**Национальный Исследовательский Университет Информационных технологий, Механики и Оптики**

**Лабораторная работа № 3 по курсу «Параллельные вычисления»**

Выполнил:

Бедаш Д.С., гр. 5110

Проверил:

Соснин Владимир Валерьевич

Задание:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ФИО студента** | **Математическое ожидание** | **Дисперсия** | **Преобразование исходной матрицы** | **Способ формирования вектора с результатами** | **Метод сортировки вектора с результатами** |
| Бедаш Дмитрий Сергеевич | -45 | 65 | Умножение М1 на скаляр 5. | Выборочная дисперсия каждой строки | Shell sort |

Дочитать всю книгу Антонова А.С. В«Параллельное программирование с использованием технологии OpenMPВ». Найти ответы на вопросы из разделов В«ЗаданиеВ» на страницах 12, 28, 35, 54, 66.

2.Переписать программу, полученную в Лабораторной работе №2, так, чтобы параллельно выполнялись все подзадачи, включая операцию сортировки.

3.Построить график параллельного ускорения Slab3(N) полученной версии программы в одной системе координат с графиком ускорением Slab2(N) лучшего варианта программы, полученной в Лабораторной работе №2.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 250 | 275 | 300 | 325 | 350 | 375 | 400 | 425 | 450 | 475 | 500 |
| Delta\_ ms\_openmp\_all | 7890 | 9786 | 15176 | 18601 | 23587 | 39913 | 66572 | 96872 | 119632 | 158933 | 181257 |
| SP | 1,07 | 1,199 | 1,008 | 1,151 | 1,0231 | 0,9118 | 1,041 | 0,724 | 0,59214 | 0,6224 | 0,6879 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 250 | 275 | 300 | 325 | 350 | 375 | 400 | 425 | 450 | 475 | 500 |
| Delta\_ ms\_openmp\_best | 7303 | 10145 | 13473 | 15616 | 19855 | 32927 | 68732 | 98931 | 116154 | 155752 | 185752 |
| SP | 1,16 | 1,157 | 1,135 | 1,371 | 1,2154 | 1,1053 | 1,008 | 0,709 | 0,60987 | 0,6351 | 0,6713 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 250 | 275 | 300 | 325 | 350 | 375 | 400 | 425 | 450 | 475 | 500 |
| Delta\_ ms\_seq | 8435 | 11733 | 15292 | 21403 | 24131 | 36394 | 69303 | 70103 | 70839 | 98913 | 124695 |

4.Для последовательной и параллельной версии полученной программы построить теоретические модели вычислительной сложности каждого из пяти этапов выполнения программы. Пусть получились соответственно следующие функции сложности О(f(n)) для параллельной версии программы: p1(N),..., p5(N), и следующие функции для последовательной версии: q1(N), …, q5(N). Если возникают сложности при создании модели, можно использовать книгу Кормена В«Алгоритмы: построение и анализВ».

Последовательная версия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название шага | лучлее | среднее | худшее | Память | Стабильность |
| Генерация случайного числа - гамма распределение | n2ln(n) | n2ln(n) | n2ln(n) | 1 | Да |
| Умножение на вектор 5 | n2 | n2 | n2 | 1 | Да |
| Произведение матриц | n3 | n3 | n3 | 1 | Да |
| Вычисление вектора | n(n+n) | n(n+n) | n(n+n) | 1 | Да |
| Сортировка шелла | N | n (log n)2 | n (log n)2 | 1 | Нет |

Паралельная

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название шага | лучлее | среднее | худшее | Память | Стабильность |
| Генерация случайного числа - гамма распределение | n2ln(n) | n2ln(n) | n2ln(n) | 1 | Да |
| Умножение на вектор 5 | n2 | n2 | n2 | 1 | Да |
| Произведение матриц | n3 | n3 | n3 | 1 | Да |
| Вычисление вектора | n(n+n) | n(n+n) | n(n+n) | 1 | Да |
| Сортировка шелла | n | n (log n)2 | n (log n)2 | 1 | Нет |

5.Для последовательной и параллельной версии программы построить теоретические модели вычислительной сложности всей программы (соответственно P(N) и Q(N)). Например, для параллельной версии может получиться так: P(N) = C1\*p1(N) + … + C5\* p5(N), где константы Ci зависят от сложности итераций каждой из 5 операций.

