实验六：OpenMP并行编程

郑海刚

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

本讲概述

- 主要内容
  - OpenMP介绍
  - 编译指令
  - 库函数
  - 环境变量
  - DGEMM的OpenMP实现

为什么要有OpenMP

- pthreads（POSIX Threads）并行化的不足：
  - 扩展性问题：硬件升级，CPU核数增加，可能需要修改代码
  - 移植性问题：windows不支持pthreads
  - 编程不友好，非计算机专业用户不易掌握pthreads api，比如函数指针的用法
- 更高级的：**共享内存**多线程编程模型
  - OpenMP(Open Multi-Processing)

OpenMP demo：hello world

- lab6/openmp/hello.c
  - 关键语句：#pragma omp parallel
  - gcc hello.c -o hello
  - gcc -fopenmp hello.c -o hello-omp
- 自动创建线程自动并行化

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

int main(void)
{
    #pragma omp parallel
    printf("Hello, world.\n");
    return 0;
}
```

```
$ hpc_practice/lab6 »./hello-omp
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
```

是否真的创建了多个线程

- lab6/openmp/hello-sleep.c
  - gcc -fopenmp hello-sleep.c -o hello-sleep
  - export OMP_NUM_THREADS=4
  - ./hello-sleep执行

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

int main(void)
{
    #pragma omp parallel
    {
        printf("Hello, world.\n");
        sleep(60);
    }
    return 0;
}
```

- 在另外一个终端查看是否多个线程
  - ps -ef | grep hello-sleep | grep -v grep |awk '{ print $2 }' | head -n 1 | xargs pstree -p

OpenMP 介绍

- openmp wikipedia
  - API标准：支持C/C++，Fortran、多指令集、多操作系统
  - 使用方式：编译器指令、运行时库、环境变量
  - 功能：线程创建、工作共享结构、环境管理、线程同步等
  - 早期是为了并行化数值计算中常规的循环（loop）

OpenMP编译器指令（compiler directives）

- 编译指令也叫编译指导语句：#define、 #include 是预处理指令
- #pragma omp parallel 是openmp编译指令
  - #pragma omp 是必须的
  - parallel 是directive-name，即具体的指令
  - [clause, ...] 子句，可选

C / C++ Directives Format

▶ **Format:**

#pragma omp	directive-name	[clause, ...]	newline
Required for all OpenMP C/C++ directives.	A valid OpenMP directive. Must appear after the pragma and before any clauses.	Optional. Clauses can be in any order, and repeated as necessary unless otherwise restricted.	Required. Precedes the structured block which is enclosed by this directive.

▶ **Example:**

```
#pragma omp parallel default(shared) private(beta,pi)
```

OpenMP Directives常用命令：parallel

- parallel
  - 用在一个代码块之前，表示这段代码将被多个线程并行执行，创建一个并行域
  - 当遇到parallel指令，会创建一组线程执行，原来的线程为主线线程，线程id为0
  - 所有的线程执行相同的代码
  - 线程数有多种方式可以设定

OpenMP Directives常用命令：for 、 parallel for

- for
  - for：用于for循环之前，将循环分配到多个线程中并行执行
  - for循环会划分为尽可能等长的部分，分配给不同的线程，分配方式由编译器决定
  - parallel for：parallel和for的结合，parallel创建多个线程，for将循环的工作分配到各个线程
    - 使用for的时候必须是parallel region，单独使用没有意义，否则只有一个线程仍将串行执行

工作共享：Work-sharing constructs

- 又叫 workload distribution（工作分发）
- parallel只是并行化，每个线程执行的任务是独立的，不涉及任务拆分
- for将一个大的任务拆分成多个任务，每个线程执行一部分
- work-sharing 除了for命令之外，还有
  - sections
  - single
  - workshare

