Введение в OpenMP (практика)

Обзор

- OpenMP — простое средство параллельного программирования **на одном компьютере** с использованием **общей памяти**
  - Единица исполнения – **поток** (thread, нить), а не процесс
- Разработка управляется OpenMP ARB. Членами являются Intel, AMD, ARM, IBM и др. **пользователи и производители суперкомпьютеров**
- OpenMP – API на **C, C++, Fortran**
- Код может быть без изменений **скомпилирован** компиляторами, **не поддерживающими** OpenMP

Концепция прагм

- `#pragma` — способ управления **implementation-defined** поведением **компилятора**
  - Microsoft: Способ обеспечения специальных компьютерных функций и функций ОС при сохранении общей совместимости со стандартом C/C++
- Если компилятор не распознаёт директиву, она игнорируется
- Все директивы OpenMP имеют вид

```
#pragma omp ...
```

Реализация многопоточности

- У директив, как правило, есть **область действия**, определяемая обычным *блоком* C/C++: `{...}`
- Вне действия директив существует один «основной» поток
- Директива

  `#pragma omp parallel`

  создаёт несколько потоков, которые (вместе с «основным») **задействуются последующими** директивами
- Критические секции выполняет только «основной» поток
- В пределах действия директив каждая переменная имеет свойство `shared` или `private`

Простая программа

```
#include<stdio.h>
#include<omp.h>  /* заголовочный файл OpenMP */

int main () {
#ifdef _OPENMP  /* идентификатор версии OpenMP */
    printf("OpenMP is supported: %d\n", _OPENMP);
#else
    printf("OpenMP is not supported, exiting\n");
    return -1;
#endif

    int myid, num_procs, num_threads;
    int a[10];
    int i;
```

Простая программа

```
num_procs = omp_get_num_procs();  /* Число CPU */
printf("Number of processors = %d\n", num_procs);

/* Явное задание числа threads */
omp_set_num_threads(4);
/* Число существующих сейчас threads */
num_threads = omp_get_num_threads();
printf("Number of threads  = %d\n", num_threads);

myid = omp_get_thread_num();  /* Номер текущего thread */
printf("Main thread ID = %d\n", myid);
```

Простая программа

```
    /* Создание дополнительных threads */
    #pragma omp parallel shared(a) private(myid, i)
    {
        /* Параллельная секция */
        num_threads = omp_get_num_threads();
        printf("Number of threads  = %d \n", num_threads);
        myid = omp_get_thread_num();
        printf("Parallel part, myid = %d\n", myid);
    }

    myid = omp_get_thread_num();
    printf("Main thread ID (2) = %d\n", myid);

    Return 0;
}
```

Компиляция и запуск

В gcc есть встроенная поддержка OpenMP.

https://gcc.gnu.org/wiki/openmp

```
gcc -fopenmp ...
```

Параллельный цикл

Полезное использование OpenMP — распараллеливание цикла. Осуществляется директивой `#pragma omp for`

Каждая итерация цикла попадает в произвольный поток.

```
    #pragma omp for
    for (i = 0; i < 10; i++) {
        a[i] = a[i] * 2;
        printf("Thread %d has multiplied element %d\n",
myid, i);
    }
```

Коллапс. Опции

```
#pragma omp for collapse(n) nowait
```

`collapse(n)` – образовать общий объём итераций из `n` последовательных **вложенных** циклов

`nowait` – отключить синхронизацию (barrier) после цикла

Синхронизация

```
#pragma omp barrier  /* барьер */

#pragma omp critical  /* критическая секция */
{ … }

/* атомарная операция с shared переменной */
#pragma omp atomic
shared_counter = shared_counter + 1
```

Elapsed time

```
double omp_get_wtime()
```

Возвращает время в секундах от некоторого момента в прошлом.

Гарантируется, что часы монотонные.

Полезные материалы

Примеры кода на OpenMP (и не только)

https://github.com/akhtyamovpavel/ParallelComputationExamples

Спецификация OpenMP v3.0

https://www.openmp.org//wp-content/uploads/spec30.pdf

Реализация OpenMP в GCC

https://gcc.gnu.org/onlinedocs/gcc-4.9.0/libgomp.pdf