МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №2 по дисциплине «Параллельное программирование»

**Параллельные алгоритмы в языке C++**

|  |  |
| --- | --- |
| Выполнил: студент группы ФИб-4302-51-00 | / Д.А. Савин / |
| Проверил: ст. преподаватель каф. ПМиИ | / В.А. Бызов / |

Киров 2021

Задание 1

Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, вычисляющую для каждого элемента целочисленного вектора количество его делителей. Элементы вектора – случайные натуральные числа из диапазона [105, 106]. Результат должен быть записан в новый вектор. Провести тестирование программы на векторах размера 5 ∙ 105, 106, 2 ∙ 106 с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты. Сделать выводы.

Код:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <ctime>  #include <execution>  #include <algorithm>  using namespace std;  const int dimm = 10000\*5;  vector <int> v1 = {};  vector <int> res = {};  chrono::time\_point<chrono::high\_resolution\_clock>start, ennd;  int getCountDev(int x)  {  int count = 1;  for (int i = 1; i <= x / 2; i++)  {  if (x % i == 0)  {  count++;  }  }  return count;  }  void display(vector <int> v)  {  for (int i = 0; i < v.size(); i++)  {  cout << v.at(i) << " ";  }  cout << endl;  }  void init()  {  srand(time(0));  for (int i = 0; i < dimm; i++)  {  v1.push\_back(rand() \* 5);  }  res.resize(v1.size());  }  int main()  {  init();  //display(v1);  start = chrono::high\_resolution\_clock::now();  transform(std::execution::par, v1.begin(), v1.end(), res.begin(), getCountDev);  ennd = chrono::high\_resolution\_clock::now();  chrono::duration<double> diff = ennd - start;  cout << "time: " << diff.count() << endl;  //display(res);  } |

Таблица 1 – Время вычисления количества делителей

элементов целочисленного вектора, с

|  |  |  |  |
| --- | --- | --- | --- |
| Размер матрицы | Политика выполнения | | |
| seq | par | par\_unseq |
| 5\*105 | 7,72047 | 3,23782 | 3,30778 |
| 106 | 14,0302 | 7,01601 | 7,21814 |
| 2\*106 | 26,889 | 13,4826 | 13,8652 |

Вывод: Самое быстрое выполнение программы оказалось с политикой выполнения par.

Задание 2

Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, суммирующую элементы вектора. Указание. Использовать алгоритм std::reduce. Провести тестирование программы на векторах размера 5 ∙ 108, 109, 2 ∙ 109 с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты. Сделать выводы.

Код:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <ctime>  #include <execution>  #include <algorithm>  #include <chrono>  using namespace std;  const int dimm = 10\*100000000;  vector <long long> v1 = {};  //vector <int> res = {};  chrono::time\_point<chrono::high\_resolution\_clock>start, ennd;  void display(vector <long long> v)  {  for (long long i = 0; i < v.size(); i++)  {  cout << v.at(i) << " ";  }  cout << endl;  }  void init()  {  srand(time(0));  for (long long i = 0; i < dimm; i++)  {  v1.push\_back(rand() % 25);  }  //res.resize(v1.size());  }  int main()  {  init();  long long result = 0;  //display(v1);  start = chrono::high\_resolution\_clock::now();  reduce(std::execution::par, v1.cbegin(), v1.cend());  ennd = chrono::high\_resolution\_clock::now();  chrono::duration<double> diff = ennd - start;  //cout << "result: " << result << endl;  cout << "time: " << diff.count() << endl;  //display(res);  } |

Таблица 2 – Время вычисления суммирования

элементов целочисленного вектора, с

|  |  |  |  |
| --- | --- | --- | --- |
| Размер матрицы | Политика выполнения | | |
| seq | par | par\_unseq |
| 5\*108 | 3,44933 | 0,607282 | 1.05086 |
| 109 | 40,042 | 15,7305 | 18,0193 |
| 2\*109 | Ошибка(не хватает памяти) | Ошибка(не хватает памяти) | Ошибка(не хватает памяти) |

Вывод: Самое быстрое выполнение программы оказалось с политикой выполнения par.

