1.并行计算(Parallel Computing)

并行计算或称平行计算是相对于串行计算来说的。并行计算(Parallel Computing)是指同时使用多种计算资源解决计算问题的过程。为执行并行计算，计算资源应包括一台配有多处理机(并行处理)的计算机、一个与网络相连的计算机专有编号，或者两者结合使用。并行计算的主要目的是快速解决大型且复杂的计算问题。

并行计算可以划分成时间并行和空间并行。时间并行即流水线技术，空间并行使用多个处理器执行并发计算，当前研究的主要是空间的并行问题。以程序和算法设计人员的角度看，并行计算又可分为数据并行和任务并行。数据并行把大的任务化解成若干个相同的子任务，处理起来比任务并行简单。

空间上的并行导致两类并行机的产生，按照 Michael Flynn(费林分类法)的说法分为单指令流多数据流(SIMD)和多指令流多数据流(MIMD)，而常用的串行机也称为单指令流单数据流(SISD)。MIMD 类的机器又可分为常见的五类：并行向量处理机(PVP)、对称多处理机(SMP)、大规模并行处理机(MPP)、工作站机群(COW)、分布式共享存储处理机(DSM)。

2. 分布式计算(Distributed Computing)

分布式计算这个研究领域，主要研究分散系统(Distributed system)如何进行计算。分散系统是一组计算机，通过计算机网络相互链接与通信后形成的系统。把需要进行大量计算的工程数据分区成小块，由多台计算机分别计算，在上传运算结果后，将结果统一合并得出数据结论的科学。

目前常见的分布式计算项目通常使用世界各地上千万志愿者计算机的闲置计算能力，通过互联网进行数据传输。如分析计算蛋白质的内部结构和相关药物的项目，该项目结构庞大，需要惊人的计算量，由一台电脑计算是不可能完成的。即使现在有了计算能力超强的超级电脑，但是一些科研机构的经费却又十分有限。

分布式计算比起其它算法具有以下几个优点：

1、稀有资源可以共享。

2、通过分布式计算可以在多台计算机上平衡计算负载。

3、可以把程序放在最适合运行它的计算机上。其中，共享稀有资源和平衡负载是计算机分布式计算的核心思想之一。

3. 并行计算与分布式计算的区别

并行计算与分布式计算都是运用并行来获得更高性能，化大任务为小任务。简单说来，如果处理单元共享内存，就称为并行计算，反之就是分布式计算。也有人认为分布式计算是并行计算的一种特例。

但是分布式的任务包互相之间有独立性，上一个任务包的结果未返回或者是结果处理错误，对下一个任务包的处理几乎没有什么影响。因此，分布式的实时性要求不高，而且允许存在计算错误(因为每个计算任务给好几个参与者计算，上传结果到服务器后要比较结果，然后对结果差异大的进行验证。

分布式要处理的问题一般是基于“寻找”模式的。所谓的“寻找”，就相当于穷举法!为了尝试到每一个可能存在的结果，一般从 0～N(某一数值)被一个一个的测试，直到我们找到所要求的结果。事实上，为了易于一次性探测到正确的结果，我们假设结果是以某个特殊形式开始的。在这种类型的搜索里，我们也许幸运的一开始就找到答案;也许不够走运以至于到最后才找到答案，这都很公平。

这么说，并行程序并行处理的任务包之间有很大的联系，而且并行计算的每一个任务块都是必要的，没有浪费的分割的，就是每个任务包都要处理，而且计算结果相互影响，就要求每个的计算结果要绝对正确，而且在时间上要尽量做到同步，而分布式的很多任务块可以根本就不处理，有大量的无用数据块，所以说分布式计算的速度尽管很快，但是真正的“效率”是低之再低 的，可能一直在寻找，但是永远都找不到，也可能一开始就找到了;而并行处理不同，它的任务包个数相对有限，在一个有限的时间应该是可能完成的。

分布式的编写一般用的是 C++(也有用 JAVA 的，但是都不是主流)，基本不用 MPI 接口。并行计算用 MPI 或者 OpenMP。

4. 集群计算(Cluster Computing)

计算机集群将一组松散集成的计算机软件或硬件连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台计算机。集群系统中的单个计算机通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。集群计算机通常用来改进单个计算机的计算速度和/或可靠性。一般情况下集群计算机比单个计算机，比如工作站或超级计算机性价比要高得多。

根据组成集群系统的计算机之间体系结构是否相同，集群可分为同构与异构两种。集群计算机按功能和结构可以分为，高可用性集群(High-availability (HA) clusters)、负载均衡集群(Loadbalancing clusters)、高性能计算集群(High-performance (HPC)clusters)、网格计算(Grid computing)。

高可用性集群，一般是指当集群中有某个节点失效的情况下，其上的任务会自动转移到其他正常的节点上。还指可以将集群中的某节点进行离线维护再上线，该过程并不影响整个集群的运行。

负载均衡集群，负载均衡集群运行时，一般通过一个或者多个前端负载均衡器，将工作负载

分发到后端的一组服务器上，从而达到整个系统的高性能和高可用性。这样的计算机集群有时也被称为服务器群(Server Farm)。一般高可用性集群和负载均衡集群会使用类似的技术，或同时具有高可用性与负载均衡的特点。Linux 虚拟服务器(LVS)项目在 Linux 操作系统上提供了最常用的负载均衡软件。

高性能计算集群，高性能计算集群采用将计算任务分配到集群的不同计算节点儿提高计算能力，因而主要应用在科学计算领域。比较流行的 HPC 采用 Linux 操作系统和其它一些免费软件来完成并行运算。这一集群配置通常被称为 Beowulf 集群。这类集群通常运行特定的程序以发挥 HPC cluster 的并行能力。这类程序一般应用特定的运行库，比如专为科学计算设计的 MPI 库。HPC 集群特别适合于在计算中各计算节点之间发生大量数据通讯的计算作业，比如一个节点的中间结果或影响到其它节点计算结果的情况。

5. 网格计算(Grid Computing)

