|  |  |  |
| --- | --- | --- |
| Кафедра ЭВМ и С  Лабораторная работа №6  «Изучение параллельных вычислений на вычислительных системах современной архитектуры»  Вариант 5 | Студент | Сидский Н. А. |
| Группа | ИВТ-261 |
| Дата |  |
| Преподаватель |  |
| Дата отчета |  |
| Балл |  |
| Подпись преподавателя |  |

Общее задание

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размерность вектора (Size) | Время исполнения OpenMP-версии на CPU, сек | Время исполнения на Intel Xeon Phi в offload режиме, сек | Время исполнения на Intel Xeon Phi в native режиме, сек | Время исполнения на Intel Xeon Phi в native режиме с Intrinsic функциями, сек |
| 1024 | 0,01165 | 0,697281 | 0,21143 | 0,222437 |
| 4096 | 0,02526 | 0,825539 | 0,219914 | 0,225469 |
| 10000 | 0,01917 | 2,043011 | 0,228653 | 0,234482 |
| 14992 | 0,03395 | 3,598321 | 0,236355 | 0,233402 |
| 20000 | 0,03970 | 5,952861 | 0,246742 | 0,250400 |
| 24992 | 0,78230 | 8,951575 | 0,242779 | 0,243994 |
| 30000 | 0,11420 | 12,77313 | 0,253075 | 0,249428 |
| 34992 | 0,15111 | 17,07816 | 0,258805 | 0,255870 |
| 40000 | 0,17425 | 24.363830 | 0,304788 | 0.281422 |
| 44992 | 0,18255 | offload error: buffer set state failed (error code 13) | terminate called after throwing an instance of 'std::bad\_alloc'  what(): std::bad\_alloc  Aborted | Segmentation fault |

Таблица исследования программы

|  |  |  |  |
| --- | --- | --- | --- |
| Размерность вектора (Size) | Время исполнения OpenMP-версии на CPU, сек | Время исполнения на Intel Xeon Phi в offload режиме, сек | Время исполнения на Intel Xeon Phi в native режиме, сек |
| 1024 | 0.004504 | 0.046728 | 0.223008 |
| 4096 | 0.013999 | 0.015165 | 0.234061 |
| 10000 | 0.048846 | 0.080304 | 0.244600 |
| 14992 | 0.089356 | 0.099163 | 0.257362 |
| 20000 | 0.165734 | 0.136039 | 0.292504 |
| 24992 | 0.208791 | 0.280327 | 2.639790 |
| 30000 | 0.366544 | 0.457054 | Killed |
| 34992 | 0.462535 | 0.429042 | Killed |
| 40000 | 0.567937 | 0.854551 | Killed |
| 44992 | 0.988734 | 1.532441 | Killed |

Сумма элементов столбцов матриц.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

#include <omp.h>

// Function for simple definition of matrix and vector elements

void DummyDataInitialization (float\* pMatrix1,float\* pMatrix2, int Size) {

int i, j; // Loop variables

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

for (j=0; j<Size; j++)

pMatrix1[i\*Size+j] = i;

pMatrix2[i\*Size+j] = i;

}

}

// Function for random definition of matrix and vector elements

void RandomDataInitialization(float\* pMatrix1, float\* pMatrix2, int Size) {

int i, j; // Loop variables

srand(unsigned(clock()));

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

for (j=0; j<Size; j++)

pMatrix1[i\*Size+j] = rand()/double(1000);

pMatrix2[i\*Size+j] = rand()/double(1000);

}

}

// Function for memory allocation and definition of objectпїЅs elements

void ProcessInitialization (float\* &pMatrix1, float\* &pMatrix2, float\* &pResult, int &Size) {

// Size of initial matrix and vector definition

do {

printf("\nEnter size of the initial objects: ");

scanf("%d", &Size);

printf("\nChosen objects size = %d\n", Size);

if (Size <= 0)

printf("\nSize of objects must be greater than 0!\n");

}

while (Size <= 0);

// Memory allocation

pMatrix1 = new float [Size\*Size];

pMatrix2 = new float [Size\*Size];

pResult = new float [Size\*Size];

// Definition of matrix and vector elements

DummyDataInitialization(pMatrix1, pMatrix2, Size);

}

#pragma offload\_attribute(push, target(mic))

void ParallelResultCalculation (float\* pMatrix1, float\* pMatrix2, float\* pResult, int Size) {

int i, j; // Loop variables

#pragma omp parallel for private (j)

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

{

for (j=0; j<Size; j++)

pResult[i\*Size+j] = pMatrix1[i\*Size+j] + pMatrix2[i\*Size+j];

}

}

#pragma offload\_attribute(pop)

// Function for computational process termination

void ProcessTermination(float\* pMatrix1,float\* pMatrix2,float\* pResult) {

delete [] pMatrix1;

delete [] pMatrix2;

delete [] pResult;

}

int main() {

float\* pMatrix1; // The first argument - initial matrix

float\* pMatrix2; // The second argument - initial matrix

float\* pResult; // Result vector for matrix-vector multiplication

int Size; // Sizes of initial matrix and vector

double \* AllResults;

struct timespec time1, time2;

double dtime;

printf("Parallel Intel Phi matrix-vector multiplication\n");

ProcessInitialization(pMatrix1, pMatrix2, pResult, Size);

clock\_gettime (CLOCK\_REALTIME, &time1);

//#pragma offload target(mic:0) in(Size) in(pMatrix1:length(Size\*Size)) in(pMatrix2:length(Size\*Size)) inout(pResult:length(Size))

ParallelResultCalculation(pMatrix1, pMatrix2, pResult, Size);

clock\_gettime (CLOCK\_REALTIME, &time2);

dtime = (time2.tv\_sec - time1.tv\_sec) + 1E-9 \* (time2.tv\_nsec - time1.tv\_nsec);

printf("\n%f seconds", dtime);

ProcessTermination(pMatrix1, pMatrix2, pResult); ;

printf("\n Time of execution: %f seconds\n", dtime);

}