1、分布式系统与计算机网络的区别和联系？

区别：

计算机网络：计算机网络（Computer Network）是指实现计算机之间通讯的软件和硬件系统的统称，从广义上讲，利用磁盘在两台微机之间拷贝数据也可以认为是一种特殊的网络。它的更加具体的定义是“以共享资源为目的，通过数据通讯线路将多台计算机互联而组成的系统”，共享的资源包括计算机网络中的硬件设备、软件或者数据。特点可以概括如下：

1、计算机网络的主要目的是实现计算机资源的共享；

2、联网计算机是分布在不同地理位置上的多台独立的计算机系统；

3、网间计算机必须遵循全网统一的网络协议。

分布式计算机系统：定义为“存在着一个能为用户自动管理资源的网络操作系统，由它调用完成用户任务所需要的资源，而整个网络像一个大的计算机系统一样对用户是透明的。”特点可以概括如下：

1、系统拥有多种通用的物理和逻辑资源，他们可以被动态的分配任务；

2、系统中分散的物理和逻辑资源通过计算机网络进行信息交换；

3、系统存在着一个以全局方式管理系统资源的分布式操作系统；

4、系统联网的各计算机及相互合作又各自独立自治；

5、系统内部结构对于用户而言完全透明。

计算机网络是松耦合系统，分布式系统是紧耦合系统，计算机网络为分布式系统提供了基础，分布式系统是计算机网络的高级形式。

联系：

一般的分布式系统是建立在计算机网络之上的，因此分布式系统与计算机网络在物理结构上基本相同；

计算机网络为分布式系统研究提供了技术基础，而分布式系统是计算机网络技术发展的高级阶段。

2、13年第二题 第四章第7题

不同之处：服务器在接收到RPC请求后立即向客户送回应答，之后再调用客户请求的过程。应答的作用是向客户确认服务器已准备开始处理该RPC请求。客户接收到服务器的确认消息后将不会阻塞，而是继续向下执行。 （课本P96）

异步和同步的区分在于是否等待服务端执行完成并返回结果。

1. 同步调用

客户方等待调用执行完成并返回结果。

2. 异步调用

客户方调用后不用等待执行结果返回，但依然可以通过回调通知等方式获取返回结果。

若客户方不关心调用返回结果，则变成单向异步调用，单向调用不用返回结果。

**二者并不相同。异步RPC 向调用者返回一个通知，这意味着客户第一次调用之后，有**

**一个额外的消息会被发送。类似地，服务端接收到它的响应消息已经发送到客户端的通知。**

**如果保证通信可靠的话，两次异步RPC 调用是一样的。**

3、13年第三题 （第7章17题）

写后读（p213）：一个进程对于数据x的写操作，那么进程无论到任何副本上都应该看到这个写操作的影响，也就是看到和自己写操作的影响或者更新的值。

顺序一致性

对于一些读写写操作的集合，所有进程看到的都是同样的顺序。也就是将并行的操作序列化，而且每个进程得到的序列都相同。

**不一定。只要更新进程接收到更新正在被处理的确认，该进程就会从数据存储断开**

**并连接另一个副本。但不能保证更新已经到达了那个副本。相比之下，如果存在阻塞协议，**

**更新进程只能在更新被传播到其他副本时才会断开连接。**

4、13年第四题

第八章课后习题3

5、13年第五题

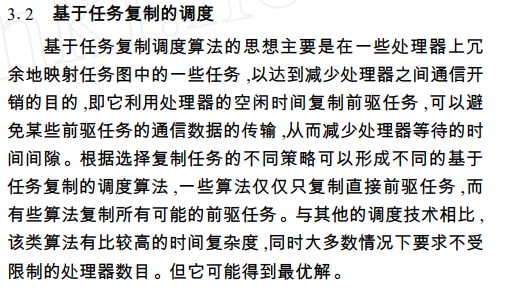
**调度算法见ppt**

两台处理机的最佳调度。

一个是最早调度算法，一个是考虑通信延迟的调度算法。

假设两个处理机为p1和p2,t1,t2,t3的执行时间分别是w(t1)，w(t2)，w(t3)，画出甘特图。

第二种注意通信延迟



基于任务复制的分簇算法是将基于任务复制的策略和基于分簇的策略相结合

的一种新的策略，其目标是：通过对前驱任务复制，减少不同处理器之间相依赖

任务之间的通信开销，对复制后的任务集进行分簇处理，通过合理的分簇策略使

整个任务集的调度长度尽可能的少，并通过对分簇操作之后的簇再进行合并除冗

操作使所需处理器的数目尽可能的少，进而提高系统的利用率。正因此，基于任

务复制的分簇策略适合于任务之间通信频繁，处理速度要求高，系统的任务量大

的情况。

基于多核处理器任务复制的分簇策略是将基于任务复制和基于簇的策略相结

合，结合了基于复制和分簇的两类策略的优点，这类算法的基本过程通常由以下

4 个步骤组成：

步骤 1：通过对任务之间的通信开销以及任务之间的依赖关系的分析，按照

一定方法适当的选取前驱任务进行复制。

步骤 2：根据任务的特征原则，以及调度策略将任务按照一定的目的分成一

系列的簇，同一簇上的任务将被安排到同一处理器上执行。

步骤 3：将不同的簇按照一定的分簇原则映射到不同的处理器上，并根据实

际处理器的数目以及在步骤 2 形成的簇数，适当对簇的数目进行调整，如果处理

器数目大于簇的数目，那么簇的数目可以不需要调整，如果处理器数目小于簇的

数目，则必须按照簇合并策略进行簇的合并处理,此外如果对任务有特殊要求，以

及满足特定调度要求，对同一簇上任务进行局部调整。

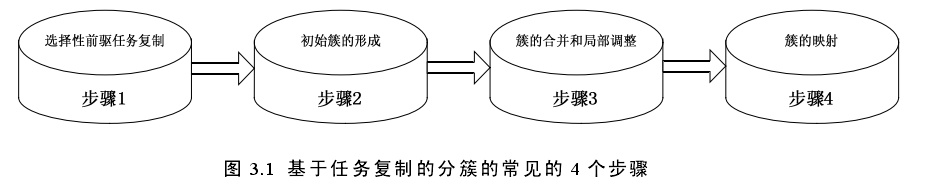
步骤

4：将最后形成的簇，按照调度策略，将簇映射到相对应的处理器上。

簇中的任务按照在簇中的顺序，一一在该处理器上执行。其这个 4

个步骤如下图3.1 所示,但是这 4 个步骤不是所有基于任务复制的分簇算法的都必须具备的。在不同基于任务复制的分簇算法中，由于调度策略的不同，以及调度目标的不同，

这 4 个步骤在顺序上会做出适当的调整。



6、2013年第6题

分片透明性：用户不必关心数据是如何分片，他们对数据的操作在全局关系上进行的，即关心如何分片对用户是透明的，因此，当分片改变时应用程序可以不变。

分片透明性是最高层次的透明性，如果用户能在全局关系一级操作，则数据如何分布，如何存储等细节不必关心，其应用程序的编写与集中式数据库相同。

位置透明性：用户不必知道所操作的数据放在何处，即数据分配到哪个或哪些站点存储对用户是透明的。因此，数据分片模式的改变，如把数据从一个站点转移到另一个站点将不会影响应用程序，因而应用程序不必改写。

逻辑透明性（局部映像透明性）：它是最低层次的透明性，该透明性提供数据到局部数据库的映像，即用户不必关心局部DBMS支持哪种数据模型、使用哪种数据操纵语言，数据模型和操纵语言的转换是由系统完成的。因此，局部映像透明性对异构型和同构异质的分布式数据库系统时非常重要的。

