**Лабораторная работа №3**

**Использование редукции**

**Цель:** изучить основные особенности использования редукции 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. Определение общих и локальных переменных

Параметр shared определяет переменные, которые будут общими для всех потоков. Параметр private указывает переменные, для которых в каждом потоке будут созданы локальные копии – они будут доступны только внутри каждого потока в отдельности (значения локальных переменных потока недоступны для других потоков). Параметры shared и private могут повторяться в одной и той же директиве несколько раз, имена переменных должны быть уже ранее определены и не могут повторяться в списках параметров shared и private. По умолчанию все переменные программы являются общими. Такое соглашение приводит к тому, что компилятор не может указать на ошибочные ситуации, когда программисты забывают описывать локальные переменные потоков в параметре private (отсутствие таких описаний приводит к тому, что переменные будут восприниматься как глобальные). Для выявления таких ошибок можно воспользоваться параметром default директивы parallel для изменения правила по умолчанию:

default ( shared | none )

При помощи этого параметра можно отменить действие правила по умолчанию (default(none)) или восстановить правило, что по умолчанию переменные программы являются общими (default(shared)).

Следует отметить, что начальные значения локальных переменных не определены, а конечные значения теряются при завершении потоков. Для их инициализации можно использовать параметр firstprivate директивы for, по которому начальные значения локальных переменных будут устанавливаться в значения, которые существовали в переменных до момента создания локальных копий. Запоминание конечных значений обеспечивается при помощи параметра lastprivate, в соответствии с которым значения локальных переменных копируются из потока, выполнившего последнюю итерацию.

##### 

##### 2. Совместная обработка локальных переменных (операция редукции)

Использование параметра lastprivate позволяет сохранить значения локальной переменной одного из потоков, но во многих случаях для обработки могут понадобиться значения всех локальных переменных. Данная возможность может быть обеспечена, например, сохранением этих значений в общих переменных – более подробно правила работы с общими переменными будет рассмотрена в следующем разделе. Другой подход состоит в использовании коллективных операций над локальными переменными, предусмотренными в OpenMP. Задание коллективной операции происходит при помощи параметра reduction директивы for:

reduction (operator: list)

где список list задает набор локальных переменных (повторное описание в параметре private переменных из списка list не требуется), для которых должна быть выполнена коллективная операция, а поле operator указывает тип этой операции. Допустимыми значениями для поля operator являются следующие операции (которые не могут быть перегружены):

+, -, \*, &, |, ^, &&, ||

Операция редукции обычно применяется к переменным, для которых вычисления задаются в виде выражений следующего вида:

- x = x <operator> <expression>

- x = <expression> <operator> x (за исключением операции вычитания)

- x <op>= <expression>

- x++, ++x, x--, --x

где x есть имя скалярной переменной, выражение expression не должно включать переменную x, возможные операции для поля operator совпадают с вышеприведенным списком, а допустимыми значениями для поля op являются операции:

+, -, \*, &, |, ^

**Пример использования редукции данных**

total = 0;

#pragma omp parallel shared(a)

{

#pragma omp for private(i,j,sum) **reduction(+:total)**

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);

total = total + sum;

}

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

printf ("Сумма элементов матрицы равна %f\n",total);

**Лабораторные задания** (№ варианта = № компьютера%5)

**Задание 1.** В соответствии с вариантом задания, написать на C++ программу программу, реализующую многопоточность на основе технологии OpenMP: она должна вычислять значение функции с помощью редукции. Измерять время работы программ для различных значений параметра N. **Результаты занести в отчёт.**

|  |  |
| --- | --- |
| **Вариант** | **Задание: написать программы, демонстрирующие работу следующей функции:** |
| **0** | На основе двух равно размерных массивов A и B (длины N) функция формирует массив C, таким образом, что: , и возвращает сумму значений элементов массива C. |
| **1** | На основе двух равно размерных массивов A и B (длины N) функция возвращает произведение ненулевых значений |
| **2** | На основе двух равно размерных массивов A и B (длины N) функция возвращает сумму значений , б**о**льших 1. |
| **3** | На основе трёх равно размерных массивов A, B и C (длины N) функция возвращает произведение ненулевых значений, полученных таким образом: если четно: , иначе |
| **4** | На основе трёх равно размерных массивов A, B и C (длины N) функция возвращает сумму значений, полученных таким образом: если четно: , иначе , неравных 1. |

**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. Как определяются общие и локальные переменные потоков?
2. Что понимается под операцией редукции?

**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>