N; i++) {
91.                 buffer[i] = -b[i];
92.                 for(j = 0; j < N; j++) {
93.                     buffer[i] += A[i * N + j] * x[j];
94.                 }
95.             }
96.
97. #pragma omp single
98.             norm = 0;
99.
100.     #pragma omp  for  reduction (+:norm) schedule(runtime)
101.                 for (i = 0; i < N; i++){
102.                     norm += buffer[i] * buffer[i];
103.                 }
104.
105.     #pragma omp parallel for schedule(runtime)
106.                 for (i = 0; i < N; ++i) {
107.                     x[i] = x[i] - buffer[i] * TAU;
```

```
            }

    #pragma omp single
            {
                res = sqrt(norm) / norm_b;
                iterationCount++;
            }
        }
    }
    end = omp_get_wtime();
    printf("Total time is %f seconds\n", (end - start));
    //check(A, x, b);

    free(A);
    free(x);
    free(b);
    free(buffer);

    return 0;

```

Приложение 4. Скрипт для запуска программы исследования на оптимальные параметры

```
#!/bin/bash
#PBS -l walltime=00:30:00
#PBS -l select=1:ncpus=4:ompthreads=4
cd $PBS_O_WORKDIR

echo "OMP_NUM_THREADS = $OMP_NUM_THREADS"

echo "Static"
echo

for (( i = 1; i <= 10; i++ ))
do
OMP_SCHEDULE="static, $i" ./omp_slae_schedule.out
done

for (( i = 50; i <= 100; i+=50))
do
OMP_SCHEDULE="static, $i" ./omp_slae_schedule.out
done

for (( i = 500; i <= 1500; i+=500 ))
do
OMP_SCHEDULE="static,$i" ./omp_slae_schedule.out
done
echo

echo "Dynamic"
echo

for (( i = 1; i <= 10; i++ ))
do
OMP_SCHEDULE="dynamic,$i" ./omp_slae_schedule.out
done

for (( i = 50; i <= 100; i+=50 ))
do
OMP_SCHEDULE="dynamic,$i" ./omp_slae_schedule.out
done

for (( i = 500; i <= 1500; i+=500 ))
do
OMP_SCHEDULE="dynamic,$i" ./omp_slae_schedule.out
done
echo

echo "Guided"
```

```
47. echo
48.
49. for (( i = 1; i <= 10; i++ ))
50. do
51. OMP_SCHEDULE="guided,$i" ./omp_slae_schedule.out
52. done
53.
54. for (( i = 50; i <= 100; i+=50 ))
55. do
56. OMP_SCHEDULE="guided,$i" ./omp_slae_schedule.out
57. done
58.
59. for (( i = 500; i <= 1500; i+=500 ))
60. do
61. OMP_SCHEDULE="guided,$i" ./omp_slae_schedule.out
62. done
63. echo
64.
```