分片透明性：分片透明性位于全局概念模式与分片模式之间，是指用户只需对全局关系进行操作，不必考虑数据的分片及存储场地，其应用程序的编写与集中式数据库相同。当分片模式改变时，只需改变全局概念模式到分片模式之间的映像，而不会影响到全局概念模式和应用程序。

位置透明性：位置透明性位于分片模式与分配模式之间，是指用户不必知道数据的存储场地，即数据分配到哪个或哪些场地存储对用户是透明的。当存储场地发生变化时，只需改变分片模式到分配模式之间的映像，而不会影响分片模式、全局概念模式和应用程序。

局部数据模型透明性：局部数据模型透明性也称局部映像透明性，位于分配模式与局部概念模式之间，是指用户不用考虑局部DBMS所支持的数据模型、使用郧种数据操纵语言，但要考虑数据如何分片、片段及其副本在各场地上的分配。

假设输入的学号为A

1\\Select Sname, Sage

from Student

where Sno = A

2\ \Select Sname, Sage

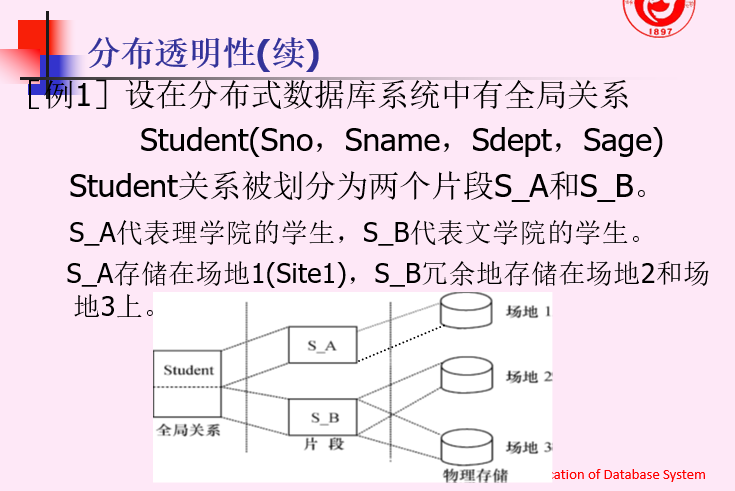
