Лабораторная работа № 1

Знакомство с технологией OpenMP

Цель:

Ознакомиться с технологией OpenMP. Научиться компилировать OpenMP программы. Изучить директивы технологии OpenMP.

Теоретические сведения

OpenMP[1] — это набор директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с единой памятью на языках C, C++ и Fortran. OpenMP поддерживается многими коммерческими компиляторами на различных платформах.

В настоящее время опубликована спецификация OpenMP версии 3.0. Разработку спецификации OpenMP ведут несколько крупных производителей вычислительной техники и программного обеспечения, чья работа регулируется некоммерческой организацией, называемой OpenMP Architecture Review Board (ARB).

OpenMP прост в использовании и включает лишь два базовых типа конструкций: директивы pragma и функции исполняющей среды OpenMP, которые подключаются как дополнительная библиотека. Директивы pragma, как правило, указывают компилятору реализовать параллельное выполнение блоков кода. Все эти директивы начинаются с #pragma omp. Как и любые другие директивы pragma, они игнорируются компилятором, не поддерживающим конкретную технологию — в данном случае OpenMP.

Функции OpenMP служат в основном для изменения и получения параметров среды. Кроме того, OpenMP включает API-функции для поддержки некоторых типов синхронизации. Чтобы задействовать эти функции библиотеки OpenMP времени выполнения (исполняющей среды), в программу нужно включить заголовочный файл omp.h. Если вы используете в приложении только OpenMP-директивы pragma, включать этот файл не требуется.

Функции исполняющей среды OpenMP имеют префикс *omp_*. Директивы OpenMP имеют следующий формат:

#pragma omp <директива> [раздел [[,] раздел]...]

Подробно со всеми директивами и их спецификациями можно ознакомиться в опубликованном стандарте OpenMP [1], сопроводительной документации к вашему компилятору, а также в учебном пособии Антонова А. С. «Параллельное программирование с использованием технологии OpenMP» [14].

Примеры программ

Рассмотрим пример параллельной программы, написанной с применением технологии OpenMP. Обратите внимание, в настройках некоторых компиляторов поддержка OpenMP по умолчанию может быть выключена. В этом случае при компиляции программы необходимо явно указать ключ использования OpenMP.

[1] от англ. Open Multi-Processing

Пример - перемножение матриц. Распределим задачу по двум потокам следующим образом: разделим строки итоговой матрицы пополам, и каждый поток будет находить значения ячеек в тех строках, за которые он «отвечает».

```
#include <omp.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>

#define M 3
#define N 3
#define K 4
#define INITIAL_THREADS 4
#define ROW_THREADS 2

void GetNthRow (double aMatrixA[], double aMatrixB[], double aMatrixC[],
                                unsigned m, unsigned n, unsigned k, unsigned idxRow);

void MatrixMul(double aMatrixA[], double aMatrixB[], double aMatrixC[],
                                unsigned m, unsigned n, unsigned k);

void InitMatrix(double aMatrix[], unsigned ItemsCount);

void OutMatrix(double aMatrix[], unsigned m, unsigned n);


int main(void)
{
        double* MatrixA=    new double [M*N];
        double* MatrixB=    new double [N*K];
        double* MatrixC=    new double [M*K];

        srand(time(NULL)); //Инициализация генератора ПСЧ

        printf("Initializing A matrix\n");
        InitMatrix(MatrixA, M*N);
        printf("\nInitializing B matrix\n");
        InitMatrix(MatrixB, N*K);
        printf("\n");

        MatrixMul(MatrixA, MatrixB, MatrixC, M, N, K);

        printf("\nMatrix A:\n");
        OutMatrix(MatrixA, M, N);
        printf("Matrix B:\n");
        OutMatrix(MatrixB, N, K);
        printf("Matrix C:\n");
        OutMatrix(MatrixC, M, K);

        delete []MatrixA;
        delete []MatrixB;
        delete []MatrixC;
```

```
	getchar();
	return 0;

}
#undef M
#undef N
#undef K

/*
Инициализации матрицы случайными величинами
*/
void InitMatrix(double aMatrix[], unsigned ItemsCount)
{
	#pragma omp parallel for num_threads(INITIAL_THREADS)
	for(int i=0; i < ItemsCount; i++)
	{
		aMatrix[i]= (int)(100 * rand()/(double)RAND_MAX);
		printf("Cell %d was initialized by %dth
thread\n",i+1,omp_get_thread_num()+1);
	}
}


/*
Процедура находит idxRow-ю строку произведения матриц aMatrixA*aMatrixB
согласованных размеров M*N*K
Результат помещается в соответствующую строку матрицы aMatrixC */
void GetNthRow (double aMatrixA[], double aMatrixB[], double aMatrixC[],
			unsigned m, unsigned n, unsigned k, unsigned idxRow)
{
	for(int iCell= 0; iCell<k; iCell++)
	{
		double cell= 0;
		for(unsigned i= 0; i<n; i++)
			cell+= aMatrixA[idxRow*n + i] * aMatrixB[k*i + iCell];
		aMatrixC[idxRow*k + iCell]= cell;
	}
}

/*
Процедура перемножает матрицы aMatrixA*aMatrixB согласованных размеров
M*N*K
Результат помещается в матрицу aMatrixC
*/
void MatrixMul(double aMatrixA[], double aMatrixB[], double aMatrixC[],
				unsigned m, unsigned n, unsigned k)
{
	#pragma omp parallel for num_threads(ROW_THREADS)
	/*
	Этой прагмой инициализируем параллельную часть цикла
	Каждая итерация цикла будет выполнена в одном из 2х созданных потоков
```

```
        Распределение итераций по потокам будет выполнено статически */
        for(int iRow=0; iRow<m; iRow++)
        {
                GetNthRow(aMatrixA, aMatrixB, aMatrixC, m, n, k, iRow);
                printf("%dth row is counting by %dth thread\n", iRow,
omp_get_thread_num() + 1);
        }
}

/*
Печать матриц
*/
void OutMatrix(double aMatrix[], unsigned m, unsigned n)
{
        for(unsigned i=0; i<m; i++)
        {
                for(unsigned j=0; j<n; j++)
                printf("%lf\t", aMatrix[i*n + j]); putchar('\n');
        }
        putchar('\n');
}


#undef INITIAL_THREADS
#undef ROW_THREADS
```

