1. 为什么需要并行计算？

贯穿整个计算机技术发展的核心目标是提高计算性能，而之前提高性能的手段是提高处理器字长、提高集成度、采用流水线等微体系结构技术实现指令级并行，提高处理器频率等等。

但是就目前而言这些手段都遇到了瓶颈1、芯片集成度已经接近极限，不可能无限制提高。2、处理器的指令级并行度提升接近极限，复杂技术得以研究应用，但难以挖掘出更多潜能。3、处理器速度和存储器速度的差异越来越大。 4、功耗和散热大幅增加超过芯片承受能力。

总而言之，单核处理器的性能提升已经接近极限，而应用领域的计算规模和复杂度仍在大幅提高，因此多核/众核并行计算技术的发展成为必然趋势。

1. 并行计算按照系统类型划分，可以分为哪几种？简述每一种系统类型的特点。
2. 多核/众核并行计算系统MC：指处理器核通过NOC（片上网络）集成在一个芯片上，共享LLC和主存，通常使用混合式内存访问机制（本地缓存加全局内存），功耗很低。是众多并行线程同时执行的计算系统，属于多处理器。
3. 对称多个处理系统SMP：使用独立的处理器和共享内存，以总线结构互联，运行一个操作系统，定制成本高，难以扩充，规模较小（2-8处理器）。相同类型CPU通过总线互联，并等同地位共享所有存储资源，即多个CPU对称工作。即一种UMA结构多处理器，PC、工作站和服务器等多采用SMP结构。
4. 大规模并行处理MPP：使用独立的处理器以及独立的内存、OS，专用的高速内联网络，难以升级和扩充，规模中等（TOP500中有80多个）。例如，可以通过专用互联网络（如Mesh、交叉开关）将数量达几百甚至几千个的SMP服务器连接成MPP，SMP服务器之间协同工作，以完成同一个任务。
5. 集群Cluster：使用商品化的刀片或机架服务器，以网络互联为一个物理上紧密的计算系统，可扩展性强，规模可小可大，是目前高性能并行计算最常见的形式（TOP500中有400多个）。以高速网卡将若干PC或SMP服务器或工作站连接而成的并行计算系统，其中每个节点有各自的独立编址的内存和磁盘，属于紧耦合同构多计算机系统（消息传递机制）。
6. 网格Grid：地理上广泛分布的异构计算资源构成的一个极为松散的计算系统，主要用于并行度很低的大规模科学计算任务。属于松耦合异构多计算机系统。
7. 并行计算按照并行程序设计方法分类，可以分为哪几种？简述每一种方法的特点。
8. 共享内存变量（Shared Memory Variables）：多线程共享存储器变量方式进行并行程序设计，会引起数据的不一致性，导致数据和资源访问冲突，需要引入同步控制机制。我们经常会设置一个共享变量，然后多个线程去操作同一个共享变量，从而达到线程通讯的目的。例如，我们使用多个线程去执行页面抓取任务，我们可以使用一个共享变量count来记录任务完成的数量。每当一个线程完成抓取任务，会在原来的count上执行加1操作。这样每个线程都可以通过获取这个count变量来获得当前任务的完成情况。当然必须要考虑的是共享变量的同步问题，这也是共享内存容易出错的原因所在。这种通讯模型中，不同的线程之间是没有直接联系的。都是通过共享变量这个“中间人”来进行交互。而这个“中间人”必要情况下还需被保护在临界区内（加锁或同步）。由此可见，一旦共享变量变得多起来，并且涉及到多种不同线程对象的交互，这种管理会变得非常复杂，极容易出现死锁等问题。
9. 消息传递方式（Message Passing）：对于分布式内存结构，为了分发数据和收集计算结果，需要在各个节点间进行数据通信，最常用的是消息。消息传递方式采取的是线程之间的直接通信，不同的线程之间通过显式的发送消息来达到交互目的。消息传递最有名的方式是actor模型，在这种模型下，一切都是actor，所有的actor之间的通信都必须通过传递消息才能达到。每个actor都有一个收件箱（消息队列）用来保存收到其他actor传递来的消息。actor自己也可以给自己发送消息。
10. MapReduce方式：Google公司提出的MapReduce并行程序设计模型，是当时最易于使用的并行程序设计方法，广泛使用于搜索引擎等大规模数据并行处理。MapReduce是Hadoop的组成部分，它是一个软件框架，基于该框架能够容易地编写应用程序，这些应用程序能够运行在由上千个商用机器组成的大集群上，并以一种可靠的，具有容错能力的方式并行地处理上TB级别的海量数据集。MapReduce的思想就是“分而治之”， Mapper负责“分”，即把复杂的任务分解为若干个“简单的任务”来处理，Reduce负责对map阶段的结果进行汇总。MapReduce适合数据复杂运算，不适合算法复杂的运算，不适合实时计算、流式计算、DAG有向图计算。即要求算法足够简单，数据量可以足够大。
11. MPI提供哪几种通信方式/接口？

通信方式：

1、点对点通信（Point-point communication）：同步通信功能（阻塞通信，等待通信操作完成后才返回）和异步通信功能（非阻塞通信，不等待通信操作完成即返回）

2、节点集合通信（Collective communication）：提供一对多的广播通信、多节点计算同步控制和对结果的规约（Reduce）计算功能

3、提供用户自定义的复合数据类型传输

接口：提供了6个最基本的编程接口，理论上任何并行程序都可以通过这六个基本API实现。

1. MPI\_Init(argc,argv):初始化MPI开始MPI并行计算程序体
2. MPI\_Finalize:终止MPI并行计算
3. MPI\_Comm\_Size(comm,size):确定指定范围内处理器/进程数目
4. MPI\_Comm\_Rank(comm,rank):确定一个处理器/进程的标识号
5. MPI\_Send(buf,count,datatype,dest,tag,comm):发送一个消息
6. MPI\_Recv(buf,count,datatype,source,tag,comm,status):接收消息

size:进程数，rank:指定进程的ID

comm:指定一个通信组（communicator）

dest:目标进程号，source:原进程标识号，tag:消息标签

5. 尝试在单机上安装并运行MPICH，并运行讲义P65页的简单示例。（运行结果截图）