from S\_A, S\_B

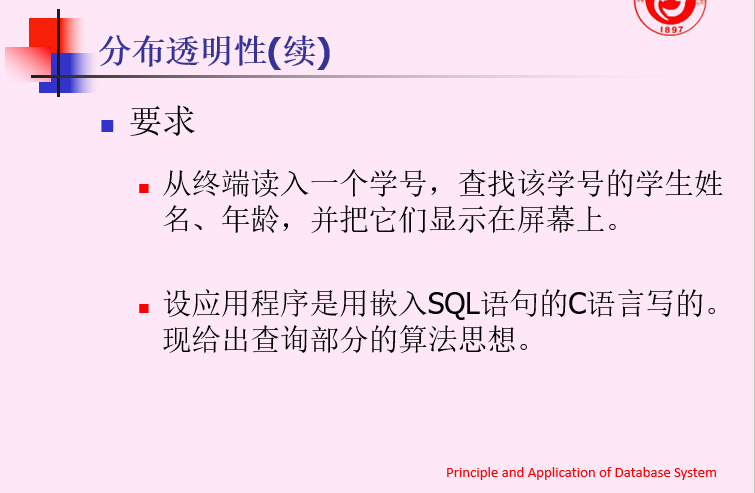
where Sno = A

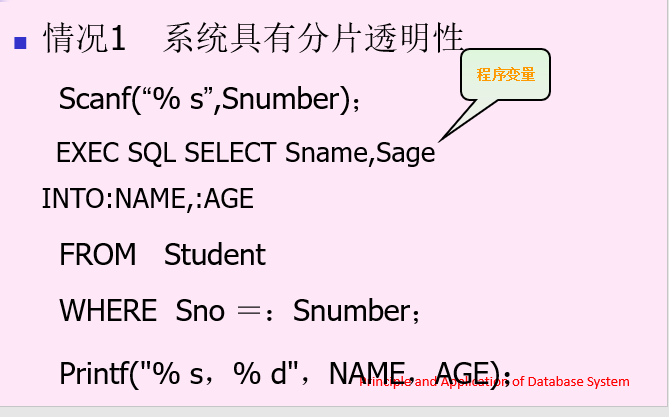
3\\ Select Sname, Sage

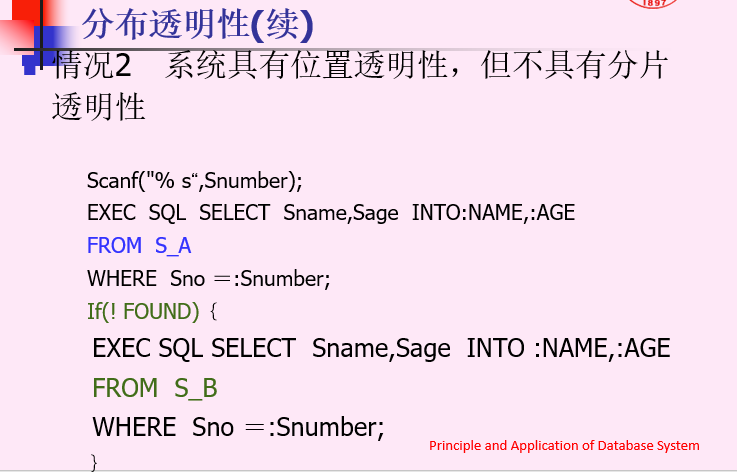
from S\_A, S\_B

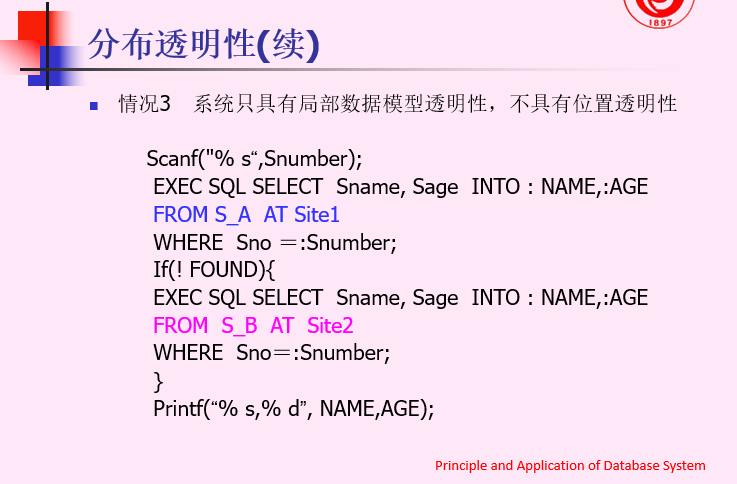
where Sno = A











2014年分布式系统

2（第一版第一章习题）答：令k为服务器的数量，则有(1/4)k<0.01。即在最坏的情况下，这时所有的服务器都已关闭，发生这种情况的概率是1/100。因此k=4。

3（第一版第二章习题）答：它们通常不提供分布透明性，这意味着应用程序开发人员需要注意通信的实现，从而导致解决方案的可扩展性很差。分布式应用程序，例如基于套接字构建的分布式应用程序，将很难移植或者和其它应用程序交互。

