草 稿 区

软件学院本科生 2020—2021 学年第 二 学期《并行与分布式程序设计》课程期末考试试卷（A 卷）

专业： 年级： 学号： 姓名： 成绩:

得 分

一、单选及填空（本题共 28 分，每空 2 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	13(1)	13(2)

1. 2018 年 7 月 22 日，完全由我国自主研发建造的“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，该计算机正式发布后有可能助力中国超算重新登顶 Top500 榜单。

   A. 神威・太湖之光　　B. 天河三号　　C. 天河二号　　D. 富岳

2. 一个 SSE 寄存器可容纳（ ）个短整型数。

   A.2　　B.4　　C.8　　D.16

3. SSE intrinsics _mm_store_ss 指令的功用是（ ）。

   A.对齐向量保存单精度浮点数　　B.未对齐向量保存单精度浮点数

   C.对齐向量保存双精度浮点数　　D.未对齐向量保存双精度浮点数

4. 一个函数或库是“线程安全的”，意味着其（ ）。

   A. 多线程执行结果能正确执行　　B.多线程执行能保护用户隐私数据

   C. 多线程执行能抵御网络攻击　　D.多线程执行存在竞争条件

5. pthread_create 函数中设置一个参数为"线程函数参数"的缘由是（ ）。

   A、它调用线程函数时可直接传递，功能更优　　B、它需求预处理之后传递给线程函数

   C、线程函数不是用户程序调用，只能选用这种方法由系统代为传递参数

   D、没有必要设置该参数

6. 代码：“for (i=0; i<10; i++)　A[i] = A[i]+1; B[i+1]=A[i+1]-1”此循环（ ）数据依赖。

   A.存在　　B.不存在　　C.不确定　　D.以上皆错

草 稿 区

7. 编译器编译 OpenMP 并行循环时，会自动生成一些代码，其中不包括（ ）。

A.创建和管理线程代码　　B.循环划分给线程的代码

C.找出数据依赖的代码　　D.线程同步的代码

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 dynamic 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. 静态任务划分相对于动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

10. 选用 MPI 主从模型处理矩阵每行排序问题，主进程不断向每个从进程发送使命、接收结果，则它从从进程接收结果考虑，以下哪种方法更好（ ）。

A、按编号次序顺次从从进程接收结果　　B、按编号逆序顺次从从进程接收结果

C、按编号次序，顺、逆序交替接收结果　　D、使用 MPI_ANY_SOURCE 和 MPI_ANY_TAG

11. MPI 标准点对点通信模式是（ ）。

A、阻塞的　　B、非阻塞的　　C、对等的　　D、主从的

12. 某串行程序运行时间为 20s，可并行化比例为 90%，若用 p 个核将其并行化，其加速比的理论上限是___。

13. 若串行快速排序算法 40s，串行起泡排序算法时间 200s，一个 5 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)_____

