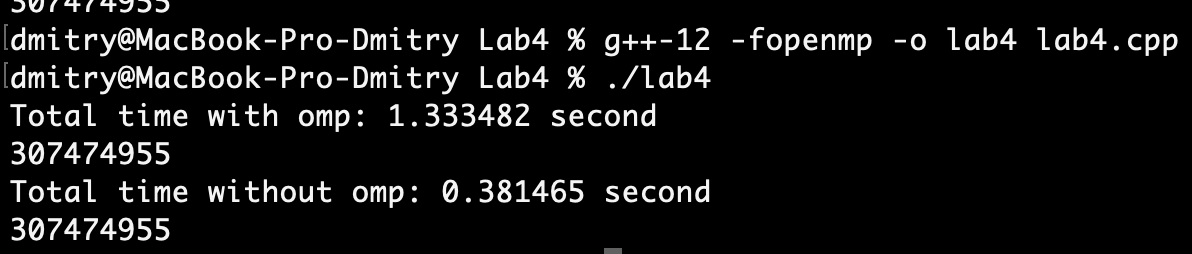
**Технологии параллельного программирования**

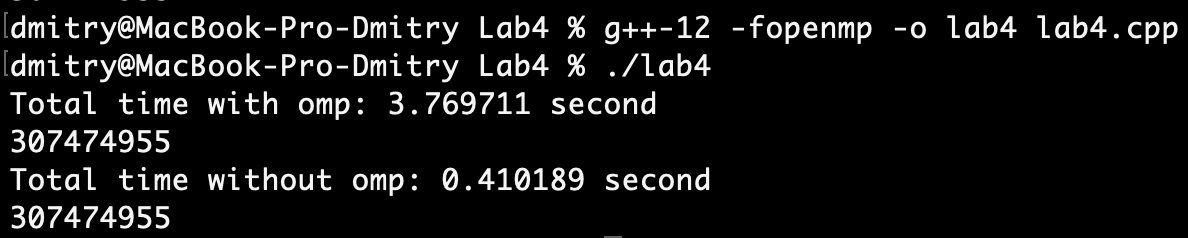
**Отчет по 4 лабораторной работе**

**Ивкучев Дмитрий ПИН-32**

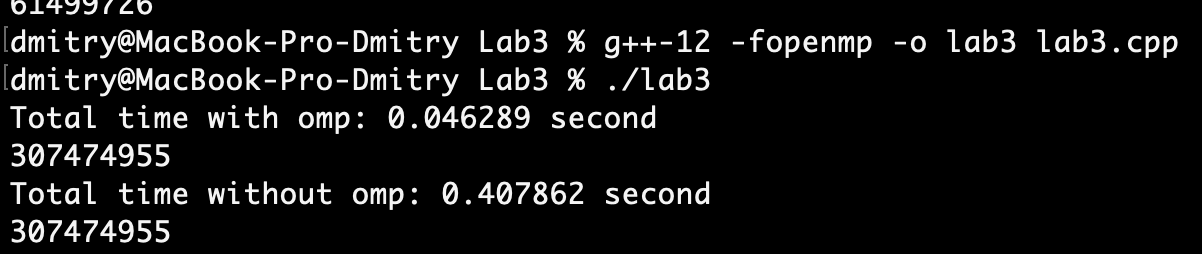
Используем атомарную операцию вместо редукции для 50 млн элементов:



То же самое с использованием critical:



Вычисления из предыдущей работы:



Код:

#include <iostream>

#include <omp.h>

using namespace std;

const int N = 50000000, LIMIT = 37000;

int main() {

int \*A = new int[N];

int \*B = new int[N];

int \*C = new int[N];

int i;

long long sum = 0;

for (i = 0; i < N; i++) {

A[i] = rand() % 10;

//cout << A[i] << "\t";

}

//cout << endl;

for (i = 0; i < N; i++) {

B[i] = rand() % 10;

//cout << B[i] << "\t";

}

//cout << endl;

double start, end;

start = omp\_get\_wtime();

#pragma omp parallel shared(A, B, C) if(N > LIMIT)

{

//pragma omp critical

//

#pragma omp for private(i)

for (i = 0; i < N; i++) {

if (A[i] > B[i]) C[i] = A[i];

else C[i] = B[i];

#pragma omp critical

sum += C[i];

//printf("Прибавлена %d\n",C[i]);

}

}

end = omp\_get\_wtime();

printf("Total time with omp: %f second\n", end-start);

printf("%lld\n",sum);

delete [] C;

C = new int[N];

sum = 0;

start = omp\_get\_wtime();

for (i = 0; i < N; i++) {

if (A[i] > B[i]) C[i] = A[i];

else C[i] = B[i];

sum += C[i];

//printf("Прибавлена %d\n",C[i]);

}

end = omp\_get\_wtime();

printf("Total time without omp: %f second\n", end-start);

printf("%lld\n",sum);

return 0;

}