网格计算是分布式计算的一种，也是一种与集群计算非常相关的技术。如果我们说某项工作是分布式的，那么，参与这项工作的一定不只是一台计算机，而是一个计算机网络，显然这种“蚂蚁搬山”的方式将具有很强的数据处理能力。网格计算的实质就是组合与共享资源并确保系统安全。

网格计算通过利用大量异构计算机的未用资源(CPU 周 期和磁盘存储)，将其作为嵌入在分布式电信基础设施中的一个虚拟的计算机集群，为解决大规模的计算问题提供一个模型。网格计算的焦点放在支持跨管理域计算 的能力，这使它与传统的计算机集群或传统的分布式计算相区别。网格计算的目标是解决对于任何单一的超级计算机来说仍然大得难以解决的问题，并同时保持解决 多个较小的问题的灵活性。这样，网格计算就提供了一个多用户环境。

6. 集群计算与网格计算的区别

(1)简单地，网格与传统集群的主要差别是网格是连接一组相关并不信任的计算机，它的运作更像一个计算公共设施而不是一个独立的计算机。网格通常比集群支持更多不同类型的计算机集合。

(2)网格本质上就是动态的，集群包含的处理器和资源的数量通常都是静态的。在网格上，资源则可以动态出现，资源可以根据需要添加到网格中或从网格中删除。

(3)网格天生就是在本地网、城域网或广域网上进行分布的。网格可以分布在任何地方。而集群物理上都包含在一个位置的相同地方，通常只是局域网互连。集群互连技 术可以产生非常低的网络延时，如果集群距离很远，这可能会导致产生很多问题。物理临近和网络延时限制了集群地域分布的能力，而网格由于动态特性，可以提供 很好的高可扩展性。

(4)集群仅仅通过增加服务器满足增长的需求。然而，集群的服务器数量、以及由此导致的集群性能是有限的：互连网络容量。也就是说如果一味地想通过扩大规模来提高集群计算机的性能，它的性价比会相应下降，这意味着我们不可能无限制地扩大集群的规模。 而网格

虚拟出空前的超级计算机，不受规模的限制，成为下一代 Internet 的发展方向。

(5)集群和网格计算是相互补充的。很多网格都在自己管理的资源中采用了集群。实际上，网格用户可能并不清楚他的工作负载是在一个远程的集群上执行的。尽管网格与集群之间存在很多区别，但是这些区别使它们构成了一个非常重要的关系，因为集群在网格中总有一席之地—— 特定的问题通常都需要一些紧耦合的处理器来解决。然而，随着网络功能和带宽的发展，以前采用集群计算很难解决的问题现在可以使用网格计算技术解决了。理解网格固有的可扩展性和集群提供的紧耦合互连机制所带来的性能优势之间的平衡是非常重要的。

7. 云计算(Cloud Computing)

云计算是最新开始的新概念，它不只是计算等计算机概念，还有运营服务等概念了。它是分布式计算、并行计算和网格计算的发展，或者说是这些概念的商业实现。云计算不但包括分布式计算还包括分布式存储和分布式缓存。分布式存储又包括分布式文件存储和分布式数据存储。

8. 云计算与并行、分布式、网格和集群计算的区别

云计算是从集群技术发展而来，区别在于集群虽然把多台机器联了起来，但其某项具体任务执行的时候还是会被转发到某台服务器上，而云可以简单的认为是任务可以被分割成多个进程在多台服务器上并行计算，然后得到结果，好处在于大数据量的操作性能非常好。云可以使用廉价的 PC 服务器 ，可以管理大数据量与大集群，关键技术在于能够对云内的基础设施进行动态按需分配与管理。云计算与并行计算、分布式计算的区别，以计算机用户来说，并行计算是由单个用户完成的，分布式计算是由多个用户合作完成的，云计算是没有用户参与，而是交给网络另一端的服务器完成的。

9. 边缘计算

边缘计算起源于传媒领域，是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。其应用程序在边缘侧发起，产生更快的网络服务响应，满足行业在实时业务、应用智能、安全与隐私保护等方面的基本需求。边缘计算处于物理实体和工业连接之间，或处于物理实体的顶端。而云端计算，仍然可以访问边缘计算的历史数据。

边缘计算（edge computing）是指一种在网络边缘进行计算的新型计算模式，其对数据的处理主要包括两部分：其一是下行的云服务，其二是上行的万物互联服务。其中，边缘计算当中的“边缘”是一个相对的概念，主要是指从数据源到云计算中心路径之间的任意计算、存储以及网络相关资源。边缘计算起源于传媒领域，是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。其应用程序在边缘侧发起，产生更快的网络服务响应，满足行业在实时业务、应用智能、安

全与隐私保护等方面的基本需求。边缘计算处于物理实体和工业连接之间，或处于物理实体的顶端。而云端计算，仍然可以访问边缘计算的历史数据。

10. 边缘计算的好处

通过解决邻近问题，您可以解决延迟问题。设备上处理方法确保仅通过网络发送非关键数据，并且可以立即对关键数据执行操作。这对于延迟敏感的应用程序很重要，例如自动驾驶汽车，其中必须等待毫秒可能是站不住脚的。边缘计算的分散方法也降低了带宽。数据处理从收集点开始，只有需要存储的数据才会发送到云端。这使边缘计算更加高效和可扩展，并减少网络负载。
从理论上讲，边缘计算还有一层额外的安全性，因为来自物联网设备的大部分数据都不会遍历网络。相反，它仍然处于创造的位置。云中的数据越少意味着泄漏或泄漏的数据就越少。

