Лабораторная работа №3

Решение системы точечных вихрей методом Рунге-Кутты (OpenMP, MPI)

Решалась следующая система дифференциальных уравнений:

Здесь – заданные константы.

При этом использовался метод Рунге-Кутты 4-го порядка с распараллеливанием вычисления правой части системы. Распараллеливание осуществлялось с использованием OpenMP и MPI – на каждом узле вычисления в нескольких потоках, что обеспечивала соответствующая директива OpenMP, а потом данные агрегировались и пересылались между узлами с помощью функции MPI\_Allreduce:

void f(int n, int k, int sz, double \* arg, double \* res, double \* omega){

double \* x = arg;

double \* y = arg + n;

double \* rx = res;

double \* ry = res + n;

int cluster\_size = n / sz + ((n % sz) != 0);

int cluster\_beg = std::min(n - 1, k \* cluster\_size);

int cluster\_end = std::min(n, (k + 1) \* cluster\_size);

#pragma omp parallel for

for(int j = cluster\_beg; j < cluster\_end; ++j){

for(int i = 0; i < j; ++i){

double dx = x[i] - x[j];

double dy = y[i] - y[j];

double m = omega[j]/(dx\*dx + dy\*dy);

rx[i] += dy \* m;

ry[i] += dx \* m;

}

for(int i = j+1; i < n; ++i){

double dx = x[i] - x[j];

double dy = y[i] - y[j];

double m = omega[j]/(dx\*dx + dy\*dy);

rx[i] += dy \* m;

ry[i] += dx \* m;

}

}

#pragma omp parallel for

for(int i = 0; i < n; ++i){

rx[i] \*= c\_m1\_2pi;

ry[i] \*= c\_1\_2pi;

}

MPI\_Allreduce(MPI\_IN\_PLACE, res, 2\*n, MPI\_DOUBLE, MPI\_SUM, MPI\_COMM\_WORLD);

}

Ниже приведён код функции, осуществляющей сам метод (она не распараллеливалась):

void rk(int n, int k, int sz,

int n\_iter, double h,

double \* p, double \* w, double \*\* res\_p){

double h\_2 = h \* 0.5;

double h\_6 = h / 6.;

int n2 = n \* 2;

size\_t vsz = n2 \* sizeof(double);

\*res\_p = new double[n2 \* n\_iter];

double \* r = \*res\_p;

double \* k1 = new double[n2 \* 4];

double \* k2 = k1 + n2;

double \* k3 = k2 + n2;

double \* k4 = k3 + n2;

double \* temp = new double[n2];

memset(r, 0, vsz \* n\_iter);

memcpy(r, p, vsz);

for(int i = 1; i < n\_iter; ++i){

memset(k1, 0, vsz \* 4);

double \* r\_cur = r + i \* n2;

double \* r\_prev = r + (i - 1) \* n2;

f(n, k, sz, r\_prev, k1, w);

for(int j = 0; j < n2; ++j)

temp[j] = r\_prev[j] + h\_2 \* k1[j];

f(n, k, sz, temp, k2, w);

for(int j = 0; j < n2; ++j)

temp[j] = r\_prev[j] + h\_2 \* k2[j];

f(n, k, sz, temp, k3, w);

for(int j = 0; j < n2; ++j)

temp[j] = r\_prev[j] + h \* k3[j];

f(n, k, sz, temp, k4, w);

for(int j = 0; j < n2; ++j)

r\_cur[j] = r\_prev[j] + h\_6 \* (k1[j] + k4[j] + 2 \* (k2[j] + k3[j]));

}

delete[] k1;

delete[] temp;

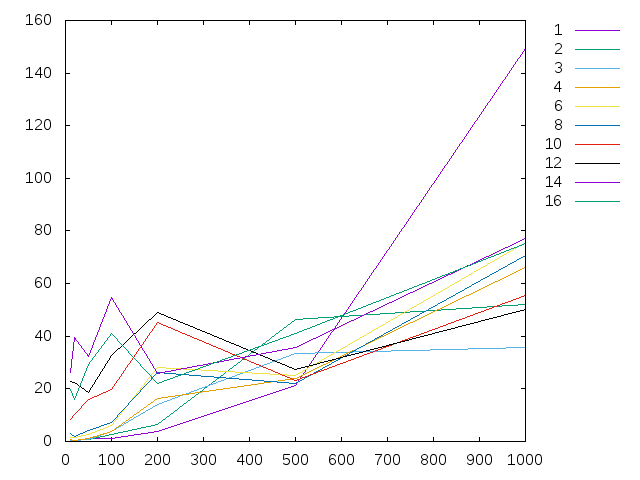
}

Ниже приведены таблица и график, иллюстрирующие время выполнения программы. Здесь

|  |  |  |
| --- | --- | --- |
| # |  | Время выполнения |
| 1 | 1 | 141 |
| 2 | 4 | 66 |
| 3 | 8 | 50 |
| 4 | 16 | 40 |

Далее приведены графики тестов MPI и OMP.

MPI, n\_iters = 4000:



OMP, n\_iters = 200:

