Параллельные циклы в OpenMP

Если в параллельной области встречается цикл, то он будет выполнен всеми нитями текущей группы, каждая нить выполнит все итерации данного цикла. Для распределения итераций цикла между различными нитями можно использовать директиву for:

```
#pragma omp for [ опция .., опция .. ]
```

Эта директива относится к идущему следом за данной директивой блоку, включающему оператор `for`.

Параллельные циклы в OpenMP

Существует некоторый набор опций для директивы for:

Опция `private` (список переменных) - список переменных локального хранения для каждой нити.

Опция `firstprivate` (список переменных) - получают свои значения из последовательной части программы.

Параллельные циклы в OpenMP

Опция `lastprivate` (список переменных) - сохраняют свои значения при выходе из параллельных нитей.

Опция `reduction` (оператор : список переменных) - выполняет действия с указанными переменными.

Опция `schedule` (тип [, размер итерации]) - задает каким образом итерации цикла распределятся между нитями.

Параллельные циклы в OpenMP

Опция `collapse` (uint) - указывает, что некоторое число последовательных тесновложенных циклов ассоциируется данной директивой. Для циклов образуется общее пространство итераций, которое делится между нитями. Если опция `collapse` не задана, то директива `for` относится только к одному непосредственно следующему за ней циклу.

Опция `ordered` - указывает на то, что цикле могут встречаться директивы `ordered`. В таком случае определяется блок в теле цикла, который должен будет выполниться в том порядке, в котором итерации идут в последовательном цикле.

Опция `nowait` - отменяет в конце параллельного цикла барьерную синхронизацию.

Параллельные циклы в OpenMP

На вид циклов, которые можно сделать параллельными, накладываются ограничения. Как правило это должен быть цикл for. Предполагается, что не должно быть зависимостей в цикле, программа не должна зависеть от того, какая именно нить какую итерацию цикла выполняет. Нельзя использовать побочные выходы из параллельного цикла. Размер блока итераций, который указывается в блоке schedule, не должен изменяться в рамках цикла. Формат параллельного цикла в упрощенной форме может выглядеть таким образом:

```
for([целочисленный тип] i = инвариант цикла; i
(>,<,=,<=,>=) инвариант цикла; i (+,-)= инвариант
цикла)
```

Параллельные циклы в OpenMP

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

int main(int argc, char * argv[]){

    int A[10], B[10], C[10], i, n;

    for(i=0; i<10; i++){ A[i] = i; B[i] = 2*i; C[i] = 0;}

    #pragma omp parallel shared(A,B,C) private(i,n)
    {
        n = omp_get_thread_num();
        #pragma omp for
        for(i=0; i<10; i++){
            C[i]=A[i]+B[i];
            printf("Нить %d сложила элементы %d\n", n, i);
        }
    }

    return 0;
}
```

Итерации цикла, помеченного директивой for, будут распределены между теми нитями, которые порождены директивой parallel.

Параллельные циклы в OpenMP

В опции `schedule` параметр `type` задаёт несколько именованных типов распределения итерации:

`static` - блочно-циклическое распределение итераций цикла. Размер блока - chunk. Первый блок из chunk итераций выполняет нулевая нить, второй блок -следующая и так далее до последней нити, затем распределение начинается снова с нулевой нити. Если значение chunk не указано, то всё множество итераций делится на части примерно одинакового размера и все нити получат примерно одинаковую нагрузку, либо нескольким последним нитям достанется чуть меньше, в том случае если число итераций нацело не делится.

Параллельные циклы в OpenMP

`dynamic` - динамическое распределение итераций цикла с фиксированным размером блока. Для начала каждая нить получает `chunk` итераций (по умолчанию `chunk = 1`). Как только одна из нитей заканчивает выполнений своей порции итераций, то ей отдаются новые порцию из `chunk` итераций. Последняя порция может содержать меньше итераций, чем все остальные.

Параллельные циклы в OpenMP

`guided` - динамическое распределение итераций цикла, при котором размер порций уменьшается с некоторого начального значения до величины `chunk` (по умолчанию `chunk` = 1) пропорционально количеству ещё не распределённых итераций, делённому на количество нитей, выполняющих цикл. Размер первоначально выделяемого блока зависит от реализации. Это может быть необходимо для более сбалансированной загрузки потоков.

Параллельные циклы в OpenMP

`auto` - способ распределения итераций вбирается компилятором и/или системой выполнения. Параметр `chunk` при этом не задается и выбирается автоматически.

runtime - способ распределения итераций выбирается во время работы по значению переменной среды `OMP_SCHEDULE`. Параметр `chunk` при этом не задается.

Параллельные циклы в OpenMP

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

int main(int argc, char ** argv){
  int i;
  #pragma omp parallel private(i)
  {
    #pragma omp for schedule(static, 2)
    {
      for( i=0 ; i < 10 ; i++){
        printf('Нить %d выполнила итерацию %d\n',
        omp_get_thread_num(), i);
        sleep(1);
      }
    }
  }
}
```

Параллельные циклы в OpenMP

Значения по умолчанию переменной OMP_SCHEDULE зависит от реализации. Если переменная задана неправильно, то поведение программы при задании опции runtime также зависит от реализации.

Задать значение переменной OMP_SCHEDULE можно с помощью команды следующего вида:
Export OMP_SCHEDULE="dynamic,1"

Параллельные циклы в OpenMP

Изменить значение переменной `OMP_SCHEDULE` из программы можно с помощью вызова функции `omp_set_schedule()`. Процедура имеет вид

```
void omp_set_schedule(omp_sched_t type, int chunk)
```

Допустимые значения констант описаны в файле omp.h. Существуют, как минимум, следующие варианты:

```
typedef enum omp_sched_t {
omp_sched_static=1,
omp_sched_dynamic=2,
omp_sched_guided=3,
omp_sched_auto=4
}
```

Параллельные циклы в OpenMP

При помощи процедуры `omp_get_schedule()` есть возможность проверить текущее значение `OMP_SCHEDULE`.

```
void omp_get_schedule( omp_sched_t * type,
int* chunk);
```

При распараллеливании цикла нужно убедиться, что итерации данного цикла не имеют информационных зависимостей. В таком случае итерации можно будет выполнить в любом порядке, в том числе параллельно. Компилятор не проверяет информационную зависимость в циклах, возможны логические ошибки в результатах программы если этого не учесть.

Параллельные циклы в OpenMP

Директива `sections` определяет набор независимых секций кода, каждая из которых выполняется своей нитью.

#pragma omp sections [опции …,]