ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ВС

Лабораторная работа №2 по дисциплине

«Параллельные вычислительные технологии» по теме:

«Умножение матрицы на вектор»

Выполнил: ст. гр. ИВ-823

Шиндель Э. Д.

Проверил: Заведующий кафедрой ВС Курносов М. Г.

Новосибирск, 2020

**Содержание:**

Задание3

Листинг4

**Задание**

1. Определить предельные размеры матрицы и вектора (параметры m, n), которые можно перемножать на узле кластера Jet.

На узле кластера Jet 8 GB памяти. Матрица квадратная (n = m). В нашей программе хранится матрица (n · n) ячеек + 2 вектора (2 · n) ячеек, каждая ячейка типа double занимает 8 байт. Тогда под массивы надо 8 · (n · n + 2 · n) байт памяти. На практике занимать всю память нельзя, возьмём 80% от 8 GB – это 0,8 · 8 GB. Составим уравнение и найдём решение: 8 · (n2 + 2 · n) = 0,8 · 8 · 230.

Получилось, что n ≈ 29307.

1. Построить график зависимости коэффициента ускорения параллельной программы от числа потоков.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M = N | Количество потоков | | | | | | | | |
| 1 | 2 | | 4 | | 6 | | 8 | |
| Time | Time | Speedup | Tim e | Speedup | Time | Speedup | Time | Speedu p |
| 15000 (1716MiB) | 1,78 | 0,89 | **2** | 0,45 | **3,96** | 0,32 | **5,56** | 0,3 | **5,93** |
| 20000 (3052MiB) | 3,15 | 1,58 | **1,99** | 0,8 | **3,94** | 0,57 | **5,53** | 0,54 | **5,83** |
| 25000 (4768MiB) | 4,93 | 2,47 | **2** | 1,24 | **3,98** | 0,89 | **5,54** | 0,84 | **5,87** |

**Листинг**

#include <stdio.h> #include <stdlib.h> #include <sys/time.h> #include <omp.h>

int n; int m;

double wtime() { struct timeval t;

gettimeofday(&t, NULL);

return (double)t.tv\_sec + (double)t.tv\_usec \* 1E-6;

}

void matrix\_vector\_product\_omp(double \*a, double \*b, double \*c, int m, int n) { #pragma omp parallel

{

int nthreads = omp\_get\_num\_threads(); int threadid = omp\_get\_thread\_num(); int items\_per\_thread = m / nthreads; int lb = threadid \* items\_per\_thread;

int ub = (threadid == nthreads - 1) ? (m - 1) : (lb + items\_per\_thread - 1);

for (int i = lb; i <= ub; i++) { c[i] = 0.0;

for (int j = 0; j < n; j++) c[i] += a[i \* n + j] \* b[j];

}

}

}

void run\_parallel() { double \*a, \*b, \*c;

a = malloc(sizeof(\*a) \* m \* n); b = malloc(sizeof(\*b) \* n);

c = malloc(sizeof(\*c) \* m);

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) a[i \* n + j] = i + j;

}

for (int j = 0; j < n; j++) b[j] = j;

double t = wtime(); matrix\_vector\_product\_omp(a, b, c, m, n); t = wtime() - t;

printf("Elapsed time: %.6f sec.\n", t); free(a);

free(b);

free(c);

}

int main(int argc, char \*\*argv)

{

n = m = 25000;

printf("Matrix-vector product (c[m] = a[m, n] \* b[n]; m = %d, n = %d)\n", m, n); printf("Memory used: %d MiB\n", ((m \* n + m + n) \* sizeof(double)) >> 20);

run\_parallel();

return 0;

}