|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 专业 年级 教 学 班 号 ( 课 序 号 ) 学号    姓名 座位号  —— ·——经—— ·  专业 年级 教 学 班 号 ( 课 序 号 ) 学号  专业  —  封 — —  — — 战 —  —  — 线 —  姓名 座位号 | — 密 | 《并行处理及分布式系统》考试试卷(第一套) 课程号 5619003030 考 试 时 间 100 分钟  适用专业年级(方向):计算机科学与技术2019级 考试方式及要求： 闭卷笔试   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 题 号 |  | 二 | 三 | 四 | 总分 | | 得 分 |  |  |  |  |  | | 阅卷人 |  |  |  |  |  | |

**一、选择题(每小题2分，共40分)**

1、当前CPU 性能提升已从依赖时钟频率提升转为更多依赖什么()

A、多核和众核技术 B、 突破物理局限 C、 改善散热 D 、采用新材料

2、当前并行软件面临的主要挑战不包括()

A、能 耗 B、伸缩性 C、研发周期 D、可靠性

3、静态线程编程模式的缺点是()

A、线程管理开销高 B、系统资源利用率低 C、线程负载不均 D、线程通信开销高

4、研究并行算法的可扩展性的目的是()

① 确定某类问题的何种并行算法与何种并行系统的组合，可有效的利用系统大量处 理机；②算法在小规模并行机上的运行性能来预测其移植到大规模并行机上后的运 行性能；③对一固定规模的问题，能确定其运行在某类并行系统上时，所需的最多 处理机台数和获得的最大加速比；④指导并行算法和并行机体系结构的改进。

A、①②③ B、②③④ C、①②③④ D、①③④

5 、n 个数求和的串行程序，通过一个循环将每个数累加到全局变量sum 中，其多 线程版本简单的将循环范围改变为每个线程负责的范围，存在的问题是()

A、负载不均 B、通信开销大

C 、CPU空闲等待严重 D 、sum 累加产生竞争条件，导致结果错误

6、对一个串行程序进行SIMD 并行化，可以优化的程序部分是()

A、声明语句 B、条件分支语句 C 、循环语句 D 、输入输出语句

7、在使用完互斥量之后最后一步是必须对其进行什么()

A、初始化 B、加 锁 C、解 锁 D、销 毁

8、动态线程编程模式的缺点是什么()

A、线程管理开销高 B、系统资源利用率低 C、线程任务分配困难 D、线程通信效率低

9、在分布式内存架构编程中，进程间不能()

A、进行通信 B、进行同步

C、发送和接收消息 D、通过读写变量交换数据

10、采 用MPI 主从模型解决矩阵每行排序问题，主进程每次向一个从进程发送10

行作为一个任务相对于每次发送1行的优点是什么()

A、更有利于负载均衡 B、减少了通信开销

C、降低了计算次数 D、减少了从进程空闲

11、两个矩阵相乘，若矩阵规模大于cache 大小，则优化访存的可行方法是()

|  |
| --- |
| A、先将两个矩阵读入 cache再进行乘法 B、先转置第一个矩阵再进行乘法  C、先转置第二个矩阵再进行乘法 |
| D、以上皆错 |  |

12 、 ,执行 pthread\_sem\_post 操作，当前线程会唤醒阻塞线程。()

A、当信号量已加锁时 B、当信号量为0时

C、当信号量已超过阈值时 D、当信号量已销毁时 13、关于路障(同步)机制，下面说法错误的是()

A、会导致速度快的线程阻塞，不应使用B 、在需要强制线程步调一致时，应使用

C、可用互斥量机制实现 D、属于一种组通信

14、主线程通过 函数获取特定线程的返回结果。()

A 、pthread\_create B 、pthread\_join C 、pthread\_cancel D 、pthread\_get 15、在使用条件变量之前必须对其进行 ()

A、初始化 B、加 锁 C、解锁 D、销 毁

16 、for(i=2;i<10;i++)A[i]=A[i-2]+1; 关于此循环说法正确的是。() A、存在数据依赖 B、不存在数据依赖

C、不确定是否数据依赖 D、以上皆错

17、OpenMP循环并行指令是()

A 、omp parallel B 、omp single C 、omp parallel for D 、omp master

18、为防止编译器不支持OpenMP, 应使用 实现OpenMP 代码和普通代码的

条件编译。()

A 、"#include" B 、"#pragma omp parallel"

C 、"#ifdef\_OPENMP" D 、"#define\_OPENMP"

19、实现任何时刻都只有一个线程进行共享变量更新的OpenMP 指 令 是 ( )

A 、omp parallel B 、omp barrier

C 、ompcritical D 、omp reduce

20 、OpenMP主要是通过 来更高层抽象的多线程编程。() A、设计实现新的语言 B、为现有语言提供大量库函数

C、为现有语言扩展编译指示 D、新编译器实现自动并行化 二 、**简答题(共5小题，每小题6分共30分**)

1、完成下面两个问题。

(1)在并行计算任务处理过程中，广泛采用哪两种方法?

(2)假如P 教授进行“英国文学调查”的授课，她有100个学生x 还有4个助教。om

学期结束的时候，要进行一次期末测试，试卷包括5道题 ，为了给学生打分，P 教 授和他的助教进行试卷批改，可以采用什么方法，并对方法进行阐述。

2、在冯 ·诺依曼系统中加入缓存的虚拟内存改变了它作为SISD 系统的类型吗?如

果加入流水线呢?多发射或硬件多线程呢?