工作共享demo：for循环

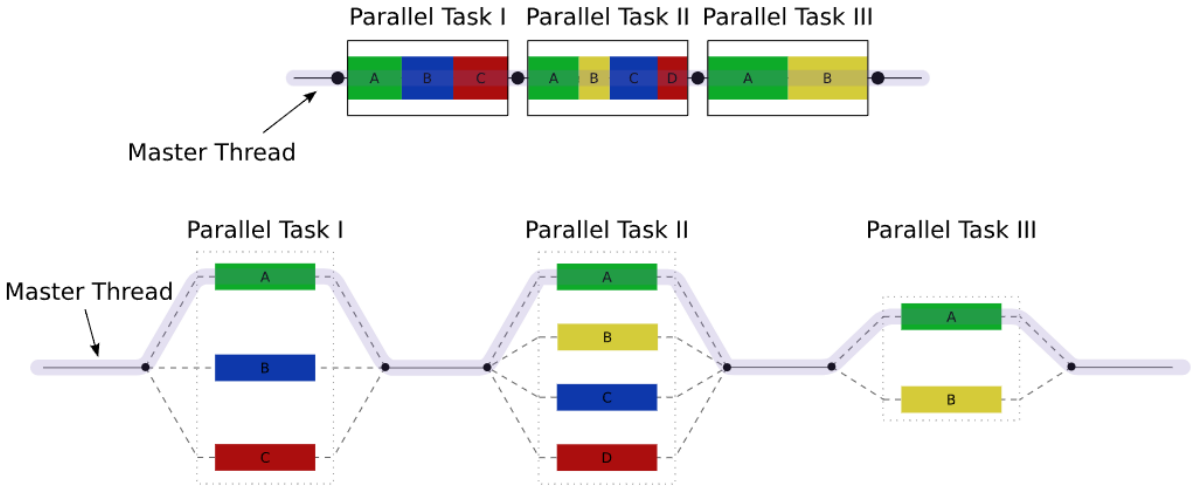
- lab6/openmp/sum-omp.c
  - 输出乱序，不一定按照i的顺序

```c
#include<stdio.h>
#include<unistd.h>

int main(){
    int N=4;
    int a[N],b[N],c[N];
    #pragma omp parallel for
    for(int i=0; i<N; i++){
        a[i]=i;
        b[i]=i*i;
        c[i]=a[i]+b[i];
        printf("c[%d]=%d\n",i,c[i]);
    }
}
```

```
$ hpc_practice/lab6 »./sum-omp
c[0]=0
c[3]=12
c[2]=6
c[1]=2
```

fork-join工作模型



[图片来源]

- 原工作流：串行，图的上半部分
- 优化后的工作流：串行+并行，图的下半部分，在执行parallel并行区域之前是单线程

OpenMP Directives 常用子句

- 子句的作用：行为或属性的设置
  - private：指定变量是线程私有的
  - shared：指定变量为多个线程间共享
  - num_threads：指定线程的个数
- 并行区域外定义的变量默认为多个线程shared，并行区域内定义的变量默认为多个线程private

两层for循环：怎么分发?

- lab6/openmp/double-loop.c
  - 变量i, j在for循环中声明
  - 变量i，j属性是private，不会相互干扰
  - 只有第一层循环被分发（collapse）
- collapse子句指定展开的层级

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

int main(){
    #pragma omp parallel for
    for (int i = 0; i < 2; ++i)
    {
         for (int j = 0; j < 3; ++j)
        {
            printf("i=%d j=%d\n from thread = %d\n",
                    i, j, omp_get_thread_num());
        }
     }
```

```
$ hpc_practice/lab6 »./double-loop
i=0 j=0
 from thread = 0
i=0 j=1
 from thread = 0
i=0 j=2
 from thread = 0
i=1 j=0
 from thread = 1
i=1 j=1
 from thread = 1
i=1 j=2
 from thread = 1
$ hpc_practice/lab6 »
```

两层for循环：变量属性

- lab6/openmp/double-loop-j.c
  - 变量j在 parallel for之前申明
  - 输出变少了
  - 变量j属性成了shared，thread 0修改了j，thread 1能立即看到，使得条件j<3提前结束
- i仍是private：被collapse的循环变量是private，不管在并行区域外还是内定义，会被分配成不同的初始值

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

int main(){
    int i = 1;
    int j = 1;
    #pragma omp parallel for
    for (i = 0; i < 2; ++i)
    {
         for (j = 0; j < 3; ++j)
        {
            printf("i=%d j=%d\n from thread = %d\n",
                    i, j, omp_get_thread_num());
        }
     }
}
```

```
$ hpc_practice/lab6 »./double-loop-j
i=0 j=0
 from thread = 0
i=0 j=1
 from thread = 0
i=0 j=2
 from thread = 0
i=1 j=0
 from thread = 1
$ hpc_practice/lab6 »
```

怎么检查变量的shared、private属性

- OpenMP没有提供查询变量属性的库函数
- 但既然都在一个进程，可以通过地址判断
  - lab6/openmp/double-loop-attr.c
  - 两个thread中看到的变量i的地址不同，j的地址相同
- 只要有写操作（对变量赋值），就要考虑变量是否是shared

