软件学院本科生 **2019—2020** 学年第二学期《并行与分布式程序设计》 课程期末考试试卷（**B** 卷）

专业： **年级：** **学号：** **姓名：** **成绩:**

草 稿 区

得 分

一、单选及填空（本题共 25 分，选择 1-11 题每题 2 分，12、13 题每空 1 分，请把答案按题号汇总到表格中）

1	2	3	4	5	6	7	8	9	10	11	12(1)	12(2)	13

1. 2018 年 7 月 22 日，我国“天河三号 E 级原型机系统”已在国家超级计算天津中心完成研制部署，并顺利通过项目课题验收，将逐步进入开放应用阶段。预示着中国 E 级计算机将很快进入实质性研发阶段。E 级计算机是指峰值计算性能在每秒（ ）的超级计算机。

   A. 一亿亿次 B. 十亿亿次 C. 百亿亿次 D. 千亿亿次

2. 一个 SSE 寄存器可容纳____个短整型数。

   A.2 B.4 C.8 D.16

3. 在下面问题中，SIMD 并行最不适合（ ）。

   A. 向量加法 B. 向量中元素排序 C. 矩阵向量乘法 D. 矩阵加法

4. n*n 的两个矩阵相乘,问题规模为（ ）。

   A.n B.n^2 C.2n^2 D.n^3

5. 一个函数是“线程安全的”，其含义是该函数（ ）。

   A.多线程执行能抵御网络攻击 B.多线程执行能保护用户隐私数据

   C.多线程执行结果也是正确的 D.以上皆错

6. 主线程创建了 4 个从线程，对它们执行 pthread_join,然后打印一条信息，从线程打印各自的线程号，未使用任何同步，则主线程打印的消息和从线程打印的线程号的相对顺序（ ）。

   A.必然主线程前、从线程后 B.必然从线程前、主线程后

   C.必然相互交织 D.各种顺序皆有可能

7. OpenMP 最常见的编程方式是寻找串行程序中（ ）结构进行并行化。

   A.赋值语句 B.循环语句 C.递归 D.输出语句

草 稿 区

8. 在 OpenMP 编程的循环调度中，假设有 15 个迭代任务和 3 个线程，若使用 stastic 调度类型，那么缺省的一次分配给一个线程的任务块大小为（ ）。

   A. 5　　B. 3　　C. 1　　D. 指数级减小

9. OpenMP 程序和 Pthread 程序中线程获得自身编号的方式分别是____。

   A.两者均为创建线程时传递参数　　B.两者均通过特定 API

   C.前者创建线程时传递参数，后者通过特定 API　　D.前者通过特定 API，后者创建线程时传递参数

10. MPI 程序中发送和接收消息的两个进程必须（ ）。

    A.在同一个网段中 B.连接在同一个路由器上 C.在同一个通信域中 D.以上皆错

11. for (i=0; i<10; i++) A[i] = A[i]+1; 此循环（ ）数据依赖。

    A.存在　　B.不存在　　C.不确定　　D.以上皆错

12. 若串行起泡排序算法时间 150s，串行快速排序算法 40s，一个 4 核并行起泡排序算法用时 50s，则该并行起泡排序算法的加速比是(1)_____，效率是(2)_____。