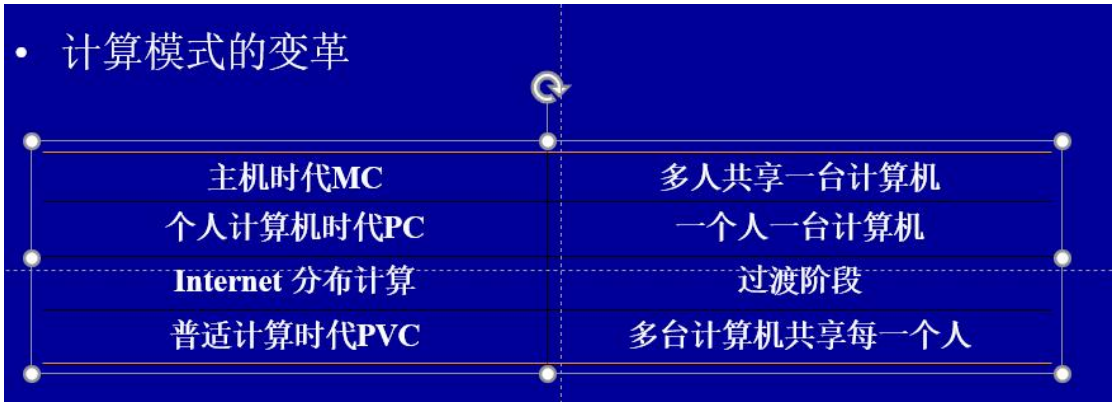
11. 物联网

物联网（The Internet of Things，简称 IOT）是指通过 各种信息传感器、射频识别技术、全球定位系统、红外感应器、激光扫描器等各种装置与技术，实时采集任何需要监控、 连接、互动的物体或过程，采集其声、光、热、电、力学、化 学、生物、位置等各种需要的信息，通过各类可能的网络接入，实现物与物、物与人的泛在连接，实现对物品和过程的智能化感知、识别和管理。物联网是一个基于互联网、传统电信网等的信息承载体，它让所有能够被独立寻址的普通物理对象形成互联互通的网络

普适计算：

IBM 在 1999 年提出了 Pervasive Computing 这个名词。此后普存计算又称为**普适计算（UC）**。
基本思想：把计算机融入到环境中去，使人们关注的重点从操作工具转移到执行任务本身上来，可以在任意时间、使用任意设备、通过任意网络来获得所需的服务。

UC 的动机：回答和解决 PC 所存在的一些根本问题：
过分复杂而难以使用
过分要求人的注意力
过分隔绝于他人和现实活动
过分的支配作用使我们的桌面和生活犹如殖民地
UC 的目的：更多集中于人与人的交互，而不是人与机器的交互。
普适计算思想的产生最根本的原因：计算资源的丰富与廉价；互联网（无线网络）的广泛使用。
计算模式的演变：计算机完成任务的一种运行、输入输出以及使用的方式。

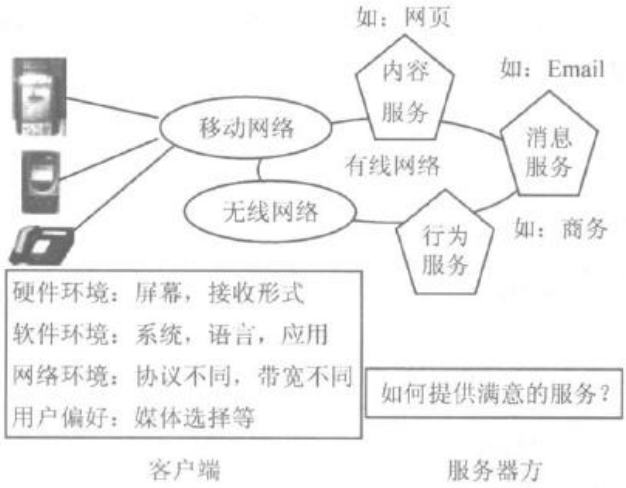


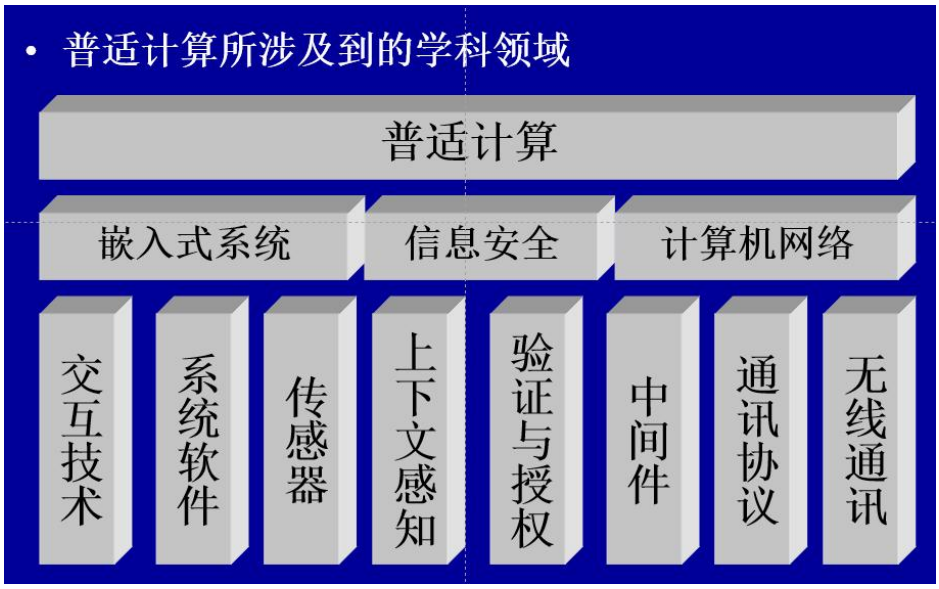
普适计算时代：大量计算机共享我们每一个人，其中数百台计算机可以在几分钟的因特网浏览中被访问，其他计算机则嵌入在墙壁、椅子、衣服、电灯开关、汽车等一切东西中。基本特征是深度的嵌入计算，即连接现实世界中一切具有计算能力但规模大小不同的东西。

普适计算的要求：普适性 数量众多的计算设备嵌入到环境中，通过这些设备用户可以随时随地得到计算服务。
透明性 计算过程对于用户是透明的。如果计算系统返回的结果无法满足用户的需求，用户也可以直接调节系统使之工作在更好的状态。在用户进行调节的同时，整个计算系统也在不断地更新和学习。整个计算和学习过程对于用户来说是不可见的，这就可以使用户最大程度地将注意力放在要完成的任务上。
动态性 在普适计算环境中，用户通常处于移动状态，这导致在特定的空间内用户集合将不断变化；另一方面，移动设备也会动态地进入或退出一个计算环境，这导致计算系统的结构也在发生动态变化。
自适应性 计算系统可以感知和推断用户需求，自发地提供用户需要的信息服务。
永恒性 计算系统不会关机或者重启，计算模块可以根据需求、系统错误或系统升级等情况加入或离开计算系统。

