Мордовский государственный университет имени Н.П.Огарёва

Специальность: автоматизированные системы обработки информации и управления

# ПРАКТИЧЕСКАЯ РАБОТА

### по системам с параллельной обработкой данных (С++)

**OpenMP**

ПР – 02069964 – 090301 – 17

|  |  |
| --- | --- |
| ВЫПОЛНИЛ:  Студент 241 группы специальности АСОИУ  С.А.Константинов  \_\_\_\_\_\_\_\_\_\_\_\_ 13.07.2017 | ПРОВЕРИЛА:  ст. преподаватель кафедры АСОИУ  к.т.н. Н. П. Плотникова  \_\_\_\_\_\_\_\_\_\_\_\_ 13.07.2017 |

Саранск 2017

**Задания по разделу «OpenMP»**

1. Реализовать приложение, выполняющее умножение элементов массива по модулю 109+7. Для распараллеливания вычислений использовать OpenMP. Выполнить замер производительности (времени работы) программы для различного количества потоков, построить график зависимости. Объяснить полученные результаты.

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  #include <omp.h>  #include <ctime>  using namespace std;  const int array\_size = 1e8;  const int modul = 1e9 + 7;  long long array\_mul = 1;  vector<int> my\_array(array\_size, 1);  void OmpFun()  {  array\_mul = my\_array[0];  #pragma omp parallel for shared(my\_array)  for (int i = 1; i < array\_size; ++i)  {  if ((array\_mul \* my\_array[i]) % modul != 0) {  array\_mul = (my\_array[i] \* array\_mul) % modul;  }  array\_mul \*= my\_array[i];  }  }  int main()  {  omp\_set\_num\_threads(1);  srand(time(nullptr));  generate(my\_array.begin(), my\_array.end(), rand);  auto start = clock();  OmpFun();  auto end = clock();  cout << "Time " << (end - start) / 1000.0 << endl;  return 0;  } |

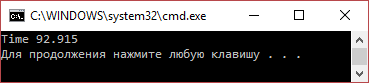


Рисунок 1 – Результат работы с 1 потоком

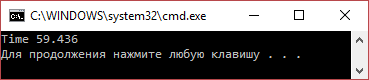


Рисунок 2 – Результат работы с 2 потоками

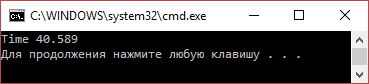


Рисунок 3 – Результат работы с 3 потоками

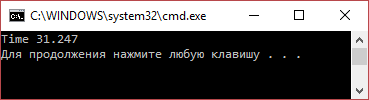


Рисунок 4 – Результат работы с 4 потоками

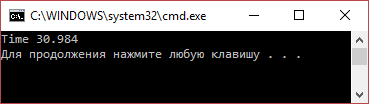


Рисунок 5 – Результат работы с 5 потоками

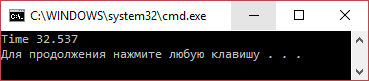


Рисунок 6 – Результат работы с 6 потоками

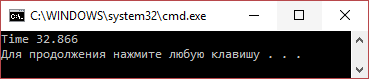


Рисунок 7 – Результат работы с 7 потоками

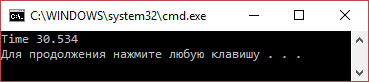


Рисунок 8 – Результат работы с 8 потоками

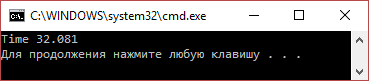


Рисунок 9 – Результат работы с 9 потоками

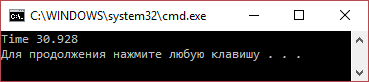


Рисунок 10 – Результат работы с 10 потоками

При увеличении вычислительных единиц время работы уменьшается.

1. Реализовать приложение, выполняющее перемножение матриц произвольного размера. Для распараллеливания вычислений использовать OpenMP. Выполнить замер производительности (времени работы) программы для различного количества потоков, построить график зависимости. Объяснить полученные результаты.

|  |
| --- |
| #include <iostream>  #include <omp.h>  #include <ctime>  #include <vector>  using namespace std;  const int array\_size = 100;  vector<vector<int>> a, b, c;  void OmpFun(vector<vector<int>> m, vector<vector<int>> n, vector<vector<int>> p) {  #pragma omp parallel for shared(m, n, p)  for (int i = 0; i < array\_size; ++i) {  for (int j = 0; j < array\_size; ++j) {  for (int k = 0; k < array\_size; ++k)  p[i][j] += m[i][k] \* n[k][j];  }  }  }  int main() {  srand(time(nullptr));  a.resize(array\_size);  b.resize(array\_size);  c.resize(array\_size);  for (int i = 0; i < array\_size; ++i) {  a[i].resize(array\_size);  b[i].resize(array\_size);  c[i].resize(array\_size);  for (int j = 0; j < array\_size; ++j) {  a[i][j] = rand();  b[i][j] = rand();  c[i][j] = 0;  }  }  omp\_set\_num\_threads(5);  auto start = clock();  OmpFun(a, b, c);  auto end = clock();  cout << "Time: " << (end - start) / 1000.0 << endl;  return 0;  } |

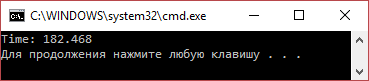


Рисунок 1 – Результат работы с 1 потоком.

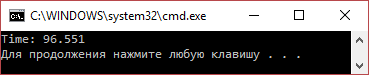


Рисунок 2 – Результат работы с 2 потоками.

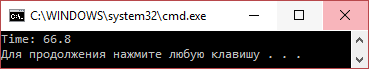


Рисунок 3 – Результат работы с 3 потоками.

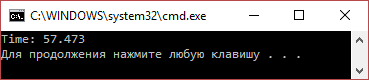


Рисунок 4 – Результат работы с 4 потоками.

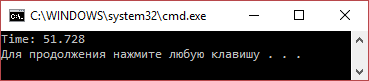


Рисунок 5 – Результат работы с 5 потоками

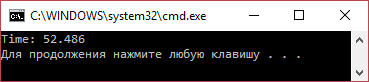


Рисунок 6 – Результат работы с 6 потоками

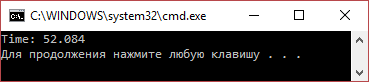


Рисунок 7 – Результат работы с 7 потоками

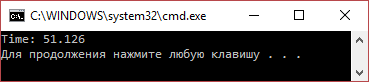


Рисунок 8 – Результат работы с 8 потоками

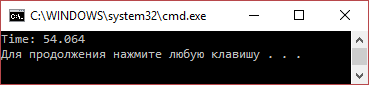


Рисунок 9 – Результат работы с 9 потоками

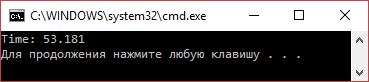


Рисунок 10 – Результат работы с 10 потоками

При увеличении вычислительных единиц время работы уменьшается.

Процессор: Intel Pentium quad core N3540, 2.66GHz