第 8 讲　　共享存储编程

1. 试分析下列循环嵌套中各语句间的相关关系：

```
①DO    I=1,N
       DO    J=2,N
S1:        A(I,J)=A(I,J-1)+B(I,J)
S2;        C(I,J)=A(I,J)+D(I+1,J)
S3         D(I,J)=0.1
       Enddo
   Enddo
②DO I= 1,N
S1:        A(I)=B(I)
S2:        C(I)=A(I)+B(I)
S3:        D(I) =C(I+1)
Enddo
③DO I=1,N
       DO J=2,N
S1:        A(I,J) = B(I,J) + C(I,J)
S2:        C(I, J)=D(I,J)/2
S3:        E(I,J)=A(I,J-1) **2+E(I,J-1)
       Enddo
   Enddo
```

2. 令 N=10^5 和 N=10^8，试编写计算π的 SPMD 并行程序，并在您现有的共享存储的平台上调试之;同时应执行在 1，2，3,4,5，6,7 和 8 个处理器上。
3. 试用 X3H5 模型，写出计算π的并行程序。
4. 下面是使用 Pthreads 方法计算π的一种并行代码段:

```
//* 计算 π Pthreads 编程代码段 *//
#include <stdio.h>
#include <pthread.h>      /* 线程控制所需的头文件 */
#include <synch.h>        /* 同步操作所需的头文件 */
extern unsigned * micro-timer;      /* 系统计时变量 */
unsigned int   fin,start;
semaphore_t semaphore;
barrier_spin_t barrier;       /* 同步信号量说明 */
double delta,pi;
typedef struct{      /* 定义参数结构 */
    int low;
    int high;
}Arg;
/* 定义线程执行的函数 */
void child(arg)
Arg * arg
{
    int low=arg→low;
    int high=arg→high;
    int i;
    double x ,part=0.0;
    for(i=low;i<=high;i++){
        x=(i+0.5) * delta;
        part+=1.0/(1.0+x * x);
    }
    psema(&semaphore);/* 利用信号量进行互斥累加 */
        pi+=4.0 * delta * part;
    vsema(&semaphore);
    -barrier_spin(&barrier);/* 线程在完成计算后需 barrier 同步 */
    pthread_exit();/* 线程终止 */
}
main(argc,argv)
int argc;
char * argv[];
{
    int no_of_threads,segments,i;
    pthread_t thread;
    Arg * arg;
```

```
if (argc != 3){
    printf("usage:pi<no-of-threads><no-of-strips> \ n");
    exit(1);
}
no-of-threads = atoi(argv[1]);
segments = atoi(argv[2];
delta = 1.0/segments;
pi = 0.0;
sema_init(&semaphore,1);      /* 初始化同步变量 */
-barrier_spin_init(&barrier,no_of_threads + 1);
start = * micro_timer;        /* 线程开始计时 */
for(i = 0;i<no_of_threads;i + + ){      /* 启动线程 */
    arg = (Arg * )malloc(size of (Arg));
    arg→low = i * segments/no_of_threads;
    arg→high = (i + 1) * segments/no_of_threads - 1;
    pthread_create(&thread,pthread_attr_default,child,arg);
}
-barrier_spin(&barrier);      /* 主进程等待所有子线程结束 */
fin = * micro_timer;          /* 线程结束记时 */
printf("%u \ n",fin - start);
printf(" \ npit \ t%15.14f \ n",pi);
}
```

①试解释上述代码段的工作流程。

②通过三种模型 X3H5、Pthreads 和 OpenMP 上计算 π 的代码段，比较它们的编程风格的异同和优缺点。

5、下面是使用经理员工模型（即主从模型）求解 N-皇后问题的并行代码段：

```
//* 求解 N-皇后经理-员工编程代码段 * //
/* Manager 程序段 */
if (!iam){/* 如果我是节点 0 */
```

```
        printf("\n STARTING…\n");
        while(get_board(board)=!DONE){
            crecv(READY,NULL,0);
            nodenbr=infonode();
            msgcount++;/*计数多少节点准备好*/
            csend(TASK,board,sizeof(two ,two D),nodenbr,0);
            msgcount--;
        }
    /*等待所有员工空闲*/
        while(msgcount!=nodes-1){
            crecv(READY,NULL,0);
            msgcount++;
        }
        /*发送 FINISHED 消息给所有节点,并退出*/
        board[0][0]=FINISHED;
        csend(TASK,board,sizeof(two D),-1,0);
        goodbye();
    }
    else{/*员工程序段*/
        for(;;){
            csend(READY,0,0,0,0);
            crecv(TASK,board,sizeof(board));
            if(board[0][0]==FINISHED
                goodbye();
            if(chk-board(board))
                move_to_right(board,0,MCOLS);
        }
    }
```

试解释上述代码段的计算过程

6．下面适用于集群计算系统中，分布式内存环境下编程的是？

A. Java Thread　　B. Posix Thread

C. MPI　　D. OpenMP

7．DNS 并行矩阵算法的输入是两个$n \times n$的矩阵，算法输出为$n \times n$的矩阵。当$p = n^3$时，该算法的时间复杂度为：

A. $O(\log n)$　　B. $O(p \log n)$　　C. $O(n)$　　D. $O(n^3 / p)$

8．什么是OpenMP的编程模型、体系结构、控制结构和数据域子句？

9. 简述 OpenMP 和多线程编程的区别和优缺点。

10. 计算圆周率π的串行代码段如下：

```
h=1.0/n;
sum =0.0;
for (i=0;i<n;i++) {
    x=h*(i+0.5);
    sum=sum+4.0/(1+x*x);
    }
pi=h*sum;
```

用 OpenMP 编程语言，写出计算π的并行代码段。