普适计算的特点：
以人为本：由于计算资源的稀缺和计算能力的有限，传统模式中都是以计算为中心展开的。在普适计算环境下，人们随时随地都可能使用到多个计算设备，这时人的注意力就成为一种稀缺资源。普适计算关注的是人们的注意力以及人们对计算的满意程度，它是一种以人为中心的计算模式。
实现了物理世界与信息空间的融合：
主机计算
能通过数量有限的大型主机获取和处理有限的信息
桌面计算
互联网出现。信息空间在不断扩大，但信息空间的入口仍然有限，只能通过有限的手段（即桌面计算设备）来获取和处理信息。
普适计算
大量异构设备共同分享和处理信息

一种普适计算环境下的服务模式：





交互技术：

物理空间和信息空间之间无需人的干预交互：即其中任一个空间状态的改变可以引起另一个空间的状态的相应改变；语气识别(在语音识别上扩展)；肢体语言识别(如手势、面部表情等)；地理位置识别等；输出的自适应性要求系统能够根据用户使用的设备类型，产生适合于该设备的接口。目前研究集中于允许应用程序根据用户接口的抽象定义，结合目标设备的特点，自动生成恰当的界面。

上下文感知：

能够感知在当前的情景中与交互的任务有关的上下文，并据此做出决策和自动地提供相应的服务。

--用户的手机能够在进入电影院后自动进入振动模式

--热水器能在用户运动回来后自动打开

自适应技术：

在普适计算环境中，各种设备拥有的资源不同并且处于变化之中。从各种嵌入式设备、各种传感器、各种笔记本电脑和个人电脑、到各种基础设施中的服务器。它们在计算能力、存储容量、能量供应、网络带宽、作用范围以及交互手段等因

素上有着很大的差异。此外，设备(特别是手持或可穿戴)在不同的环境中移动时，周围环境和用户需求都可能发生比较大的改变，有可能出现用户的资源请求与现场资源不匹配的矛盾。
普适计算中的自适应主要包括：软件自适应；硬件自适应；服务自适应
硬件自适应目前的研究重点是可重构技术
软件的自适应主要是从操作系统剪裁和软件在线维护等方面展开研究
服务自适应和上下文感知计算类似：区别在于前者主要是从服务的手段和质量来解决用户需求和可用资源矛盾的问题

研究主要集中在中间件和系统软件：它们对普适计算中大量的联网的设备、物体、计算机实体进行管理，为它们之间的数据交换、消息交互、服务发现、任务协调、任务迁移等等提供系统级的支持。

服务模型、发现、组织、及安全体系：
普适计算中的服务通常是无缝的、透明的、和自适应的。传统的服务模式为请求一服务模式。普适计算环境是一个高度异构、动态的变化系统，服务分布在不同的设备和系统，既有垂直整合的服务，也有水平方向的服务。一个强壮的服务模式和组织，能够更好的支持信息的传递、月及务的实施。被服务的实体进入某空间后，通过服务发现协议获得所有授权的可用服务，在不需要用　户介入的情况下，能够自动通信，根据相关上下文实施服务。安全体系可以保证用户的隐私，服务安全以及用户和服务提供者的利益。

云计算：

网格计算：
“网格计算”这个词来源于另一专业名词“电力供应网”
--电力供应网的原意是电力供应商根据用户的需要供应电力，消费者只需支付自己使用的那部分电费，网格计算的基本思想也因此被引申。

--它利用互联网把地理上广泛分布的各种资源(包括计算资源、存储资源、带宽资源、软件资源、数据资源、信息资源、知识资源等)连成一个逻辑整体，为用户提供一体化信息和应用服务(计算、存储、访问等)。

云计算：
计算资源作为“服务”可以通过互联网来获取。在“云”里可能有成千上万台计算机，对于“云”外面的使用者来说，他看到只是一个统一的“服务”（或接口）界面，就像在使用一台巨大的虚拟“计算机”，用户可以通过互联网像使用本地计算机一样使用“云”计算机。

云计算与网格计算：
技术本质上来说“云”计算与网格并没有根本上的区别
----基于互联网实现分布式计算资源的整合，以服务的形式输出给需要的人，按需提供服务的计算模式。
但是，网格要整合的是资源差异性比较大的节点（硬件、操作系统、应用资源都

不同的计算机或平台），并且其分布式节点可以分布于全球各地（属于不同所有者），通过互联网连接成“一体”
“网格”至今仍然主要在学术界或专业领域应用。
----还需要互联网取得进一步发展，在可靠性、安全性和数据传送性能方面获得根本解决并且在业务与管理平台、认证计费系统等逐渐完善后，才有可能在商业上广泛应用。
“云”计算整合的是“集群”服务器资源，这些资源属于某个公司所有，可以实现很高程度的控制，实现更优的配置与管理，并且通常服务器都是通过局域网连接（“云”计算企业的多个数据中心之间则可以通过专线连接），具有更好的网络可靠性、安全性和连接性能，同时大部分“云”计算提供商都具有非常强大的互联网运营平台和技术能力。

云计算与集群计算：
集群是另一种计算模式，它向用户提供强大的计算能力，但这些用户只限于内部使用（通过 LAN 或专有网络），而“云”计算将“集群”计算能力通过互联网向普通的外部用户提供。

云计算产生的条件：
互联网的发展（空间、带宽）
分布、并行、虚拟化技术的发展
----底层的物理和软件技术

互联网上大型的数据中心计算和存储能力出现冗余
基于互联网的“服务”存取技术逐渐成熟

云计算中的关键问题：
- **如何封装原始的“云”计算资源能力**
  - **资源虚拟化等硬件技术**
  - **资源供给管理**
  - **资源配置**
  - **资源监视等中间件技术**
  - **分布式、并行计算等软件技术**
- **以更加简单的接口形式提供给用户或第三方开发者**
  - **WEB Services 等分布式接口技术**
  - **云应用软件的编程技术和编程环境**