二、多项选择题（本题共 10 分，每小题 2 分，请把本题答案按题号汇总到表格中）

得 分

1	2	3	4	5

1. 以下是推动并行计算发展的因素的有（ ）。

A.单核 CPU 发展中功耗是一个瓶颈　　B.单 CPU 发展已能满足应用需求

C.多核、众核具有巨大的功耗优势　　D.各种应用、软件的强烈需求

2. 下面属于混合编程模式的是（ ）。

A. MPI_THREAD_SINGLE　　B. MPI_THREAD_FUNNELED

C. MPI_THREAD_SERIALIZED　　D. MPI_THREAD_MULTIPLE

3. 关于 OpenMP 循环并行程序的编写，下列说法中正确的是（ ）。

A、程序员无需编写线程创立和管理代码　　B、程序员无需关注循环中的数据依赖问题

C、程序员需指出哪个循环应并行　　D、程序员可以不显示编写线程同步代码

4. Pthread 编程中可以用来实现路障的方式有（ ）。

A.忙等待　　B.条件变量　　C. 信号量　　D.读写锁

5. 以下选项关于阿姆达尔定律描述正确的是（ ）。

A.只要一个串行程序中存在不可并行化的部分，它通过并行化的加速比就是受限的

B.串行程序中如果有比例为 r 的部分不可并行化，则根据该定律，加速比就是 1/r

C.该定律描述了一种加速比存在上限的情况，现实中并行化往往达不到该上限

D.只要并行的核数目足够多、性能足够强，并行化加速比是可以无限增大的

草　稿　区

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. MPI 编程非阻塞通信中支持查看通信状态的有 MPI_Wait()、MPI_Test()等原语。（ ）
2. 尽可能多的增加线程数量可以提高并行程序的运行效率。（ ）
3. “冯•诺伊曼瓶颈”与 CPU 计算能力远不如存储读写能力有关。（ ）
4. 控制流语句进行 SIMD 并行化很困难的原因是控制流语句可能导致同一数据执行不同指令。（ ）
5. 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。（ ）
6. 在 OpenMP 编程中，能进行 parallel for 循环并行化的代码循环变量一般必须是带符号整数或指针。（ ）
7. MPI 编程中支持自定义的归约操作，但不支持自定义通信中的数据类型。（ ）
8. 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。（ ）
9. 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必然相互交织。（ ）
10. OpenMP 中指定多线程中只要主线程运行代码块的指令是“omp single”。（ ）

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 循环并行化需要的条件。 (4 分)

4. MPI 编程是一个消息传递接口函数库，提供了基础的点对点的通信接口，如 MPI_Send、MPI_Recv，此外还提供了多种组通信的方式。请写出不少于 6 个组通信原语，并分别简述其功能。（6 分）

MPI_Bcast　MPI_AllReduce
MPI_Reduce　MPI_Reduce_scatter
MPI_Gather
MPI_Scatter

得 分

五、按要求写出相应代码（本题共 32 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
                t1 = _mm_loadu_ps(a[i] + k);
                __②__;
                __③__;
                sum = _mm_add_ps(sum, t1);
            }
            __④__;
            sum = _mm_hadd_ps(sum, sum);
            _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

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <pthread.h>

using namespace std;

typedef struct{
      int    threadId;
} threadParm_t;

const int ARR_NUM = 10000;
const int ARR_LEN = 10000;
const int THREAD_NUM = 4;
const int seg = ARR_NUM / THREAD_NUM;

vector<int> arr[ARR_NUM];
pthread_mutex_t      mutex;

void init(void)
{
   for (int i = 0; i < ARR_NUM; i++) {
      arr[i].resize(ARR_LEN);
      for (int j = 0; j < ARR_LEN; j++)
         arr[i][j] = rand();
   }
}
void *arr_sort(void *parm)
{
   threadParm_t *p = (threadParm_t *) parm;
   int r = p->threadId;
   for (int i = r * seg; i < (r + 1) * seg; i++)
      sort(arr[i].begin(), arr[i].end());
   ___①___;
   printf("Thread %d.", r);
   pthread_mutex_unlock(___②___);
}
int main(int argc, char *argv[])
{
   init();
   mutex = PTHREAD_MUTEX_INITIALIZER;
   pthread_t thread[THREAD_NUM];
   threadParm_t threadParm[THREAD_NUM];
   for (___③___)
   {
      threadParm[i].threadId = i;
      pthread_create(___④___);
   }
   for (int i = 0; i < THREAD_NUM; i++)
   {
      ___⑤___
   }
   pthread_mutex_destroy(&mutex);
}
```

①:

②:

③:

④:

⑤:

草 稿 区

草 稿 区

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>
#ifdef _OPENMP
#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 ____①____
   Hello();

   return 0;
}   /* main */
void Hello(void) {
   #____①____
   int my_rank = ____②____;
   int thread_count = omp_get_num_threads();
   # 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 2 个进程的并行排序的代码。（6 分）

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char ** argv)
{
   int rank, a[1000], b[500];

   MPI_Init(&argc, &argv);
   ____①____;
   if (rank == 0) {
      ____②____;
      sort(a, 500);
      ____③____;
   /* Serial: Merge array b and sorted part of array a */
   }
   else if (rank == 1) {
      MPI_Recv(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      sort(b, 500);
      MPI_Send(b, 500, MPI_INT, 0, 0, MPI_COMM_WORLD);
   }
   MPI_Finalize();
   return 0;
}
```

① :

②:

③: