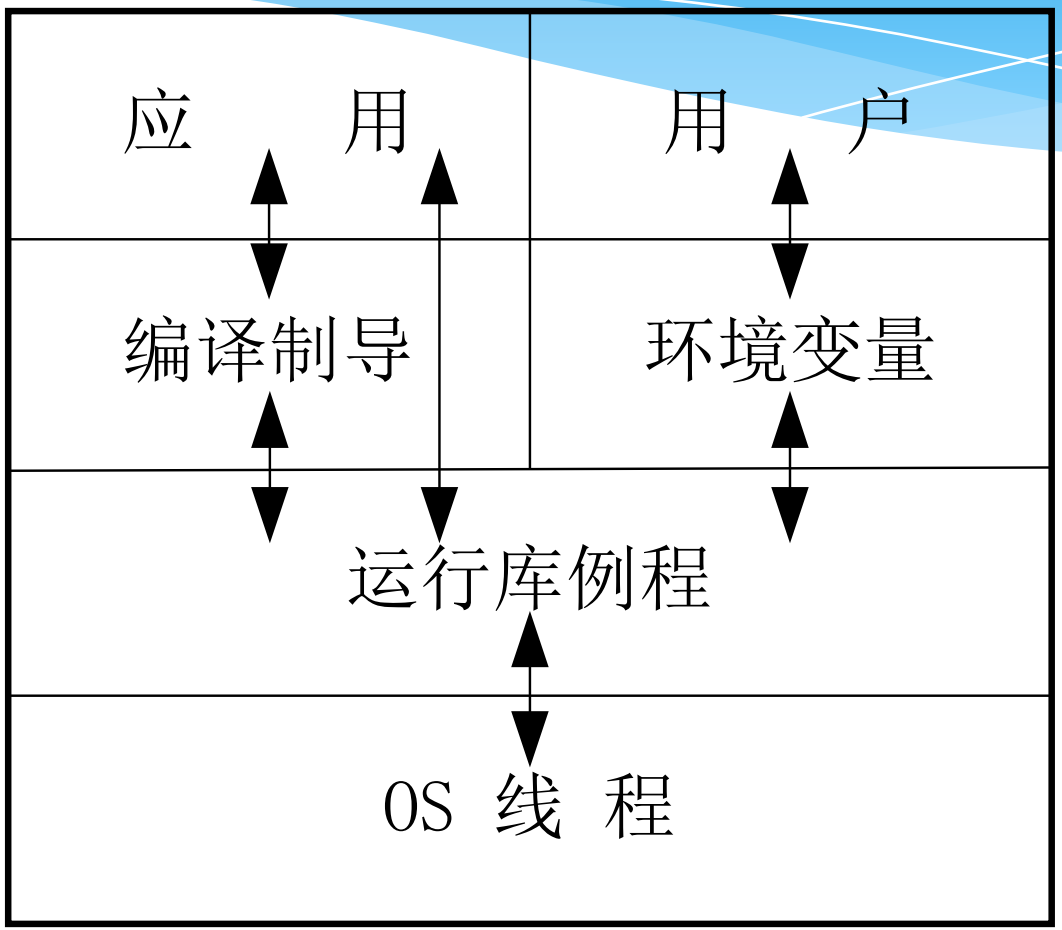
高性能计算与云计算课程实验

授课老师：何克晶 副教授

OpenMP编程简介

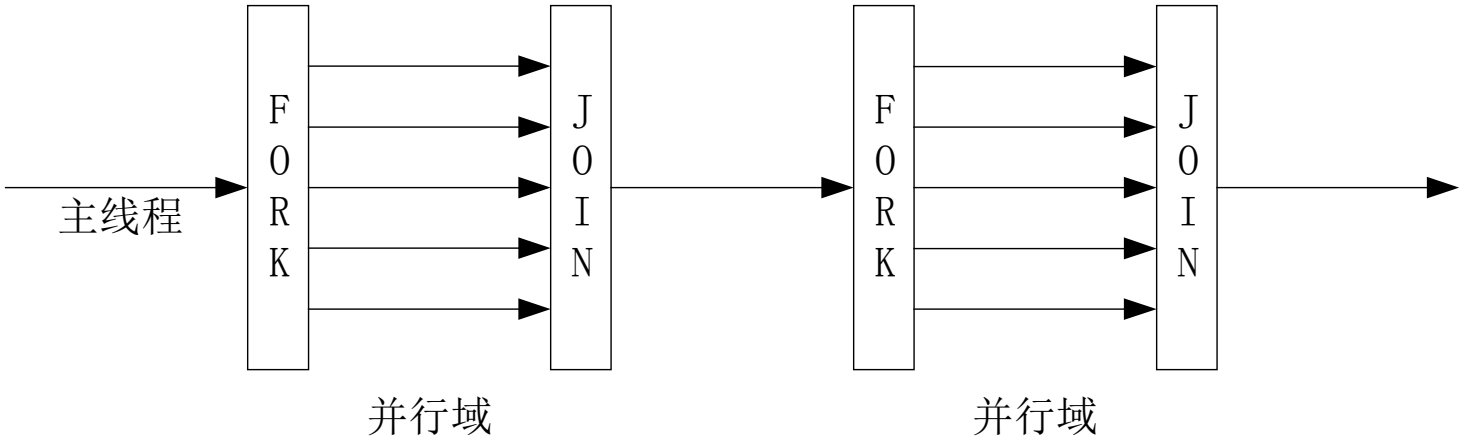
* 诞生于1997年，目前已经推出OpenMP 3.1版本。
* 标准版本3.0，2008年5月，支持Fortran/C/C++。
* 面向共享内存以及分布式共享内存的多处理器多线程并行编程语言。
* 一种编译指导语句，能够显式指导多线程、共享内存并行的应用程序编程接口（API）
* 具有良好的可移植性，支持多种编程语言。
* 支持多种平台
  * 大多数的类UNIX系统以及Windows NT系统（Windows 2000，Windows XP，Windows Vista等）

OpenMP体系结构



OpenMP并行模式

* Fork-Join



设置并行线程数目

* 1.环境变量OMP_NUM_THREADS

如Linux下可以直接在命令行export OMP_NUM_THREADS = 4

* 2.库函数omp_set_num_threads(int t)

* 3.默认为系统逻辑cpu数目

OpenMP编译制导

#pragma omp	directive-name	[clause, ...]	newline
制导指令前缀。对所有的OpenMP语句都需要这样的前缀。	OpenMP制导指令。在制导指令前缀和子句之间必须有一个正确的OpenMP制导指令。	子句。在没有其它约束条件下，子句可以无序，也可以任意的选择。这一部分也可以没有。	换行符。表明这条制导语句的终止。

OpenMP并行for语句

* for语句指定紧随它的循环语句必须由线程组并行执行;
* 语句格式
  * #pragma omp for [clause[[,]clause]…] newline
  * [clause]=
    * Schedule(type [,chunk])
    * ordered
    * private (list)
    * firstprivate (list)
    * lastprivate (list)
    * shared (list)
    * reduction (operator: list)
    * nowait

* 注意！！

  数据相关性和循环依赖性

  循环语句块应该是单出口与单入口的，在循环过程中不能使用break、goto和return语句。

  可以使用continue语句，因为这个语句不影响循环执行的次数。

OpenMP简单实例

```
int i;int j
#pragma omp parallel for private(j)
for（i=0; i<2; i++）
    for（j=6; j<10; j++）
        printf（ “i=%d j=%d\n”,
                    i, j）;
```

执行结果:

```
i=0 j=6
i=1 j=6
i=0 j=7
i=1 j=7
i=0 j=8
i=1 j=8
i=1 j=9
i=0 j=9
```

```
int i;int j;
for（i=0; i<2; i++）
#pragma omp parallel for
   for（j=6; j<10; j++）
        printf（"i=%d j=%d\n",
                    i, j）;
```

执行结果:

```
i=0 j=6
i=0 j=8
i=0 j=9
i=0 j=7
i=1 j=6
i=1 j=8
i=1 j=7
i=1 j=9
```

OPENMP计算PI

- 使用并行域并行化的程序

```
#include <omp.h>
static long num_steps = 100000;
double step;
#define NUM_THREADS 2
void main ()
{       int i;
        double x, pi, sum[NUM_THREADS];
        step = 1.0/(double) num_steps;
        omp_set_num_threads(NUM_THREADS)
       #pragma omp parallel
        {
                double x;
                int id;
                id = omp_get_thraead_num();
                for (i=id, sum[id]=0.0;i< num_steps; i=i+NUM_THREADS){
                                x = (i+0.5)*step;
                                sum[id] += 4.0/(1.0+x*x);
                }
        }
        for(i=0, pi=0.0;i<NUM_THREADS;i++)pi += sum[i] * step;
}
```

Thanks!