云计算的现状：
- **“云”计算目前也没有一套统一的标准**
  - **微软为了保护其已有的终端桌面系统优势，提出了“云+端”的概念，即终端计算+“云”计算将结合向用户提供计算，终端计算将继续发挥作用。**
  - **IBM 结合自己在高性能计算、中间件等方面的优势，将高性能计算机集群以“云”计算形式向用户提供。主要面向企业客户。**
  - **Google 的 “云”不仅仅是由大量的分布式的普通计算机构成，而**

且其客户主要是普通的消费者（可能基于其广告盈利模式来提供“云”计算服务），同时向企业用户延伸。

需要改进问题：

- 网络的成本和带宽

目前虽然宽带已经比较经济，但真正能够无限制使用的用户规模还很小。并且带宽还无法满足用户的流畅体验，特别是无法满足大规模的“云”计算用户的需求。

- 互联网可以随时随地无缝接入。
- 可靠性
  - 互联网能够实现更高的连接可靠性
  - 数据中心本身的可靠性
- 安全性
  - 数据安全性
  - 应用安全性
- 服务与用户界面
  - 服务与用户界面主要面向专业的开发人员
  - “云”服务的接口尚没有标准，应用之间难以兼容
  - “云”服务的客户端（比较公认的是浏览器）也还无法满足用户桌面应用的体验需求
- 用户习惯

在未来，云计算并不会完全取代桌面计算：在用户计算需求的满足上”云“计算会逐渐成为主流，桌面计算则成为辅助的设施。

普适计算与云计算

- 概念不同
  - 普适计算是学术概念
  - 云计算是商业概念
- 高度不同
  - 普适计算是一种思想：融合
  - 云计算是一种手段
- 研究的重心不同
  - 普适计算强调终端与服务
  - 云计算更关注集合与协同
- 应该成为未来并存的计算模式

试题 part：

名词解释题：

1. 网格计算和集群计算

网格计算：分布式计算，是一门计算机科学。它研究如何把一个需要非常巨大的计算能力才能解决的问题分成许多小的部分，然后把这些部分分配给许多计算机进行处理，最后把

这些计算结果综合起来得到最终结果。网格诞生于那些非常需要进行协作的研究和学术社区。研究中非常重要的一个部分是分发知识的能力 —— 共享大量信息和帮助创建这些数据的计算资源的效率越高，可以实现的协作的质量就越好，协作级别也越广泛。网格计算的本质在于以有效且优化的方式来利用组织中各种异构松耦合资源，来实现复杂的工作负载管理和信息虚拟化功能。（注意，一个组织可能会跨越很多部门、物理位置等。我们此处使用的是 “组织” 一词的抽象意义。）

集群计算：关注计算资源，一般包含同种处理器和操作系统。集群包含的处理器和资源的数量通常都是静态的。从物理上看，集群计算也是在同一个地理位置上进行的。

2. 网络的四种体系结构

Host/terminal 主机+终端，多用户系统，host 系统

工作站/文件服务器系统 workstation/file server 共享文件服务器上的文件和设备

客户机/服务器方式 client/server 共享服务器资源，包括文件、设备和其他软硬件

对等网络方式（Peer-to-peer network）每台设备都处于对等地位，既做客户机也做服务器

3. 负载共享和负载均衡

4. 深度计算和广度计算

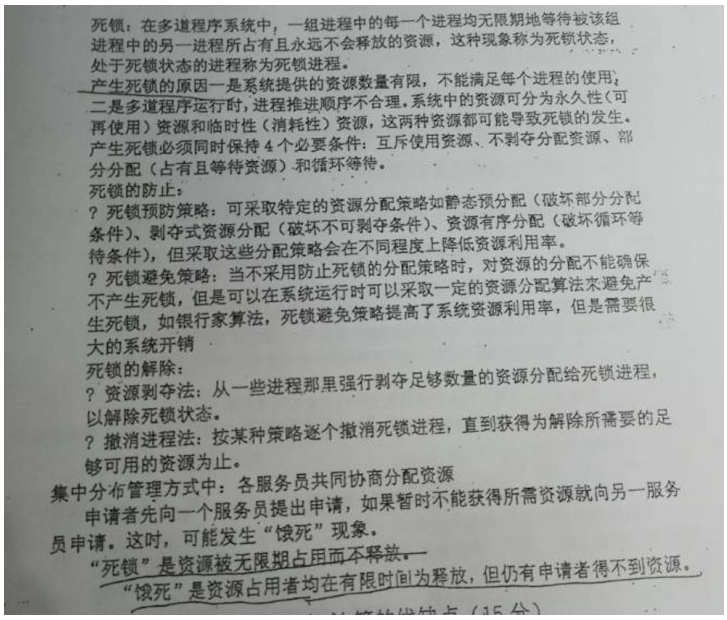
深度：具体一点就是访问一个顶点之后，我继而访问它的下一个邻接的顶点，如此往复，直到当前顶点一被访问或者它不存在邻接的顶点。同样，算法导论采用了“聪明的做法”，用三种颜色来标记三种状态。但这三种状态不同于广度优先搜索：WHITE 未访问顶点，GRAY 一条深度搜索路径上的顶点，即被发现时，BLACK 此顶点的邻接顶点被全部访问完之后——结束访问次顶点

广度：具体一点就是每个顶点只访问它的邻接节点（如果它的邻接节点没有被访问）并且记录这个邻接节点，当访问完它的邻接节点之后就结束这个顶点的访问。广度优先用到了“先进先出”队列，通过这个队列来存储第一次发现的节点，以便下一次的处理；而对于再次发现的节点，我们不予理会——不放入队列，因为再次发现的节点：无非是已经处理完的了或者是存储在队列中尚未处理的。

5. 死锁和饿死

相同点：二者都是由于竞争资源而引起的，与资源的分配策略有关，因而防止饿死与死锁可从公平性方面考虑如 FCFS 先到先服务算法。

不同点：①从进程状态考虑，死锁进程都处于等待态（等待某一不可被剥夺资源被释放），饿死进程可能处于忙式等待(就绪队列上等待可剥夺处理机资源)。（忙式等待：不进入等待状态的等待实际状态为”运行“或者”就绪“忙式等待空耗处理器资源因而是低效的,进程无法向前推进等待某一事件，但不主动放弃处理器而是不断循环检测资源是否可用）。②死锁进程等待永远不会被释放的资源，饿死进程等待会被释放但却不会分配给自己的资源。③死锁一定发生了循环等待，而饿死则不然，这也表明通过资源分配图可以检测死锁存在与否，但不能检测是否有进程饿死。④死锁一定涉及多个进程，而饥饿或被饿死的进程可能只有一个。


