**Лабораторная работа №1**

**Распараллеливание по данным для циклов**

**Цель:** изучить основные особенности распараллеливания данных по циклам в OpenMP на примере использования в рамках языка С++.

Для получения **теоретических сведений** настоятельно рекомендуется при домашней подготовке изучить материалы, представленные в списке литературы в конце разработки, а также прочие материалы по тематике лабораторной работы, представленные в открытых источниках.

Далее следует краткий конспект материала, приведенного в данных источниках, в конце включающий короткие примеры фрагментов программ.

**1. Понятие параллельной программы**

Технология OpenMP (Open Multi-Processing) позволяет разделять код на несколько потоков, в специально указываемых при помощи директив местах кода – параллельных фрагментах.

В технологии OpenMP, потоки одной и той же параллельной программы выполняются в общем адресном пространстве, что обеспечивает возможность использования общих данных для параллельно выполняемых потоков.

**2. Организация взаимодействия параллельных потоков**

Пусть два потока исполняют один и тот же программный код:

n=n+1;

для общей переменной n. Тогда эта операция может быть выполнена поочередно (что приведет к получению правильного результата), либо оба потока могут одновременно прочитать значение переменной n, увеличить и записать в эту переменную новое значение (будет получено неправильное значение). Подобная ситуация, получила наименование гонки потоков. Для исключения гонки необходимо, чтобы изменение значений общих переменных осуществлялось в каждый момент времени только одним единственным потоком – иными словами, необходимо обеспечить взаимное исключение потоков при работе с общими данными. В OpenMP взаимоисключение может быть организовано при помощи неделимых (atomic) операций, механизма критических секций (critical sections) или специального типа семафоров – замков (locks).

Помимо взаимоисключения, при параллельном выполнении программы во многих случаях является необходимым та или иная синхронизация (synchronization) вычислений, выполняемых в разных потоках: например, обработка данных, выполняемая в одном потоке, может быть начата только после того, как эти данные будут сформированы в другом потоке.

**3. Структура OpenMP**

В составе технологии OpenMP можно выделить:

- Директивы,

- Библиотеку функций,

- Набор переменных окружения.

**4. Формат директив OpenMP может быть представлен в следующем виде**:

#pragma omp <имя\_директивы> [<параметр>[[,] <параметр>]...]

Для иллюстрации приведем пример директивы:

#pragma omp parallel default(shared)

\ private(beta,pi)

Пример показывает также, что для задания директивы может быть использовано несколько строк программы – признаком наличия продолжения является знак обратного слеша «\».

**5. Директива parallel для определения параллельных фрагментов**

Для выделения параллельных фрагментов программы следует использовать директиву parallel:

#pragma omp parallel [<параметр> ...] <блок\_программы>

Для блока должно выполняться правило «один вход – один выход», т. е. передача управления извне в блок и из блока за пределы блока не допускается.

Директива parallel является одной из основных директив OpenMP. Правила, определяющие действия директивы, состоят в следующем:

- Когда программа достигает директиву parallel, создается набор потоков, исходный поток программы является основным потоком этого набора и имеет номер 0.

- Программный код блока, следующий за директивой, дублируется или может быть разделен при помощи директив между потоками для параллельного выполнения.

- В конце программного блока директивы обеспечивается синхронизация потоков – выполняется ожидание окончания вычислений всех потоков; далее все потоки завершаются – дальнейшие вычисления продолжает выполнять только основной поток (в зависимости от среды реализации OpenMP потоки могут не завершаться, а приостанавливаться до начала следующего параллельного фрагмента – такой подход позволяет снизить затраты на создание и удаление потоков).

**6. Основные понятия параллельной программы: фрагмент, область, секция**

Для обозначения динамически возникающих параллельно выполняемых участков программного кода в OpenMP используется понятие параллельных областей.

Ряд директив OpenMP допускает использование как непосредственно в блоках директивы, так и в параллельных областях. Такие директивы носят наименование отделяемых директив.

Для более точного понимания излагаемого учебного материала сведем воедино введенные термины и понятия (в скобках даются названия, используемые в стандарте 2.5):

- Параллельный фрагмент (parallel construct) – блок программы, управляемый директивой parallel.

- Параллельная область (parallel region) – параллельно выполняемые участки программного кода, динамическивозникающие в результате вызова функций из параллельных фрагментов.

- Параллельная секция (parallel section)– часть параллельного фрагмента, выделяемая для параллельного выполнения при помощи директивы **section**.

