OpenMP总结

2019 年 2 月 19 日

1 简介

一段典型的程序可由下图描述。其中主线在几个结点上，可以并行运行，随后再汇总成一个线程。

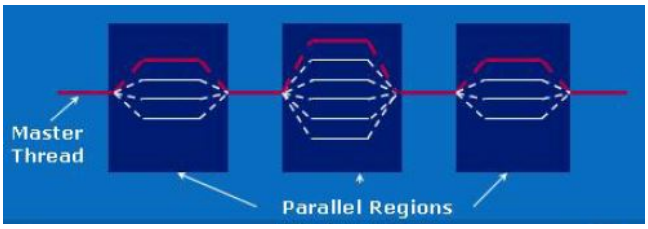

图 1: 程序结构

使用OpenMP可以将单一线程改造成多线程执行。只需在程序中注明相应指令，编译器自动将代码转换成并行代码。

举例：

```
#pragma omp parallel  // omp 指令
{
    int i;
    printf("hello world\n");
    for(i=0;i<6;++i){
        printf("iter:%d\n",i);
    }
}
```

这样编译器会自动将代码并行化。

2 OpenMP指令初步

#pragma omp parallel for 可以将for循环并行化。

举例：

```
#pragma omp parallel for  // omp 指令
for(i=0;i<6;++i){
    function();
}
```

#pragma omp parallel for shared()/ private() 可以限定变量的属性：是多个线程共享还是私有。

共享变量在多个并行进程间共享，因此多个线程之间会相互干扰；而私有变量则不会受其他线程的干扰。

举例：

```
int x,y;
#pragma omp parallel for private(x,y) // omp 指令
for(i=0;i<6;++i){
    x = a[i];
    y = b[i];
    function(x,y);
}
```

其中x，y为本线程的中间变量，不希望收到其他线程的干扰，所以才使用私有的方式。

如果想要某变量对所有线程可见，那就需要使用共享模式。

举例：

```
int shared_variable = 0;
#pragma omp parallel for shared(shared_variable) // omp 指令
for(i=0;i<6;++i){
    function(shared_variable);
}
```

#pragma omp critical 但是有些变量较为特殊，无论共享还是私有都不对，比如累加操作sum。它既希望对多个线程可见，又不希望多个线程同时修改该变量，出现不可预知的操作。

举例：

```
int sum = 0;
#pragma omp parallel for shared(sum) // omp 指令
for(i=0;i<6;++i){
    #pragma omp critical // omp 指令
    sum += a[i]+b[i];
}
```

reduction() 类似上述加和操作sum这样的变量，可以使用规约(reduction)操作。这类变量会将多个线程的处理结果，汇总成一个变量。

因此上述代码可以改为

```
#pragma omp parallel for reduction(+:sum) // omp 指令
for(i=0;i<6;++i){
    #pragma omp critical // omp 指令
    sum += a[i]+b[i];
}
```

其中，“+:”符号代表累加操作。另外还支持以下操作：

Operator	Initial Value
+	0
*	1
-	0
^	0

Operator	Initial Value
&	~0
|	0
&&	1
||	0

图 2: 规约操作

进程调度（schedule）

schedule(static/dynamic/guided)

static 静态调度。比如某代码需要100次迭代，使用代码schedule(static，4)，则原循环被分成100/4=25 块。其中某块将会被分给某线程执行。

dynamic 动态调度。上述的某块代码将会被动态分配给某线程。

guided 类似动态调度，开销较小。

section 上述过程采用自动的方式将程序分块，使用section可以手动将程序分块。

举例：

```
#pragma omp parallel sections // omp 指令
{
    #pragma omp section
    red();
    #pragma omp section
    green();
    #pragma omp section
    yellow();
}
```

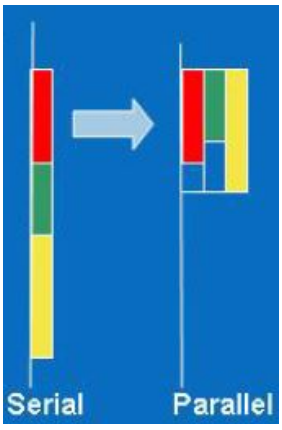

图 3: section

这样，上述三个操作将在逻辑上并行执行（具体执行过程，考虑操作系统的调度，实际CPU核的数量）。

single/master single可以规定某操作只执行一遍。

举例：

```
#pragma omp parallel  // omp 指令
{
    function1();
    #pragma omp single
    {
        function2();
    }
    function3();
}
```

上述代码中，function2()只会被一个线程执行，并不会被每个线程执行。类似操作还有master，规定只有主线程可以执行对应代码。

3 OpenMP API

OpenMP还提供相应的API函数，比如：

1. omp_get_thread_num()，返回当前线程ID
2. omp_get_num_threads()，返回并行线程个数