3、假 设 comm\_sz=4,x 是一个拥有 n=14 个元素的向量。分别完成如下任务分配， 分配方法应该通用性足够强，无论 comm\_sz 和 n 取何值都能分配。你应该使你 的分配“公正”,如果q 和 r 是任意的2个进程，分给q 和 r 的 x 分量个数的差 应尽可能小。

(1)如何用块划分法在一个程序的进程间分发x 的元素。

(2)如何用循环划分法在一个程序的进程间分发x 的的元素。

(3)如何用b=2 大小的块用块-循环划分法在一个程序的进程间分发x 的值。

4、链表操作 Insert 和 Delete都由两个不同的阶段组成。在第一个阶段，这两个操 作要么查找新结点的位置，要么查找要删除结点的位置。在第一个阶段的输出结果 确定后，在第二个阶段要么插入一个新结点，要么删除一个存在的结点。其实，对 链表来说，把这种类型的操作分成两个函数调用是十分常见的。对于这两个操作， 第一阶段都只涉及对链表的读访问，只有第二阶段才修改链表。如果在第一阶段使 用一个读锁来锁链表是否安全?在第二阶段用写锁来锁链表是否安全?请解释你 的答案。

5 、OpenMP 为归约变量创建私有变量，这些私有变量的值按照归约操作符的类型 初始化。假设私有变量占一个byte,例如，如果归约操作符是加法，那么私有变量 初始化为0,当操作符分别为\*(乘法)、&&(逻辑与)、 || (逻辑或)、&(位与)、 | (位或)、^(异或)时，私有变量初始化为什么?

三、程序填空题(1空2分、共10分)

下面代码是mpi\_output.c, 该程序实现按进程号的顺序打印，即，进程0先输 出，然后进程1,以此类推可以将消息发送到进程0进行打印，根据注释完成程序 填空。

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

const int MAX\_STRING=100;

int

main(void){ nt src;

char msg[MAX\_STRING];

int my\_rank,comm\_sz;// 定义进程编号和进程总数变量 MPI\_Comm comm

//MPI 初始化

(1)

comm=MPI\_COMM\_WORLD; //获取进程数量

年级 教 学 班 号 ( 课 序 号 ) 学号  姓名 座位号

(2) //获取进程编号

(3) if(my\_rank=0){

printf("Proc%d of%d \n",my\_rank,comm\_sz); for(src=1;src<comm\_sz;src++){

受,~~gk;~~

ese(

sprintf(msg,"Proc%d of%d \n",my\_rank,comm\_sz);

/发送消息

(5)

MPI\_Finalize(; return 0;

}/\*main\*/

**四** **、编程实现(2小题，共20分)**

1、(10分)估算π值的例子很多，其中最简单的是：



,请使用忙等待实现计算π的Pthreads程序，将π的最终结果存入全局变量sum 中。 void Thread\_sum(void \*rank){

—



1. (10分)使用OpenMP 实现奇偶交换排序算法。 voidOdd\_even(int a[],intn){

— 线 —



—

附录：常用的MPIAPI 接 口

intMPI\_Send(void\*buf,int count,MPI\_Datatype datatype,intdest,int tag,MPI\_Comm comm) intMPI\_Reev(void\*buf,int count,MPI\_Datatype datatype,int source,int tag,MPI\_Comm comm,

MPI\_Status \*status);

nt MPI\_Sendreev(void \*sendbuf,intsendent,MPI\_Datatypesendtype,intdest,int sendtag. void\*recvbuf,intrecvent,MPI\_Datatype recvtype,int source,intrecvtag,

MPI\_Comm comm,MPI\_Status \*status);

intMPI\_Sendrecv\_replace(void\*buf,int count,MPI\_Datatype datatype,intdest,int sendtag,

int source,intrecvtag.MPI\_Commcomm,MPI\_Status \*status);

intMPI\_Bcast(void\*buffer,int count,MPI\_Datatypedatatype,int root,MPI\_Comm comm); intMPI\_Scatter(void \*sendbuf,intsendent,MPI\_Datatypesendtype,void \*recvbuf,

intrecvent,MPI\_Datatype recvtype,introot,MPI\_Comm comm);

intMPI\_Gather(void \*sendbuf,intsendent,MPI\_Datatypesendtype,void \*recvbuf,

intreevent,MPI\_Datatype reevtype,int root,MPI\_Comm comm);

intMPI\_Allgather(void \*sendbuf,intsendent,MPI\_Datatypesendtype,void\*recvbuf,

intrecvcnt,MPI\_Datatype recvtype,MPI\_Comm comm);

int MPI\_Reduce(void\*input\_data,void\*output\_data,int count,MPI\_Datatype datatype,

MPI\_Opoperator,intdest,MPI\_Comm comm);

常用的PthradsAPI 接 口

intpthread\_create(pthread\_t\*tidp,const pthread\_attr\_t \*attr,void\*(\*start\_rtn)(void\*),void\*arg); intpthread\_create(pthread\_t\*tidp,const pthread\_attr\_t \*atr,void\*(\*start\_rtn)(void\*),void \*arg); intpthread\_muex\_init(pthread\_mutex\_t \*restrict mutex,const pthread\_mutexattr\_t\*restrict attr);

nt pthread\_mutex\_lock(pthread\_mutex\_t\*mutex);

nt pthread\_mutex\_unlock(pthread\_mutex\_t\*mutex);

intpthread\_mutex\_destroy(pthread\_mutex\_t \*mutex);

!