**7. Параметры директивы parallel**

Приведем перечень параметров директивы parallel:

- if (scalar\_expression)

- private (list)

- shared (list)

- default (shared | none)

- firstprivate (list)

- reduction (operator: list)

- copyin (list)

- num\_threads (scalar\_expression)

**8. Распределение вычислительной нагрузки между потоками (распараллеливание по данным для циклов)**

Для распараллеливания циклов в OpenMP применяется директива for:

#pragma omp for [<параметр> ...] <цикл\_for>

После этой директивы итерации цикла распределяются между потоками и могут быть выполнены параллельно, такое распараллеливание возможно только в случае, когда между итерациями цикла нет информационной зависимости.

Важно отметить, что для распараллеливания цикл for должен иметь «вид» цикла со счетчиком):

for (index = first; index < end; increment\_expr)

Здесь index должен быть целой переменной; на месте знака "<" в выражении для проверки окончания цикла может находиться любая операция сравнения "<=", ">" или ">=". Операция изменения переменной цикла должна иметь одну из следующих форм:

- index++, ++index,

- index--, --index,

- index+=incr, index-=incr,

- index=index+incr, index=incr+index,

- index=index-incr

Параметрами директивы for являются:

- schedule (type [,chunk])

- ordered

- ordered

- private (list)

- shared (list)

- firstprivate (list)

- lastprivate (list)

- reduction (operator: list)

**9. Управление распределением итераций цикла между потоками**

При разном объеме вычислений в разных итерациях цикла желательно иметь возможность управлять распределением итераций цикла между потоками – в OpenMP это обеспечивается при помощи параметра schedule директивы **for**. Поле type параметра schedule может принимать следующие значения:

- static – статический способ распределения итераций до начала выполнения цикла. Если поле chunk не указано, то итерации делятся поровну между потоками. При заданном значении chunk итерации цикла делятся на блоки размера chunk и эти блоки распределяются между потоками до начала выполнения цикла.

- dynamic – динамический способ распределения итераций. До начала выполнения цикла потокам выделяются блоки итераций размера chunk (если поле chunk не указано, то полагается значение chunk = 1). Дальнейшее выделение итераций (также блоками размера chunk) осуществляется в момент завершения потоками своих ранее назначенных итераций.

- guided – управляемый способ распределения итераций. Данный способ близок к предшествующему варианту, отличие в том, что начальный размер блоков итераций определяется в соответствии с некоторым параметром среды реализации OpenMP, а затем уменьшается экспоненциально (следующее значение chunk есть некоторая доля предшествующего значения) при каждом новом выделении блока итераций. При этом получаемый размер блока итераций не должен быть меньше значения chunk (если поле chunk не указано, то полагается значение chunk = 1).

- runtime – способ распределения итераций, при котором выбор конкретной схемы (из ранее перечисленных) осуществляется в момент начала выполнения программы в соответствии со значением переменной окружения OMP\_SCHEDULE. Так, например, для задания динамического способа при размере блока итераций 3, следует определить:

setenv OMP\_SCHEDULE "dynamic,3"

здесь способ распределения итераций между потоками можно менять, не корректируя при этом код программы (т. е. без повторной компиляции и сборки программы).

**10. Синхронизация вычислений по окончании выполнения цикла**

По умолчанию, все потоки, прежде чем перейти к выполнению дальнейших вычислений, ожидают окончания выполнения итераций цикла даже, если некоторые из них уже завершили свои вычисления – конец цикла представляет собой некоторый барьер, который потоки могут преодолеть только все вместе. Можно отменить указанную синхронизацию, указав параметр nowait в директиве for – тогда потоки могут продолжить вычисления за переделами цикла, если для них нет итераций цикла для выполнения.

**Примеры**

Для понимания подходов к выполнению заданий лабораторной работы приведем несколько примеров:

**Пример 1. «Hello, World!»**

Приведем практически стандартную первую программу, разрабатываемую при освоению новых языков программирования – программу, осуществляющую вывод приветственного сообщения «Hello World !» Итак:

#include <omp.h> *// заголовочный файл «omp.h»*

void main () {

*/\* Выделение параллельного фрагмента\*/*

**#pragma omp parallel**

{

printf("Hello World !\n");

}

*/\* Завершение параллельного фрагмента \*/*

}

Пример 2. Пример распараллеливания цикла

В приведенной программе для директивы parallel указаны параметры директивы shared и private, они определяют доступность данных в потоках программы – переменные, описанные как shared, являются общими для потоков; для переменных с описанием private создаются отдельные копии для каждого потока, эти локальные копии могут использоваться в потоках независимо друг от друга. Следует отметить, что если в блоке директивы parallel нет ничего, кроме директивы for, то обе директивы можно объединить в одну:

#include <omp.h>

const int NMAX=100;

void main ()

{

int i, j;

float sum;

float a[NMAX][NMAX];

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

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

a[i][j] = i+j;

*// инициализация данных*

#pragma omp parallel shared(a) private(i)

{

**#pragma omp for** private(j,sum)

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

{

sum = 0;

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

sum += a[i][j];

printf ("Сумма элементов строки %d равна %f\n",i,sum);

}

} */\* Завершение параллельного фрагмента \*/*

}

**OpenMP в MS Visual Studio 2010**

В MS Visual Studio 2010 для использования возможностей технологии OpenMP, нужно в установить директиву компиляции проекта «/openmp», для этого в пункте:

**Главное меню**->**Project**->***имя\_проекта* Properties ...**->

**Configuration Properties**->**C/C++**->**Language->Open MP Support**

следует установить значение: «**Yes (/openmp)**», также нужно подключить к проекту заголовочный файл «omp.h».

