ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ВС

Лабораторная работа №3 по дисциплине «Параллельные вычислительные технологии» по теме: «Параллельное численное интегрирование»

Выполнил:
ст. гр. ИВ-823
Шиндель Э. Д.

Проверил:
Заведующий
кафедрой ВС
Курносов М. Г.

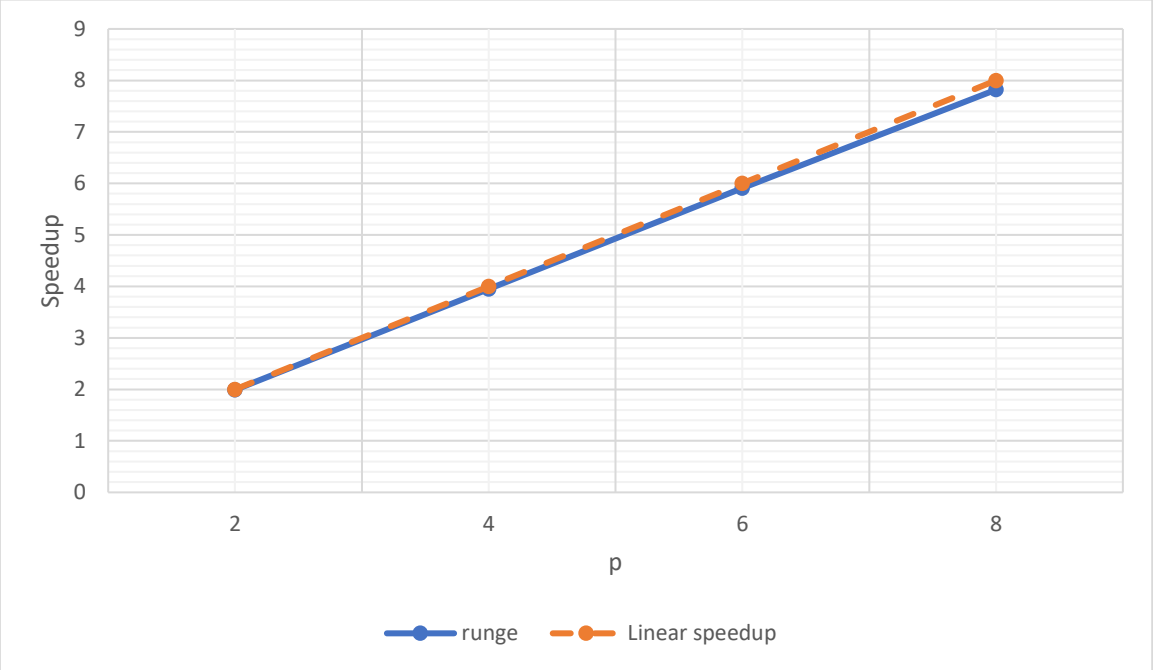
Новосибирск, 2020

Содержание:

Задание 1
B-2

$$f(x) = \frac{\ln(1+x)}{x}, \qquad a = 0.1, \quad b = 1$$

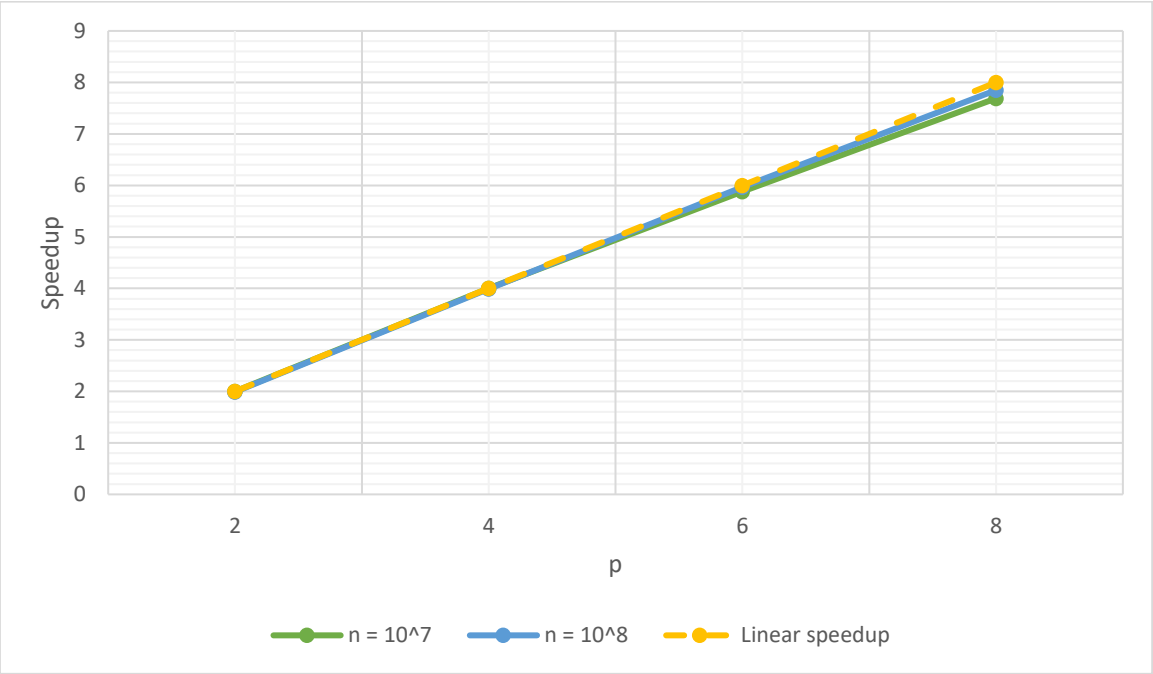
Количество потоков								
1	2		4		6		8	
Time	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
20.03	10.06	**1.99**	5.07	**3.95**	3.39	**5.91**	2.56	**7.82**



Задание 2
B-2

$f(x,y) = e^{(x+y)^2}$, $\Omega = \{x \in (0,1),\ y \in (0,1-x)\}$

n	Количество потоков								
	1	2		4		6		8	
	Time	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
10^7	1	0.5	2	0.25	4	0.17	5.88	0.13	7.69
10^8	9.97	5	1.99	2.5	3.99	1.67	5.97	1.27	7.85



Листинг

```
//Задание 1
#include <stdio.h>
#include <math.h>
#include <omp.h>

double func(double x) {
    return (log(1.0 + x) / x);
}

int main()
{
    const double eps = 1E-5;
    const double a = 0.1;
    const double b = 1.0;
    const int n0 = 100000000;
    double sq[2];

    printf("Numerical integration: [%f, %f], n0 = %d, EPS = %f\n", a, b, n0, eps);

    double t = omp_get_wtime();
    #pragma omp parallel
    {
        int n = n0, k;
        double delta = 1;
        for (k = 0; delta > eps; n *= 2, k ^= 1) {
            double h = (b - a) / n;
            double s = 0.0;
```

```
            sq[k] = 0;
            #pragma omp barrier // Ждем пока все потоки закончат обнуление sq[k]

            #pragma omp for nowait
            for (int i = 0; i < n; i++)
            s += func(a + h * (i + 0.5));

            #pragma omp atomic
            sq[k] += s * h; // Ждем пока все потоки обновят sq[k]
            #pragma omp barrier

            if (n > n0) delta = fabs(sq[k] - sq[k ^ 1]) / 3.0;
        }
            #pragma omp master
            printf("Result Pi: %.12f; Runge rule: EPS %e, n %d\n",
                                                    sq[k] * sq[k], eps, n / 2);
    }
    t = omp_get_wtime() - t;
    printf("Elapsed time (sec.): %.6f\n", t);

    return 0;
}
```

```
//Задание 2
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <omp.h>

double getrand(unsigned int *seed) {
    return (double)rand_r(seed) / RAND_MAX;
}

double func(double x, double y) {
    return exp(pow(x + y, 2));
}

int main()
{
    const int n = 100000000;
    printf("Numerical integration by Monte Carlo method: n = %d\n", n);
    int in = 0;
    double s = 0;

    double t = omp_get_wtime();
    #pragma omp parallel
    {
        double s_loc = 0;
        int in_loc = 0;
        unsigned int seed = omp_get_thread_num();
```

```
        #pragma omp for nowait
        for (int i = 0; i < n; i++) {
            double x = getrand(&seed);           /* x in (0, 1) */
            double y = getrand(&seed) * (1 - x); /* y in (0, 1 - x) */
            if ((y < 1 - x) && (y > 0) && (x > 0) && (x < 1)) {
                in_loc++;
                s_loc += func(x, y);
            }
        }

        #pragma omp atomic
        s += s_loc;

        #pragma omp atomic
        in += in_loc;

    }
    double v = in / n;
    double res = v * s / in;
    t = omp_get_wtime() - t;

    printf("Result: %.12f, n %d\n", res, n);
    printf("Elapsed time (sec.): %.6f\n", t);

    return 0;
}
```