Задание 3

Используя параллельные алгоритмы стандартной библиотеки языка C++, реализовать классический алгоритм умножения матриц. Указание. Алгоритмы использовать для вычисления скалярного произведения. Провести тестирование программы на матрицах размера 512, 1024, 2048 с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты с результатами лабораторной работы 1. Сделать выводы.

Код:

|  |
| --- |
| #include <iostream>  #include <ctime>  #include <chrono>  #include <execution>  using namespace std;  const int DIMM = 2048;  chrono::time\_point<chrono::high\_resolution\_clock>start, ennd;  void display(vector<vector<int>> v)  {  for (long long i = 0; i < v.size(); i++)  {  for (int j = 0; j < DIMM; j++)  {  cout << v[i][j] << " ";  }  cout << endl;  }  cout << " \_\_\_ " << endl;  }  int main()  {    vector<vector<int>> v1 (DIMM);  vector<vector<int>> v2 (DIMM);  vector<vector<int>> out (DIMM);  srand(time(0));  for (long long i = 0; i < DIMM; i++)  {  v1[i] = vector<int>(DIMM);  v2[i] = vector<int>(DIMM);  for (int j = 0; j < DIMM; j++)  {  v1[i][j] = rand() % 25;  v2[i][j] = rand() % 25;  }  }  //display(v1);    //display(v2);    start = chrono::high\_resolution\_clock::now();  std::transform(std::execution::seq, v1.begin(), v1.end(), out.begin(),  [v2](vector<int> row) -> vector<int> {  vector<int> ret;  for (int i = 0; i < v2[0].size(); i++) {  int rowcolsum = 0;  for (int k = 0; k < row.size(); k++) {  rowcolsum += row[k] \* v2[k][i];  }  ret.push\_back(rowcolsum);  }  return ret;  });  ennd = chrono::high\_resolution\_clock::now();  //display(out);  chrono::duration<double> diff = ennd - start;  cout << "time: " << diff.count() << endl;  } |

Таблица 3 – Время вычисления скалярного произведения двумерных матриц, с

|  |  |  |  |
| --- | --- | --- | --- |
| Размер матрицы | Политика выполнения | | |
| seq | par | par\_unseq |
| 512 | 4,10183 | 2,0201 | 2,06471 |
| 1024 | 32,0655 | 15,3371 | 15,9672 |
| 2048 | 402,157 | 126,027 | 127,076 |

Вывод: Наилучшую асимптотику имеет политика par.

Задание 4

Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, сортирующую массив действительных чисел. Протестировать на массивах разной размерности (3-5 вариантов) со всеми политиками выполнения. Замерить время выполнения. Размеры массивов подобрать таким образом, чтобы нагрузить систему. Сравнить результаты. Результаты должны быть представлены в удобном, понятном и наглядном виде.

Код:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <execution>  #define MAX\_VALUE 10  using namespace std;  int main()  {  vector<int> sizes{ (int)pow(10,6), (int)pow(10,7), (int)pow(10,8), 2 \* (int)pow(10,8) };  for (int i = 0; i < sizes.size(); i++)  {  vector<int> vector;  for (int j = 0; j < sizes[i]; j++)  {  vector.push\_back(0 + ((double)rand() / RAND\_MAX) \* (MAX\_VALUE - 0));  }  std::chrono::time\_point<std::chrono::high\_resolution\_clock> start, end;  start = std::chrono::high\_resolution\_clock::now();  sort(std::execution::par\_unseq, vector.begin(), vector.end());  end = std::chrono::high\_resolution\_clock::now();  std::chrono::duration<double> diff = end - start;  cout << diff.count() << " " << endl;  }  } |

Таблица 4 – Время сортировки массива действительных чисел, с

|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | Политика выполнения | | |
| seq | par | par\_unseq |
| 106 | 0,130378 | 0,16859 | 0,131528 |
| 107 | 1,64085 | 1,58277 | 1,62918 |
| 108 | 16,2974 | 14,2002 | 14,5327 |
| 2\*108 | 34,134 | 31,9171 | 32,4683 |

Вывод: Наилучшую асимптотику имеет политика par.