**Лабораторные задания** (№ варианта = № компьютера%2)

**Задание 1.** Написать на C++ программу в соответствии с вариантом задания, при этом программы должны оптимально использовать предлагающиеся вычислительные ресурсы. Для реализации многопоточности использовать технологию OpenMP. Измерять время работы программы для различных значений параметров. **Результаты занести в отчёт.** Также, при запуске программ, запустить диспетчер задач Windows, в нём установить счётчик потоков, и просмотреть сколько потоков выполнения использует процесс, соответствующий запущенной программе. **Результаты также занести в отчёт.**

|  |  |
| --- | --- |
| **Вариант** | **Задание: написать программы, демонстрирующие работу следующей функции:** |
| **0** | Дана матрица из MxN натуральных (ненулевых) элементов (задаются случайно).  Написать программу, считающую сумму единиц в бинарной записи числа всех попарных произведений элементов для каждой строки. |
| **1** | Дана матрица из MxN натуральных (ненулевых) элементов (задаются случайно).  Написать программу, считающую количество семёрок в десятеричной записи числа всех попарных сумм элементов для каждой строки. |

**Задание 2.** Модифицировать программу, составленную по Заданию 1, убрав возможность параллельного использования вычислительных ресурсов. Измерять время работы программы для тех же значений параметров, что были использованы при выполнении Задания 1. **Результаты сравнить и занести в отчёт.**

**Контрольные вопросы**

1. В чем состоят основы технологии OpenMP?
2. В чем состоят основные преимущества и недостатки технологии OpenMP?
3. Что понимается под параллельной программой в рамках технологии OpenMP?
4. Какие проблемы возникают при использовании общих данных в параллельно выполняемых потоках?
5. Какой формат записи директив OpenMP?
6. В чем состоит назначение директивы parallel?
7. В чем состоят понятия фрагмента, области и секции параллельной программы?
8. Как осуществляется распараллеливание циклов в OpenMP? Какие условия должны выполняться, чтобы циклы могли быть распараллелены?
9. Какие возможности имеются в OpenMP для управления распределением итераций циклов между потоками?
10. Как определяется порядок выполнения итераций в распараллеливаемых циклах?

**Требования к сдаче работы**

1. При домашней подготовке изучить теоретический материал по тематике лабораторной работы, представленный в списке литературы ниже, выполнить представленные примеры, занести в отчёт результаты выполнения.
2. Продемонстрировать выполнение лабораторных заданий.
3. Ответить на контрольные вопросы.
4. Показать преподавателю отчет.

**Литература**

1. Спецификации стандарта OpenMP (на английском языке):

<http://openmp.org/wp/openmp-specifications/>

2. Материалы, представленные на сайте intuit.ru в рамках курса «Intel Parallel Programming Professional (Introduction)»:

<http://old.intuit.ru/department/supercomputing/ppinteltt/4/>

3. С.А. Лупин, М.А. Посыпкин Технологии параллельного программирования. – М.: ИД «ФОРУМ»: ИНФРА-М, 2011. – С. 119-145. *(Глава, посвященная OpenMP)*

4. Канг Су Гэтлин, Пит Айсенси - OpenMP и C++:

<http://msdn.microsoft.com/ru-ru/library/dd335940.aspx>

5. API OpenMP C и C++

<http://msdn.microsoft.com/ru-ru/library/8y6825x5.aspx>

**Приложения**

|  |  |
| --- | --- |
| **Таблица 1. Сводный перечень директив OpenMP** | |
| **Директива** | **Описание** |
| private (list) | Параметр для создания локальных копий для перечисленных в списке переменных для каждого имеющегося потока. Исходные значения копий не определены. Директивы: parallel, for, sections, single |
| firstprivate (list) | Тоже что и параметр private и дополнительно инициализация создаваемых копий значениями, которые имели перечисленные в списке переменные перед началом параллельного фрагмента. Директивы: parallel, for, sections, single |
| lastprivate (list) | Тоже что и параметр private и дополнительно запоминание значений локальных переменных после завершения параллельного фрагмента. Директивы: for, sections |
| shared (list) | Параметр для определения общих переменных для всех имеющихся потоков. Директивы: parallel |
| default (shared | none) | Параметр для установки правила по умолчанию на использование переменных в потоках. Директивы: parallel |
| reduction (operator: list) | Параметр для задания операции редукции. Директивы: parallel, for, sections |
| nowait | Параметр для отмены синхронизации при завершении директивы. Директивы: for, sections, single |
| if (expression) | Параметр для задания условия, только при выполнении которого осуществляется создание параллельного фрагмента. Директивы: parallel |
| ordered | Параметр для задания порядка вычислений в распараллеливаемом цикле. Директивы: for |
| schedule (type [, chunk]) | Параметр для управления распределением итераций распараллеливаемого цикла между потоками. Директивы: for |
| copyin (list) | Параметр для инициализации постоянных переменных потоков. Директивы: parallel |
| copyprivate (list) | Копирование локальных переменных потоков после выполнения блока директивы single. Директивы: single |
| num\_treads | Параметр для задания количества создаваемых потоков в параллельной области. Директивы: parallel |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Таблица 2. Сводная таблица по использованию параметров в директивах OpenMP** | | | | | | |
| **Параметр** | **Шапка таблицы** | | | | | |
| **parallel** | **for** | **sections** | **single** | **parallel for** | **parallel sections** |
| **if** | x |  |  |  | x | x |
| **private** | x | x | x | x | x | x |
| **shared** | x | x |  |  | x | x |
| **default** | x |  |  |  | x | x |
| **firstprivate** | x | x | x | x | x | x |
| **lastprivate** |  | x | x |  | x | x |
| **reduction** | x | x | x |  | x | x |
| **copyin** | x |  |  |  | x | x |
| **schedule** |  | x |  |  | x |  |
| **ordered** |  | x |  |  | x |  |
| **nowait** |  | x | x | x |  |  |
| **num\_threads** | x |  |  |  | x |  |
| **copypri-vate** |  |  |  | x |  |  |

|  |  |
| --- | --- |
| **Таблица 3. Сводный перечень функций OpenMP** | |
| **Функция** | **Описание** |
| void omp\_set\_num\_threads (int num\_threads) | Установить количество создаваемых потоков |
| int omp\_get\_max\_threads (void) | Получение максимально-возможного количества потоков |
| int omp\_get\_num\_threads (void) | Получение количества потоков в параллельной области программы |
| int omp\_get\_thread\_num (void) | Получение номера потока |
| int omp\_get\_num\_procs (void) | Получение числа вычислительны элементов (процессоров или ядер), доступных приложению |
| void omp\_set\_dynamic (int dynamic) | Установить режим динамического создания потоков |
| int omp\_get\_dynamic (void) | Получение состояние динамического режима |
| void omp\_set\_nested (int nested) | Установить режим поддержки вложенных параллельных фрагментов |
| int omp\_get\_nested (void) | Получения состояние режима поддержки вложенных параллельных фрагментов |
| void omp\_init\_lock(omp\_lock\_t \*lock)  void omp\_init\_nest\_lock(omp\_nest\_lock\_t \*lock) | Инициализировать замок |
| void omp\_set\_lock (omp\_lock\_t &lock)  void omp\_set\_nest\_lock (omp\_nest\_lock\_t &lock) | Установить замок |
| void omp\_unset\_lock (omp\_lock\_t &lock)  void omp\_unset\_nest\_lock (omp\_nest\_lock\_t &lock) | Освободить замок |
| int omp\_test\_lock (omp\_lock\_t &lock)  int omp\_test\_nest\_lock (omp\_nest\_lock\_t &lock) | Установить замок без блокировки |
| void omp\_destroy\_lock(omp\_lock\_t &lock)  void omp\_destroy\_nest\_lock(omp\_nest\_lock\_t &lock) | Перевод замка в неинициализированное состояние |
| double omp\_get\_wtime (void) | Получение времени текущего момента выполнения программы |
| double omp\_get\_wtick (void) | Получение времени в секундах между двумя последовательными показателями времени аппаратного таймера |
| int omp\_in\_parallel (void) | Проверка нахождения программы в параллельном фрагменте |

|  |  |
| --- | --- |
| **Таблица 4. Сводный перечень переменных окружения OpenMP** | |
| **Переменная** | **Описание** |
| OMP\_SHEDULE | Переменная для задания способа управления распределением итераций распараллеливаемого цикла между потоками. Значение по умолчанию: static |
| OMP\_NUM\_THREADS | Переменная для задания количество потоков в параллельном фрагменте. Значение по умолчанию: количество вычислительных элементов (процессоров/ядер) в вычислительной системе. |
| OMP\_DYNAMIC | Переменная для задания динамического режима создания потоков. Значение по умолчанию: false. |
| OMP\_NESTED | Переменная для задания режима вложенности параллельных фрагментов. Значение по умолчанию: false. |