Дополнительное чтение

Спецификации OpenMP (англ.) [1]
Статья в Википедии (русск.) [2]
Статья в Википедии (англ.) [3]
OpenMP и C++ (русск.) [4]
Начало работы с OpenMP (русск.) [5]
Параллельное программирование с использованием технологии OpenMP (русск.) [6]

Задание к выполнению лабораторной работы

Ознакомиться со средствами для организации параллельного выполнения программы, предоставляемыми технологией OpenMP. Изучить директивы синхронизации и балансировки нагрузки OpenMP.

Разработать параллельную программу, выполняющую решение задачи согласно варианту.

Выполнить замеры времени решения задачи параллельным и последовательным вариантами программы. Сравнить полученные результаты и объяснить их.

Вариант №1

Про табулировать сложно-вычисляемую функцию на заданном отрезке с заданным шагом.

Вариант №2

Выполнить сложение двух матриц одинакового размера.

Вариант №3

Найти сумму максимальных элементов строк матрицы.

Вариант №4

Найти площадь выпуклого многоугольника, заданного координатами вершин.

Вариант №5

Найти в данном тексте все палиндромы.

Вариант №6

Найти в тексте все вхождения данного образца.

Вариант №7

Дана последовательность вещественных чисел. Сократить количество десятичных разрядов после запятой каждого числа до двух.

Вариант №8

Дана последовательность арифметических выражений, операндами которых являются однозначные числа, а число операций не больше двух. Найти значения всех выражений.

Вариант №9

Дана матрица вещественных чисел. Преобразовать матрицу таким образом, чтобы элементы ее строк шли по убыванию.

Вариант №10

Вывести все согласные, которые отсутствуют в данном тексте.

Содержание отчета

1. Название и цель работы.
2. Задание к выполнению лабораторной работы согласно варианту.
3. Описание алгоритма решения задачи в виде блок-схем или словесное. Описание используемых параллельных методов вычислений.
4. Программа в виде исходных кодов (с поясняющими комментариями), а также в откомпилированном виде для демонстрации на ЭВМ.
5. Примеры работы программы на тестовых данных. Расчеты выполнить как минимум на 4 разных порциях данных и как минимум на 4 разных количествах ядер. Рассчитать коэффициенты ускорения и построить графики зависимости коэффициента ускорения от количества используемых для расчета ядер.
6. Выводы по работе.

Библиографический список

1. OpenMP API specification [Электронный ресурс, англ.]: спецификации OpenMP - режим доступа http://openmp.org/wp/openmp-specifications/.
2. OpenMP [Электронный ресурс]: Материал из Википедии — свободной энциклопедии / Авторы Википедии // Википедия, свободная энциклопедия. - Сан-Франциско: Фон

Викимедия, 2009.— Режим доступа: http://ru.wikipedia.org/wiki/OpenMP, — свободный.

3. OpenMP [Электронный ресурс, англ.]: Материал из Википедии — свободной энциклопедии / Авторы Википедии // Википедия, свободная энциклопедия. — Сан-Франциско: Фон Викимедия, 2009.— Режим доступа: http://en.wikipedia.org/wiki/OpenMP, — свободный.
4. OpenMP и C++ [Электронный ресурс, англ.] / Канг Су Гэтлин, Пит Айсенси // MSDN Magazine. — 2005. — Октябрь. — Режим доступа: http://www.microsoft.com/Rus/Msdn/Magazine/2005/10/OpenMP. mspx
5. Начало работы с OpenMP [Электронный ресурс] / Richard Gerber // Intel. — Режим доступа: http://software.intel.com/ru- ru/articles/getting-started-with-openmp/
6. *Антонов, А.С.* Параллельное программирование с использованием технологии OpenMP / А. С. Антонов // Издательство московского университета, 2009. — ISBN 978-5211-05702-9