P(N) =

Q(N) =

6.На основании экспериментов с последовательной и параллельной версией программы построить графики зависимости времени их выполнения от N, т.е. Tseq(N), Tpar(N). На системе координат Tseq(N), построить график Q(N), подобрав коэффициенты Ci так, чтобы графики как можно ближе совпали. Аналогично сделать для Tpar(N) и P(N). N следует варьировать в тех же пределах и с тем же шагом, что и в Лабораторной работе

**Характеристика системы**

**Процессор:**

model name : Intel(R) Core(TM) i5 CPU M 480 @ 2.67GHz

cpu MHz : 1197.000

cache size : 3072 KB

cpu cores : 2

clflush size : 64

cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual

**ОЗУ – два модуля с одинаковыми параметрами:**

Memory Device

Array Handle: 0x001D

Error Information Handle: Not Provided

Total Width: 64 bits

Data Width: 64 bits

Size: 2048 MB

Form Factor: DIMM

Set: None

Locator: DIMM\_A

Bank Locator: DIMM\_A

Type: DDR3

Type Detail: Synchronous

Speed: 1333 MHz

Manufacturer: CE80

Serial Number: 9249EC3A

Asset Tag: 1051

Part Number: M471B5773DH0-CH9

Rank: 1

**OS:**

Description: Ubuntu 13.04

Codename: raring

Linux core version 3.8.0-35-generic

**GCC:**

COLLECT\_GCC=gcc

COLLECT\_LTO\_WRAPPER=/usr/lib/gcc/x86\_64-linux-gnu/4.7/lto-wrapper

Целевая архитектура: x86\_64-linux-gnu

Параметры конфигурации: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.7.3-1ubuntu1' --with-bugurl=file:///usr/share/doc/gcc-4.7/README.Bugs --enable-languages=c,c++,go,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.7 --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.7 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --enable-plugin --with-system-zlib --enable-objc-gc --with-cloog --enable-cloog-backend=ppl --disable-cloog-version-check --disable-ppl-version-check --enable-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --with-tune=generic --enable-checking=release --build=x86\_64-linux-gnu --host=x86\_64-linux-gnu --target=x86\_64-linux-gnu

Модель многопоточности: posix

gcc версия 4.7.3 (Ubuntu/Linaro 4.7.3-1ubuntu1)

Вывод : В результате проведенной лабораторной работы был получен опыт ручной оптимизации последовательных программ.

На малых матрицах получилось получить прирост производительности до 40% . наблюдается падение производительности при росте размеров матриц.

Код Программы:

#include <stdio.h>

#include <stdlib.h>

#include <gsl/gsl\_rng.h>

#include <gsl/gsl\_randist.h>

#include <math.h>

#include <sys/time.h>

#define C\_PARAM = 5

void insertionsort(double a[], int n, int stride) {

int j;

for (j=stride; j<n; j+=stride) {

double key = a[j];

int i = j - stride;

while (i >= 0 && a[i] > key) {

a[i+stride] = a[i];

i-=stride;

}

a[i+stride] = key;

}

}

void shellsort(double a[], int n)

{

int i, m;

for(m = n/2; m > 0; m /= 2)

{

#pragma omp parallel for shared(a,m,n) private (i) default(none)

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

insertionsort(&(a[i]), n-i, m);

}

}

void printmatrix (double a[],int n)

{

int i;

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

printf("|%0.1f|", a[i]);

}

}

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

{

int i,j,k, N,r;

N = atoi(argv[1]);

struct timeval T1, T2;

long delta\_ms;

double vector = 5;

double random\_number;

double M1[N][N];

double M2[N][N];

double M3[N][N];

double V[N];

double sred;

double num = 0;

double Nt=N;

double a,b;

gettimeofday(&T1, NULL);

printf("Started\n");

gsl\_rng \* rgen = gsl\_rng\_alloc(gsl\_rng\_taus);

a=31,15;

b=-1,444445;

#pragma omp parallel shared(r)

{

#pragma omp for schedule(guided,1) private (i,j) nowait

for (r=0; r<100; r++)

{

//printf("\n\nMatrix1\n");

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

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

random\_number= gsl\_ran\_gamma(rgen,a,b);

M1[i][j] = random\_number;

//printf("|%f|", M1[i][j]);

}

//printf("\n");

}

//printf("Matrix2\n");

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

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

M2[i][j] = M1[i][j]\*vector;

//printf("|%f|", M2[i][j]);

}

//printf("\n");

}

//printf("Matrix3 resoult\n");

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

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

M3[i][j] = 0;

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

M3[i][j] = M3[i][j] + M1[i][k]\*M2[k][j];

}

//printf("|%f|", M3[i][j]);

}

//printf("\n");

}

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

V[i] =0;

sred=0;

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

sred += M3[i][j];

}

sred = sred/Nt;

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

num = M3[i][j];

num=num-sred;

V[i] = num\*num;

}

V[i]= V[i]/Nt;

}

//printf("\nBefore sort:\n");

//printmatrix(V,N);

//printf("\nShell sort:\n");

shellsort(V,N);

//printmatrix(V,N);

//printf("\n\n");

}

}

printf("\nStopt\n");

gettimeofday(&T2, NULL);

delta\_ms = 1000\*(T2.tv\_sec - T1.tv\_sec) + (T2.tv\_usec - T1.tv\_usec)/1000;

printf("\nN=%d. Milliseconds passed: %ld\n", N, delta\_ms);

return 0;

}