4、（第一版第五章习题）答：当一个进程接受到ELECTION消息，它会检查消息是谁开始发送的，如果是它自己开始的（例如它的位置在列表的首位），它会把消息变成协调者消息并在正文里描述它。如果消息不是它开始的，它会加入其进程号并沿着环向前发送。然而，如果它更早地发送了自己的选举消息并发现了竞争者，它会将创始人的进程号与自己的相比较。如果其他进程拥有较小的号码，它会丢弃那个消息而不是传递消息。如果竞争者更大，那么消息将以平常的方式发送。这样，如果多个选举消息被开始发送，那么入口最大的消息将会幸存，其他消息将沿着路由被忽略掉。

5

6 如13年

2015年试题

2、（第三章第一题）答: 在单线程情况下，命中cache花了15ms，未命中cache花了90ms。加权平均值为2/3 X 15 + 1/3 X 90。这意味着请求花了40ms，服务器每秒可以完成25次。对于多线程，所有磁盘等待都是交迭的，所以每个请求花了15ms，服务器每秒可以处理66个请求。

3、page200

**以数据为中心的一致性模型**

**1.严格一致性（strict consistency）**

对于数据项x的任何读操作将返回最近一次对x进行写操作的结果所对应的值。

严格一致性是限制性最强的模型，但是在分布式系统中实现这种模型代价太大，所以在实际系统中运用有限。

**2.顺序一致性**

任何执行结果都是相同的，就好像所有进程对数据存储的读、写操作是按某种序列顺序执行的，并且每个进程的操作按照程序所制定的顺序出现在这个序列中。

也就是说，任何读、写操作的交叉都是可接受的，但是所有进程都看到相同的操作交叉。顺序一致性由Lamport（1979）在解决多处理器系统的共享存储器时首次提出的。

**3.因果一致性**

所有进程必须以相同的顺序看到具有潜在因果关系的写操作。不同机器上的进程可以以不同的顺序看到并发的写操作（Hutto和Ahamad 1990）。

假设P1和P2是有因果关系的两个进程，例如P2的写操作信赖于P1的写操作，那么P1和P2对x的修改顺序，在P3和P4看来一定是一样的。但如果P1和P2没有关系，那么P1和P2对x的修改顺序，在P3和P4看来可以是不一样的。

下表列出一些一致性模型，以限制性逐渐降低的顺序排列。

|  |  |
| --- | --- |
| 一致性 | 描述 |
| 严格 | 所有共享访问按绝对时间排序 |
| 线性化 | 所有进程以相同的顺序看到所有的共享访问。而且，访问是根据（非唯一的）全局时间戮排序的 |
| 顺序 | 所有进程以相同顺序看到所有的共享访问。访问不是按时间排序的 |
| 因果 | 所有的进程以相同的顺序看到困果相关的共享访问 |
| FIFO | 所有进程以不同的进程提出写操作的顺序相互看到写操作。来自不同的进程的写操作可以不必总是以同样的顺序出现 |

另一种不同的方式是引入显式的同步变量，下表是这样的一致性模型的总结。

|  |  |
| --- | --- |
| 一致性 | 描述 |
| 弱 | 只有在执行一次同步后，共享数据才被认为是一致的 |
| 释放 | 退出临界区时，使共享数据成为一致 |
| 入口 | 进入临界区时，使属于一个临界区的共享数据成为一致 |

**以客户为中心的一致性模型**

**1.最终一致性**

最终一致性指的是在一段时间内没有数据更新操作的话，那么所有的副本将逐渐成为一致的。例如OpenStack Swift就是采用这种模型。以一次写多次读的情况下，这种模型可以工作得比较好。

**2.单调读**

如果一个进程读取数据项x的值，那么该进程对x执行的任何后续读操作将总是得到第一次读取的那个值或更新的值

**3.单调写**

一个进程对数据x执行的写操作必须在该进程对x执行任何后续写操作之前完成。

**4.写后读**

一个进程对数据x执行一次写操作的结果总是会被该进程对x执行的后续读操作看见。

**5.读后写**

同一个进程对数据项x执行的读操作之后的写操作，保证发生在与x读取值相同或比之更新的值上。

4、熟悉的分布式计算模型和特点