死锁：在多道程序系统中，一组进程中的每一个进程均无限期地等待被该组进程中的另一进程所占有且永远不会释放的资源，这种现象称为死锁状态，处于死锁状态的进程称为死锁进程。
产生死锁的原因一是系统提供的资源数量有限，不能满足每个进程的使用；二是多道程序运行时，进程推进顺序不合理。系统中的资源可分为永久性（可再使用）资源和临时性（消耗性）资源，这两种资源都可能导致死锁的发生。产生死锁必须同时保持4个必要条件：互斥使用资源、不剥夺分配资源、部分分配（占有且等待资源）和循环等待。
死锁的防止：
？死锁预防策略：可采取特定的资源分配策略如静态预分配（破坏部分分配条件）、剥夺式资源分配（破坏不可剥夺条件）、资源有序分配（破坏循环等待条件），但采取这些分配策略会在不同程度上降低资源利用率。
？死锁避免策略：当不采用防止死锁的分配策略时，对资源的分配不能确保不产生死锁，但是可以在系统运行时可以采取一定的资源分配算法来避免产生死锁，如银行家算法，死锁避免策略提高了系统资源利用率，但是需要很大的系统开销
死锁的解除：
？资源剥夺法：从一些进程那里强行剥夺足够数量的资源分配给死锁进程，以解除死锁状态。
？撤消进程法：按某种策略逐个撤消死锁进程，直到获得为解除所需要的足够可用的资源为止。
集中分布管理方式中：各服务员共同协商分配资源
申请者先向一个服务员提出申请，如果暂时不能获得所需资源就向另一服务员申请。这时，可能发生“饿死”现象。
“死锁”是资源被无限期占用而不释放。
“饿死”是资源占用者均在有限时间为释放，但仍有申请者得不到资源。

- 分析为什么三层客户/服务计算模式可以减少数据库系统 license 个数
- 分析任务划分粒度的大小对任务调度的影响
- 对网格计算发展趋势的看法

网格计算专门针对复杂科学计算的新型计算模式。这种计算模式是利用互联网把分散在不同地理位置的计算机组织成一个虚拟的超级计算机，每一台参与计算的计算机就是一个节点，而整个计算是由成千上万个节点组成的一张网格。这样组织起来的虚拟的超级计算机有 2 个优势：①数据处理能力超强，网格计算是分布式计算的一种，参与工作的是计算机网络，显然比以个人计算机为的单位的计算方式具有更加强大的数据处理能力。②能充分利用网上的闲置处理能力，网格计算模式首先把要计算的数据分割，通常实现的软件是一个预先编制好的屏幕保护程序，然后，不同节点的计算机可以根据自己的处理能力下载一个或多个数据片断和这个屏幕保护程序，只要位于节点的计算机的用户不使用计算机时，屏保程序就会工作网格计算至少需要具备以下 3 种基本功能：①任务管理：用户提交任务，为任务指定所需资源，删除任务并监测任务运行状态②任务调度：用户提交的任务由该功能按照任务类型、所需资源、可用资源等安排运行日程和策略。③资源管理：确定并监督网络资源状况，收集任务运行时的资源占用数据。为实现网格计算的目标，必须重点解决以下 3 个问题，这也是目前 Internet 普遍存在的问题：①异构性：由于网格由分布在广域网上下同管理域的各种计算资源组成，怎样实现异构机器间的合作和转换是首要问题。②可扩展性：要在网络资源规模不断扩大、应用不断增长的情况下，不降低性能。③动态自适应性。

- 设计一套软件能够在前端计算机录入成绩，在后端输出该学生信息和总分
- 动态负载均衡有哪三种实现模式？它们的基本思想分别是什么？每种实现模型各有什么优缺点？

基于 DNS 负载均衡，基于硬件负载均衡，基于软件负载均衡。三种方案各有优劣，DNS 负载均衡可以实现在地域上的流量均衡，硬件负载均衡主要用于大型服务器集群中的负载需求，而软件负载均衡大多是基于机器层面的流量均衡。在实际场景中，这三种是可以组合在一起使用。

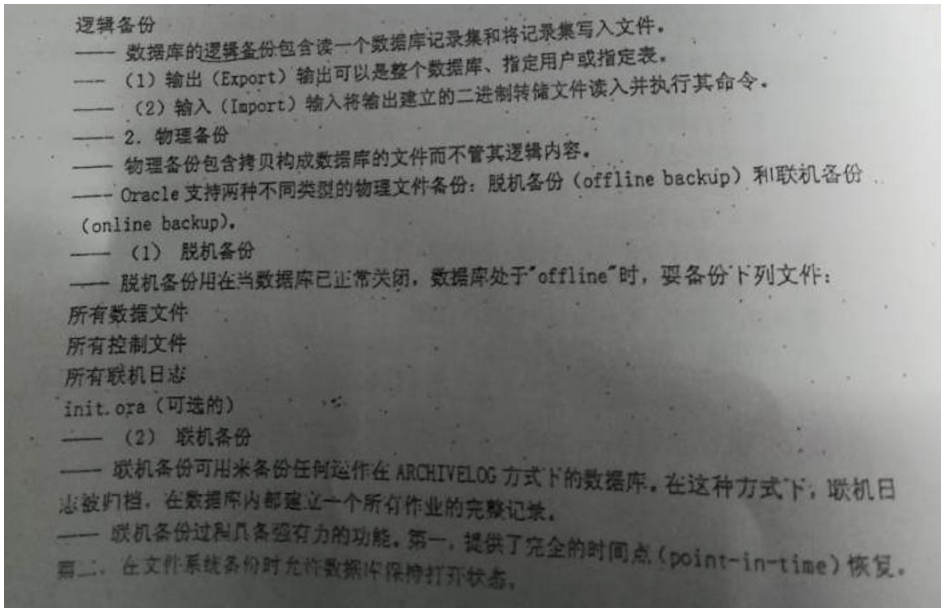
1. 使用 DNS 做负载均衡的方案，天然的优势就是配置简单，实现成本非常低，无需额外的开发和维护工作。但是也有一个明显的缺点是：当配置修改后，生效不及时。这个是由于 DNS 的特性导致的，DNS 一般会有多级缓存，所以当我们修改了 DNS 配置之后，由于缓存的原因，会导致 IP 变更不及时，从而影响负载均衡的效果。另外，使用 DNS 做负载均衡的话，大多是基于地域或者干脆直接做 IP 轮询，没有更高级的路由策略，所以这也

是 DNS 方案的局限所在。

2. 硬件的负载均衡，性能是非常的好，每秒能处理的请求数达到百万级，即几百万/秒的负载，价格昂贵。因为这类设备一般用在大型互联网公司的流量入口最前端，以及政府、国企等企业。采用 F5 这类硬件做负载均衡的话，主要就是省心省事，买一台就搞定，性能强大，一般的业务不在话下。而且在负载均衡的算法方面还支持很多灵活的策略，同时还具有一些防火墙等安全功能。

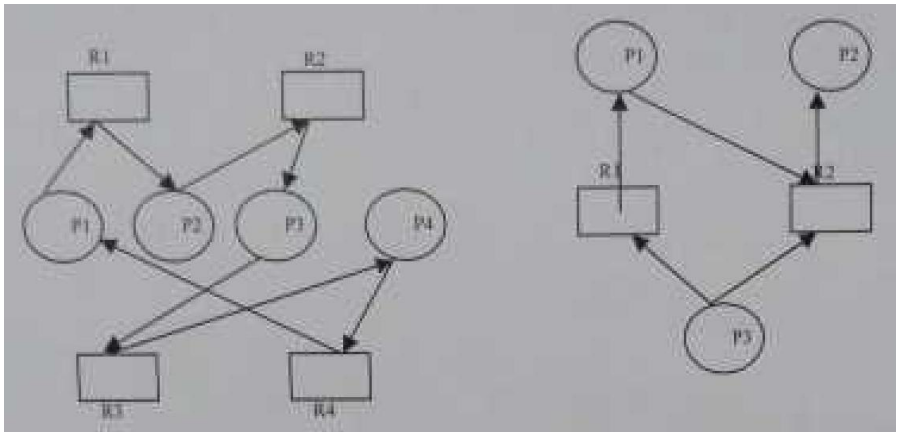
3. 软件负载均衡是指使用软件的方式来分发和均衡流量。软件负载均衡，分为 7 层协议 和 4 层协议。网络协议有七层，基于第四层传输层来做流量分发的方案称为 4 层负载均衡，例如 LVS，而基于第七层应用层来做流量分发的称为 7 层负载均衡，例如 Nginx。这两种在性能和灵活性上是有些区别的。基于 4 层的负载均衡性能要高一些，一般能达到 几十万/秒 的处理量，而基于 7 层的负载均衡处理量一般只在 几万/秒 。基于软件的负载均衡的特点也很明显，便宜。在正常的服务器上部署即可，无需额外采购，就是投入一点技术去优化优化即可，因此这种方式是互联网公司中用得最多的一种方式。

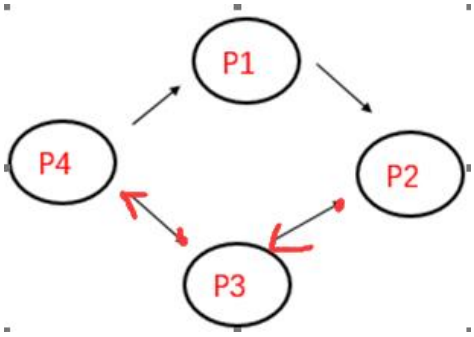
- 比较二层和三层 C/S 计算的优缺点
- 逻辑备份可以实现数据库的全面恢复吗？逻辑备份的优点是什么？逻辑备份是否可以替代物理备份？

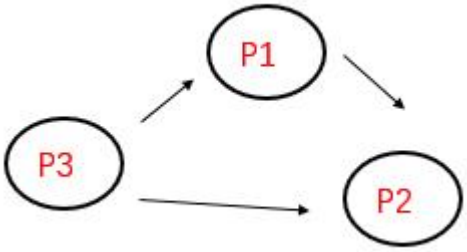

逻辑备份
—— 数据库的逻辑备份包含读一个数据库记录集和将记录集写入文件。
—— (1) 输出 (Export) 输出可以是整个数据库、指定用户或指定表。
—— (2) 输入 (Import) 输入将输出建立的二进制转储文件读入并执行其命令。
—— 2. 物理备份
—— 物理备份包含拷贝构成数据库的文件而不管其逻辑内容。
—— Oracle 支持两种不同类型的物理文件备份：脱机备份 (offline backup) 和联机备份 (online backup)。
—— (1) 脱机备份
—— 脱机备份用在当数据库已正常关闭，数据库处于"offline"时，要备份下列文件：
所有数据文件
所有控制文件
所有联机日志
init.ora (可选的)
—— (2) 联机备份
—— 联机备份可用来备份任何运作在 ARCHIVELOG 方式下的数据库。在这种方式下，联机日志被归档，在数据库内都建立一个所有作业的完整记录。
—— 联机备份过程几条很有力的功能。第一，提供了完全的时间点 (point-in-time) 恢复。第二，在文件系统备份时允许数据库保持打开状态。

1. **什么情况会出现死锁和饿死？**
   a) 死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，他们都将无法进行下去。此时系统处于死锁状态，这些永远在互相等待的进程称为死锁进程。
   b) 饿死是资源占用者均在有限时间内释放，但由于调度策略的不公平，仍有进程持续申请不到资源，无法进行下去，导致饿死。

