Лабораторная работа №1

Создание простейшей программы с использованием OpenMP

Цель работы: изучить способы создания параллельных программ для симметричных мультипроцессоров с использованием стандарта OpenMP, освоить особенности конкурентного выполнения кода в нескольких потоках.

Теоретические материалы

OpenMP – это стандартное API для написания параллельных приложений, которое основано на абстракции разделяемого адресного пространства (C, C++, Fortran)

Состоит из:
- директив компилятора
- функций времени выполнения
- переменных окружения

Спецификация стандарта поддерживается организацией OpenMP Architecture Review Board: http://www.openmp.org Актуальная спецификация стандарта – OpenMP 4.0.

- Для использования OpenMP требуется, чтобы компилятор языка программирования
  - поддерживал директивы OpenMP
  - содержал набор run-time библиотек

- Проект Visual C++, который использует OpenMP, должен компилироваться с ключом **/openmp :**
  - (Свойства конфигурации->C/C++->Язык)
- Компилятору GCC, который используется в Unix-подобных системах (или, допустим, в Code Blocks) требуется указать ключ **–fopenmp**
  - ***Примечание: при использовании GCC в окружении MinGW под Windows может оказаться, что компилятор собран без этой поддержки. Выходом из такой ситуации является только замена компилятора***

OpenMP использует модель исполнения fork-join. Из главного потока (нити) порождаются потоки-рабочие, которые по окончании сливаются снова с главным.

Для использования технологии OpenMP в коде C/C++ помимо установки флагов компилятора необходимо подключить библиотеку omp:

#include <omp.h>

после чего можно использовать директивы и библиотеку функций.

Синтаксис директив OpenMP

#pragma omp directive_name [clause[clause ...]]
<структурный блок >

Действия, соответствующие директиве, применяются непосредственно к структурному блоку, расположенному за директивой. Структурным блоком может быть любой оператор, имеющий единственный вход и единственный выход.

Количество нитей, выполняющих работу, определяется переменной окружения OMP_NUM_THREADS или вызовом функции omp_set_num_threads().

Определение нитью «своих» координат в вычислительном пространстве:

- свой номер: **omp_get_thread_num()**;

- общее число нитей: **omp_get_num_threads()**.

Как уже отмечалось выше, мастер-нить имеет номер 0, у остальных же нитей номера больше 0 и идут по порядку. При этом планировщик нитей автоматически распределяет их по доступным вычислительным ресурсам (назначает процессорам и ядрам).

Данные функции, несмотря на их простоту, очень важны, так как индивидуальный номер нити и их общее число – это параметры, на основе которых определяется, какая именно работа будет выполняться нитями.

Основная директива распараллеливания, которая задаёт параллельный блок – это
#pragma omp parallel [clause ...] newline <структурный блок >

Возможные параметры clause приведены ниже:

– **if (scalar_expression)** – нити для выполнения блока создаются только если условие в if выполняется;

– **private (list)** - определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным;

– **shared (list)** - определяет список переменных, которые будут общими для всех потоков параллельной области; правильность использования таких переменных должна обеспечиваться программистом;

– **reduction (operator: list)** - определяет оператор и переменную для редуцирования;

– **num_threads(integer-expression)** - определяет количество потоков, которые будут выполнять код; по умолчанию – все доступные на этом устройстве.

Задание на лабораторную работу

Без директив параллельных циклов и без sections/section в рамках параллельного блока решите задания, описанные ниже.

Часть 1.

1. Создайте новый проект C/C++.
2. Включите использование OpenMP в проекте.
3. Напишите простую программу, которая в 5 нитей выведет на экран сообщение:
   «Привет! Я – поток #»
   * вместо # должен быть номер нити, выполняющей вывод.

Запустите программу несколько раз подряд. Поясните результаты.

Часть 2.

Модифицируйте программу таким образом, чтобы чётные нити продолжили выводить на экран сообщение из **Части 1**, а нечётные заполнили значениями массив размером N*10, где N – номер варианта (номер студента по списку группы). При этом обеспечьте, чтобы нечётные потоки были как можно более равномерно загружены работой (разница в количестве заполняемых элементов массива, порученных разным нитям, не должна превышать 1 элемент).