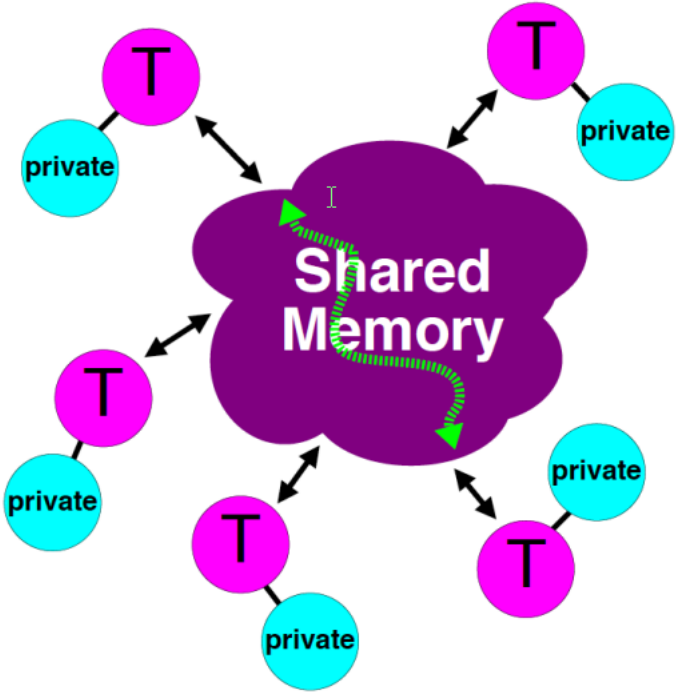
```c
#include<stdio.h>
#include<unistd.h>
#include<omp.h>

int main(){
    int i = 1;
    int j = 1;
    #pragma omp parallel for
    for (i = 0; i < 2; ++i)
    {
         for (j = 0; j < 3; ++j)
        {
            printf("i=%d j=%d\n from thread = %d\n",
                    i, j, omp_get_thread_num());
            printf("i:%p, j:%p\n",&i,&j);
        }
    }
}
```

```
$ hpc_practice/lab6 »./double-loop-attr
i=0 j=0
 from thread = 0
i:0x7ffcb90ba624, j:0x7ffcb90ba68c
i=0 j=1
 from thread = 0
i:0x7ffcb90ba624, j:0x7ffcb90ba68c
i=0 j=2
 from thread = 0
i:0x7ffcb90ba624, j:0x7ffcb90ba68c
i=1 j=0
 from thread = 1
i:0x7fcd143fbe14, j:0x7ffcb90ba68c
$ hpc_practice/lab6 »
```

OpenMP 共享内存模型

- 硬件上不同CPU核访问内存是一样的
- 线程之间共享内存



Structured Blocks

- 编译指令后面跟的语句必须符合Structured Blocks要求
- OpenMP-API-Specification-5-2.pdf：P76
  - （1）循环语句
  - （2）用一对大括号{}括起来的语句
  - what does structured-block refer to?
- 限制：单一的入口单一的出口
  - 入口不能是分支
  - 出口不能是分支：break、goto、setjmp、longjmp

for 循环语句块

- Structured Blocks： P2页 1.2.2节
  - for 循环语句块内不能有break、return
  - 可以有exit等进程退出的函数
- lab6/openmp/sum-omp.c 加一行break语句，编译出错

```
$ hpc_practice/lab6 »gcc -fopenmp sum-omp.c -o sum-omp
sum-omp.c: In function ‘main’:
sum-omp.c:12:18: error: break statement used with OpenMP for loop
   12 |         if(i==2) break;
      |                  ^~~~~
$ hpc_practice/lab6 »
```

OpenMP Runtime库函数

- lab6/openmp/omp-runtime.c
  - 头文件：#include <omp.h>
  - 函数：
    - omp_get_thread_num()
    - omp_get_num_threads()
    - ...

OpenMP 环境变量

- Environment variable wikipedia
  - env 查看系统所有环境变量
  - echo $xx 显示单个变量
  - export 设置当前环境变量
- OMP_NUM_THREADS
  - lab6/openmp/hello.c
    - export OMP_NUM_THREADS=2
    - ./hello-omp

```
$ hpc_practice/lab6 »export OMP_NUM_THREADS=2
$ hpc_practice/lab6 »./hello-omp
Hello, world.
Hello, world.
$ hpc_practice/lab6 »export OMP_NUM_THREADS=4
$ hpc_practice/lab6 »./hello-omp
Hello, world.
Hello, world.
Hello, world.
Hello, world.
```

OpenMP主要参考材料

- OpenMP-API-Specification-5-2.pdf、 OpenMP3.1
- Wikipedia: OpenMP
- Lawrence Livermore National Laboratory: OpenMP
- Purdue University ECE563： OpenMP Tutorial