2. **将下面两个资源分配图转换为等待图。**







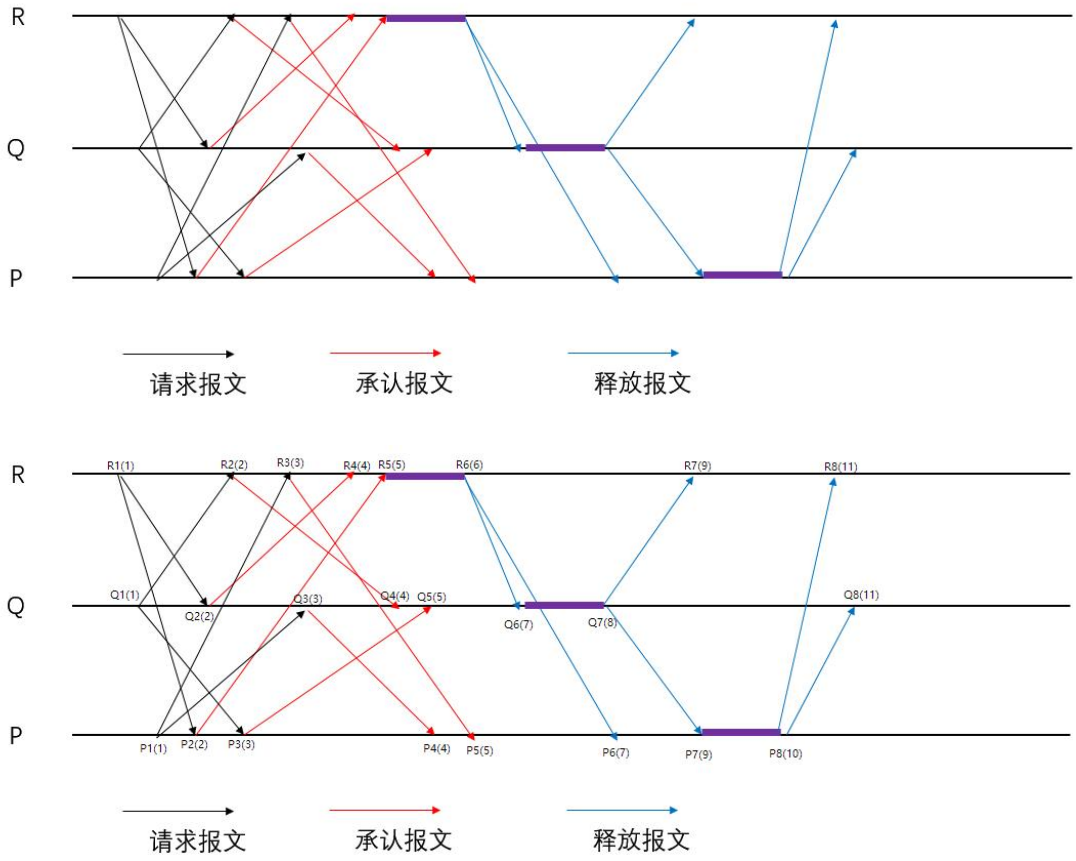
3. 假设使用 AND 模型，确定以上两个图是否存在死锁？

在使用 AND 条件的系统中，死锁条件是在等待图中存在回路。所以图一中存在回路 P1—P2—P3—P4,故图一存在死锁。图二则没有闭合的回路，所以不存在死锁。

4. 一个分布计算系统中存在 3 个进程 P、Q、R，PQ 和 R 分别在 p1,q1 和 r1 时刻（逻辑时钟 p1>q1>r1）发出进去同一临界区的请求，系统采用 Lamport 时间戳互斥算法进行互斥操作。

1. 画出时空图，标出各进程之间所有报文传递情况，并注明报文类别。找出 PQ 发出请求后分别进入临界区的时间段。
2. 设定 PQR 上各事件的标量逻辑时钟。
3. 你能够改进临界区调度算法以提高性能？

1.



3.可以通过减少报文数量来提高算法性能。例如：①进程在发出报文后，不需要得到所有进程的回应，只需得到其请求子集中的所有进程的回应即可进入临时区。

②释放临界区时只需向其请求子集中的所有进程发送释放报文。

5.MPI 编程

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

main(int argc,char *argv[]){
    int i,tmp,sum =0,n;
    int group_size; //进程总数
    int my_rank;    //进程id
    MPI_Status status;
    MPI_Init(&argc,&argv);  //初始化MPI
    MPI_Comm_size(MPI_COMM_WORLD,&group_size); //得到总的进程数
    MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);  //得到自己的进程数
    if(my_rank == 0){ //第一个进程发送消息
        MPI_Send(&n,1,MPI_DOUBLE,1,99,MPI_COMM_WORLD); //发送消息
        for(i=my_rank;i<n;i+=group_size) sum += i;
        MPI_Recv(&tmp,1,MPI_DOUBLE,1,88,MPI_COMM_WORLD,&status);
        sum += tmp;
        printf("\n The result = %d",sum);
    }
    else{
        MPI_Recv(&n,1,MPI_DOUBLE,0,99,MPI_COMM_WORLD,&status);
        for(i=my_rank;i<n;i+=group_size) sum += i;
        MPI_Send(&sum,1,MPI_DOUBLE,0,88,MPI_COMM_WORLD);
    }
    MPI_Finalize(); //结束main
}
```

6.说明 Map\Reduce 技术原理。试解释如何在 Hadoop 环境下，通过 Map\Reduce 技术，实现大文本文件中每个单词出现的次数。

1. MapReduce 模型主要包含 Mapper 类和 Reducer 类两个抽象类。Mapper 将输入键值对(key/value pair)映射到一组中间格式的键值对集合。；Reducer 将与一个 key 关联的一组中间数值集归约（reduce）为一个更小的数值集。

2. 在 hadoop 中，输入的数据是<文本具体地址，文本内容>，之后 map 函数将文本内容进行分词，变成<文本来源，具体词汇>的键值对的集合，同是筛除文本中的不必要成分，比如英语中的 is、a，中文中的“的”，“是”等，之后由 reduce 函数对词频进行统计，用一个<具体词汇，出现总数>的键值对集合记录所有词的词频，每多统计到一个词，就在特定的项上的“出现总数”上加一。最后，根据 reduce 得到的这个键值对集合，输出对应词的词频