13. 一个环形互联网络的等分宽度是_____。

得 分

二、多项选择题（本题共 15 分，每小题 3 分，请把本题答案按题号汇总到表格中）

1	2	3	4	5

1. 下列关于“冯•诺伊曼瓶颈”的表述正确的是（ ）。

   A. 与 CPU 和内存分开设计有关　　B. CPU 计算能力发展迅速，CPU 与高速缓存间数据传输成为瓶颈

   C. 与 CPU 计算能力不够有关　　D. CPU 计算能力发展迅速，CPU 与内存数据传输成为瓶颈

2. SIMD 向量计算编程中，主要会遇到那些额外开销 （ ）

   A. 打包/解包开销　　B. 数据传输开销　　C. 对齐开销　　D. 控制流导致额外开销

3. 动态任务划分相对于静态任务划分的优点是（ ）。

   A.确保负载均衡　　B.任务粒度细　　C.计算复杂度低　　D.并行效率高

4. 关于互斥量 mutex,下面说法正确的是（ ）。

   A.它将保护区域内的运算变成原子操作 B.任何时刻只允许一个线程进入保护区域

   C.保险期间，对并发操作都应用互斥量保护 D.互斥量加锁、解锁开销远大于一般运算

5. 下面做法中，人们普遍认为可以提高并行程序的运行效率的有（ ）。

   A. 减少内存中数据的访问次数 B. 减少线程的创建、销毁操作

   C. 尽量均衡地分配任务　　D. 尽可能多的增加线程数量

得 分

三、判断题（本题共 10 分，每小题 1 分，请把本题答案按题号汇总到表格中）

1	2	3	4	5	6	7	8	9	10

1. 截止到 2019 年末，Top 500 的超算机中中国是拥有超算机数量最多的国家。（ ）
2. 并行计算领域，近些年硬件技术飞速发展，软件生态环境几乎停滞；中国团队至今尚未获得过戈登•贝尔奖。（ ）
3. 指令级并行让多个处理器部件或功能单元串行执行指令来提高计算性能；主要实现方式有流水线和多发射。（ ）
4. 单指令多数据流（SIMD）的一种广泛应用是向量处理器，此外，GPU 也使用 SIMD 并行方式。（ ）
5. 多指令多数据流（MIMD）系统主要包括共享内存系统和分布式内存系统。（ ）
6. SSE intrinsics _mm_load_pd 命令的功能是“对齐向量读取双精度浮点数”。（ ）
7. 当问题规模不变时，随着处理器数量增大，加速比必然增大。（ ）
8. MPI 编程中支持自定义数据类型。（ ）
9. OpenMP 是分布式内存架构下的一种编程工具。（ ）
10. OpenMP 编译指示的作用范围是其后一个语句。（ ）

草 稿 区

得 分

四、简答题（本题共 20 分）

1. 与串行程序设计相比，并行程序设计的复杂性主要体现在哪些方面？（4 分）

草　稿　区

2. 请对下列并行算法设计中的名词进行解释。(6 分)

(1)竞争条件：

(2)临界区：

(3)数据依赖：

3. 在 OpenMP 编程中，简述 for 循环需要的条件。 (4 分)

4.MPI 编程中提供了多种组通信的原语。请写出不少于 6 个组通信原语，并分别简述其功能。(6 分)

得 分

五、按要求写出相应代码（本题共 30 分）

草 稿 区

1. 补全下面程序代码，程序实现的是 SSE 版本的矩阵乘法, c=a*b。（10 分）

①:

②:

③:

④:

⑤:

```
#include <stdio.h>

#include <pmmintrin.h>
#include <stdlib.h>
#include <algorithm>

const int maxN =1024;                    // magnitude of matrix

int n;
float a[maxN][maxN];
float b[maxN][maxN];
float c[maxN][maxN];

void sse_mul(int n, float a[][maxN], float b[][maxN], float c[][maxN]){
    __m128 t1, t2, sum;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < i; ++j)
            ①
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j){
            c[i][j] = 0.0;
            sum = _mm_setzero_ps();
            for (int k = n - 4; k >= 0; k -= 4){          // mul and sum every 4 elements
                ②
                t2 = _mm_loadu_ps(b[j] + k);
                t1 = _mm_mul_ps(t1, t2);
                ③
            }
            sum = _mm_hadd_ps(sum, sum);
            ④
            _mm_store_ss(c[i] + j, sum);
                for (int k = (n % 4) - 1; k >= 0; --k){        // handle the last n%4 elements
                ⑤
            }
        }
    }
    for (int i = 0; i < n; ++i) for (int j = 0; j < i; ++j)  ①
}
```

2. 补全下面程序代码，程序是用 Pthread 实现子线程的线程号等字符串的输出。（10 分）

草 稿 区

①:

②:

③:

④:

⑤:

```
#include <stdio.h>
#include <stdlib.h>
#__①__

/* Global variable: accessible to all threads */
int thread_count;
void* Hello(void* rank); /* Thread function */

int main(int argc, char* argv[]) {
   long thread; /* Use long in case of a 64-bit system */
   pthread_t* thread_handles;

   /* Get number of threads from command line */
   thread_count = strtol(argv[1], NULL, 10);
   thread_handles = malloc(thread_count*sizeof(pthread_t));
   for (__②__)
      pthread_create(__③__);
   printf("Hello from the main thread\n");
   for (thread = 0; thread < thread_count; thread++)
      __④__
   free(thread_handles);
   return 0;
} /* main */
void* Hello(void* rank) {
   long my_rank = (long) rank;    /* Use long in case of 64-bit system */
   pthread_mutex_lock(&mutex);
   printf("Hello from thread %ld of %d\n", my_rank, thread_count);
   __⑤__
   return NULL;
} /* Hello */
```

3. 补全下面程序代码，程序是用 OpenMP 实现子线程的线程号等字符串的输出。（6 分）

```
#include <stdio.h>
#include <stdlib.h>
#__①__
#include <omp.h>
#endif

void Hello(void); /* Thread function */

int main(int argc, char* argv[]) {
   /* Get number of threads from command line */
   int thread_count = strtol(argv[1], NULL, 10);

# pragma omp __②__
   Hello();

   return 0;
}   /* main */
void Hello(void) {
   #__①__
   int my_rank = omp_get_thread_num();
   int thread_count = __③__
   # else
      int my_rank = 0;
      int thread_count = 1;
   # endif
   printf("Hello from thread %d of %d\n", my_rank, thread_count);

}   /* Hello */
```

①:

②:

③:

4. 补全下面程序代码，程序是用 MPI 编程实现不同进程的进程号等字符串的输出。（4 分）

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char *argv[]){
   int myid, numprocs;
   int namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];
   MPI_Init(&argc,&argv);
   __①__    \\查询进程号赋值给 myid
   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
   MPI_Get_processor_name(processor_name,&namelen);
   fprintf(stderr,"Hello World! Process %d of %d on %s\n",myid, numprocs, processor_name);
   __②__    \\结束 MPI 程序
  return 0;
}
```

①:

②:

草 稿 区