Work 11 Кириленко Константин 351



void integral2(const double a, const double b, const double a2, const double b2,

const double h, const double h2, double\* res)

{

int i, j, n, n2;

double sum1, sum2; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b - a) / h); // количество точек сетки интегрирования

n2 = (int)((b2 - a2) / h2); // количество точек сетки интегрирования

sum1 = 0.0;

sum2 = 0.0;

int NProc, ProcId;

MPI\_Comm\_size(MPI\_COMM\_WORLD, &NProc);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

long long na = n / NProc;

long long nb = (ProcId + 1) \* na;

if (NProc == ProcId + 1) {

nb = n;

}

long long st = ProcId \* na;

for (i = st; i < nb; i++)

{

x = a + i \* h + h / 2.0;

sum2 = 0.0;

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

{

y = a2 + j \* h2 + h2 / 2.0;

sum2 += ((pow(e, sin(PI \* x) \* cos(PI \* y)) + 1) / ((b - a) \* (b2 - a2))) \* h \* h2;

}

sum1 += sum2;

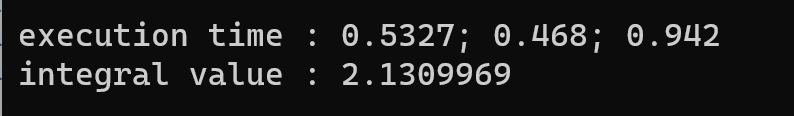
}

MPI\_Reduce(&sum1, res, 1, MPI\_LONG\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

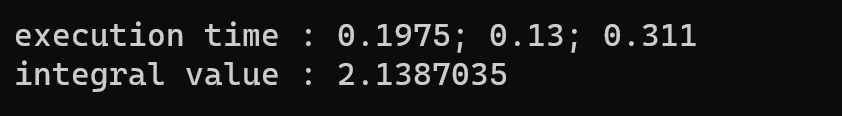
}

при h =0.01:

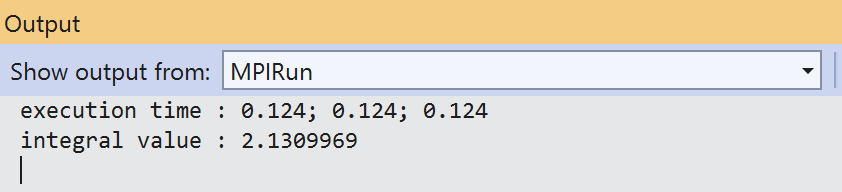
Последовательный:



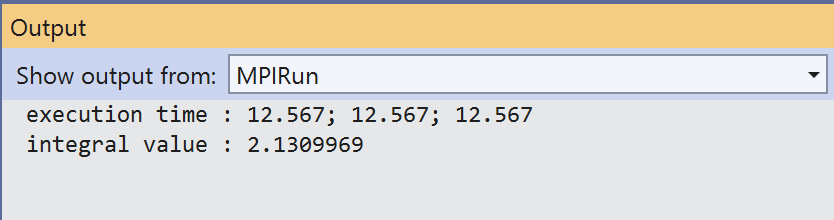
OMP:



MPI:



h=0.001:



Версия программы, распараллеленная с помощью MPI, работает значительно быстрее последовательной и параллельной OMP