Задание 1. Вариант 2.
Численное решение задачи Дирихле.
Метод попеременных направлений.
Разработка параллельной программы и исследование ее эффективности.

Постановка задачи.
Дана последовательная программа, реализующая метод попеременных направлений.
Требуется разработать параллельную программу с использованием технологии OpenMP и провести исследование ее эффективности.

Цель.

Получить навыки распараллеливания существующих программ на языке Си с использованием технологии OpenMP.

Распараллеливание осуществляется с помощью анализа последовательной программы, аналогично анализу распараллеливающего компилятора. Поэтому не предполагается знания указанного алгоритма.

Требуется.

1. Разработать параллельную версию программы с использованием технологии OpenMP
2. Исследовать время выполнения разработанной программы в зависимости от размера сетки и количества используемых потоков на вычислительном комплексе IBM Regatta.
3. Построить таблицу:
   - Для вычислительного комплекса IBM Regatta:

Размер сетки	Последовательный алгоритм	Параллельный алгоритм							
		1 процессор		2 процессора		4 процессора		8 процессоров	
		Время	Ускорение	Время	Ускорение	Время	Ускорение	Время	Ускорение
256x256 x256									
384x384 x384									
512x512 x512									

Ускорение (*speedup*), получаемое при использовании параллельного алгоритма для ***p*** процессоров, определяется величиной:

$$\mathbf{Speedup(n) = T_1(n)/T_p(n),}$$

где $T_1(n)$- время последовательного выполнения задачи,
Tp(n)- время параллельного выполнения задачи при использовании ***p*** процессоров.

4. Построить графики - зависимость ускорения от количества процессоров для разных размеров сетки.
5. Подготовить отчет о выполнении задания, включающий таблицу с временами, графики, текст программы. Сделать выводы по полученным результатам (объяснить убывание или возрастание производительности параллельной программы при увеличении числа используемых процессоров, сравнить поведение параллельной программы в зависимости от размера сетки).

Методические указания.

1. Трансляция программ.

2. Запуск программ на счет.
3. Последовательная программа.
4. Литература.

1. **Трансляция программ.**

1.1. Для компиляции OpenMP-программ на вычислительном комплексе IBM Regatta необходимо использовать компилятор GCC:

gcc -fopenmp -o <имя_программы> <имя_программы>.c <опции_оптимизации>

1.2. Для компиляции последовательной программы на вычислительном комплексе IBM Regatta необходимо использовать компилятор GCC:

gcc -o <имя_программы> <имя_программы>.c <опции_оптимизации>

2. **Запуск программ на счет.**

2.1. Для запуска OpenMP-программы на счет на вычислительном комплексе IBM Regatta используйте команду:

ompsubmit -n <число_процессоров> -w <лимит_счетного времени> <имя_программы> <параметры_программы>

Предполагаемое время счета задания может быть задано в следующем формате:

- чч:мм:сс
- сс
- мм:сс.

2.2. Для запуска последовательной программы на счет на вычислительном комплексе IBM Regatta используйте команду:

ompsubmit -n 1 -w <лимит_счетного времени> <имя_программы> <параметры_программы>

3. **Последовательная программа.**

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#define  Max(a,b) ((a)>(b)?(a):(b))

#define  N   256
double   maxeps = 0.1e-7;
int itmax = 100;
int i,j,k;
double eps;

double A [N][N][N];

void relax();
void init();
void verify();
void wtime();

int main(int an, char **as)
{   int it;
   double time0, time1;
   init();
   /* time0=omp_get_wtime (); */
   wtime(&time0);
   for(it=1; it<=itmax; it++)
   {
      eps = 0.;
```

```
      relax();
      printf( "it=%4i   eps=%f\n", it,eps);
      if (eps < maxeps) break;
   }
   wtime(&time1);
   /* time1=omp_get_wtime (); */
   printf("Time in seconds=%gs\t",time1-time0);
   verify();
   return 0;
}


void init()
{
   for(i=0; i<=N-1; i++)
      for(j=0; j<=N-1; j++)
         for(k=0; k<=N-1; k++)
         {   if(i==0 || i==N-1 || j==0 || j==N-1 || k==0 || k==N-1)
               A[i][j][k]= 0.;
            else A[i][j][k]= ( 4. + i + j + k) ;
         }
}

void relax()
{
   for(i=1; i<=N-2; i++)
      for(j=1; j<=N-2; j++)
         for(k=1; k<=N-2; k++)
         {
            A[i][j][k] = (A[i-1][j][k]+A[i+1][j][k])/2.;
         }
   for(i=1; i<=N-2; i++)
      for(j=1; j<=N-2; j++)
         for(k=1; k<=N-2; k++)
         {
            A[i][j][k] =(A[i][j-1][k]+A[i][j+1][k])/2.;
         }
   for(i=1; i<=N-2; i++)
      for(j=1; j<=N-2; j++)
         for(k=1; k<=N-2; k++)
         {
            double e;
            e=A[i][j][k];
            A[i][j][k] = (A[i][j][k-1]+A[i][j][k+1])/2.;
            eps=Max(eps,fabs(e-A[i][j][k]));
         }
}

void verify()
{  double s;
   s=0.;
   for(i=0; i<=N-1; i++)
      for(j=0; j<=N-1; j++)
         for(k=0; k<=N-1; k++)
         {
            s=s+A[i][j][k]*(i+1)*(j+1)*(k+1)/(N*N*N);
         }
   printf("  S = %f\n",s);
}
```

```
void wtime(double *t)
{
   static int sec = -1;
   struct timeval tv;
   gettimeofday(&tv, (void *)0);
   if (sec < 0) sec = tv.tv_sec;
   *t = (tv.tv_sec - sec) + 1.0e-6*tv.tv_usec;
}
```

4. Литература.

- Антонов А.С. «Параллельное программирование с использованием технологии OpenMP: Учебное пособие».-М.: Изд-во МГУ, 2009. - 77 с. http://parallel.ru/info/parallel/openmp/
- OpenMP Application Program Interface. Version 3.1 July 2011 http://www.openmp.org/mp-documents/OpenMP3.1.pdf
- Инструкция по использованию вычислительного комплекса IBM Regatta http://www.regatta.cmc.msu.ru/instr.htm
- Презентация лекции «Технология параллельного программирования OpenMP» ftp://ftp.keldysh.ru/K_student/MSU2012/MSU2012_OpenMP.ppt
- Презентация лекции «Суперкомпьютерные вычислительные технологии. Параллельные алгоритмы численного решения задачи Дирихле». Лекция 4. http://angel.cs.msu.su/~popova/SuperComp2012/Lecture4.pdf