《分布式系统》复习

一、填空题

1. 访问透明性是指对不同数据表示形式以及资源访问方式的隐藏。而位置透明是用户无法判别资源在系统中的物理位置。
2. 迁移透明性是指分布式系统中的资源移动不会影响该资源的访问方式。而复制透明是指对同一个资源存在多个副本的隐藏。
3. 一个开放的分布式系统就是根据一系列准则来提供服务，这些准则描述了所提供服务的语法和语义。
4. 集群计算系统一个突出的特征是它的同构性；它提供了最大限度的分布式透明性。可用于单个程序在多台计算机上并行地运行。
5. 分布式系统是一组自治的计算机集合，通过通信网络相互连接，实现资源共享和协作，而呈现给用户的是单个完整的计算机系统。
6. 客户/服务器结构的应用程序通常划分为三层，它们是：用户接口层、处理层和数据层。
7. 两个旅行社甲和乙为旅客到某航空公司订飞机票，形成互斥的资源是 飞机票。
8. 某游戏公司欲开发一个大型多人即时战略游戏，游戏设计的目标之一是能够支持玩家自行创建战役地图，定义游戏对象的行为和之间的关系。针对该目标，公司应该采用解释器架构风格最为合适。
9. 在消息队列系统中，队列由队列管理器来管理，它与发送或接收消息的应用程序直接交互。
10. 所谓幂等，简单地说，就是对接口的多次调用所产生的结果和调用一次是一致的。
11. 业务系统实现幂等的通用方式一般是排重表校验。
12. 对提供者而言，云计算可以三种部署模式，即共有云、私有云和混合云。
13. 在云计算技术中，中间件 位于服务和服务器集群之间，提供管理和服务即云计算体系结构中的管理系统。
14. 分布式系统体系结构样式很多，其最重要的有：分层体系结构；基于对象的体系结构、以数据为中心的体系结构以及基于事件的体系结构等四类。
15. 客户/服务器结构的应用程序通常划分为三层，它们是：用户接口层、处理层和数据层。
16. 分布式软件体系结构主要分集中式、非集中式和各种混合形式三大类。其非集中式体系结构又分为结构化的点对点、非结构化的点对点、超级对等体三种。
17. TCP/IP 体系结构的传输层上定义的两个传输协议为传输控制协议(TCP)和用户数据报协议(UDP)。
18. 客户/服务器结构的应用程序通常划分为三层，它们是：用户接口层、处理层和数据层。
19. 在逻辑时钟算法中，Lamport 定义了一个称作“先发生”的关系，表达式 a→b 表示 a 在 b 之前发生。先发生关系是一个传递关系。
20. 对于域名：test.com，DNS 服务器查找顺序是（先查找.com 域，再查找 test 主机 ）。
21. 实现软件自适应的基本技术分为要点分离、计算映像和基于组件的设计三种类型。
22. 分布式的自主系统指的是自我管理、自我恢复、自我配置和自我优化等各种自适应性。
23. 一个线程独立地执行它自己的程序代码。线程系统一般只维护用来让多个线程共享 CPU 所必需的最少量信息。
24. 任何一个分布式系统都无法同时满足一致性（Consistency）、可用性（Availability）和分

区容错性（Partition tolerance），最多只能同时满足两项。

25. 在服务器的组织结构中，迭代服务器是自己处理请求，将响应返回给客户；而并发服务器将请求传递给某个独立线程或其他进程来处理。
26. 服务器集群在逻辑上由三层组成，第一层是逻辑交换机；第二层是应用/计算服务；第三层是文件/数据库系统。
27. 有两种实现线程包的基本方法：一是可以构造一个完全在用户模式下执行的线程；二是由内核来掌管线程并进行调度。
28. 在代码迁移的框架结构中，进程包含三个段，它们是代码段、资源段和执行段三个段。
29. 进程对资源的绑定有三种类型：一是按标识符绑定；二是按值绑定；三是按类型绑定。而三种类型的资源对机器的绑定是未连接资源、附着连接资源和紧固连接资源。
30. 中间件是一种应用程序，它在逻辑上位于应用层中，但在其中包含有多种通用协议，这些协议代表各自所在的层，独立于其他更加特别的应用。
31. 在 RPC 操作中，客户存根的功能是将得到的参数打包成消息，然后将消息发送给服务器存根。
32. 虚拟化可采用两种方法，一是构建一个运行时系统，提供一套抽象指令集来执行程序。二是提供虚拟机监视器。
33. 中间件是一种应用程序，它在逻辑上位于应用层中，但在其中包含有多种通用协议，这些协议代表各自所在的层，独立于其他更加特别的应用。
34. 在远程对象调用中，远程接口使每个远程方法都具有方法签名。
35. 所有 DCE 的底层编程模型都是客户-服务器模型。而 DCE 本身的一部分是由分布式文件服务、目录服务、安全服务以及分布式时间服务等构成的。
36. 在面向消息的通信中，通常分为面向消息的瞬时通信和持久通信两种机制。
37. 在面向消息的瞬时通信中，通常采用套接字接口和消息传递接口。
38. 在面向持久的通信中，消息队列系统为持久异步通信提供多种支持。它提供消息的中介存储能力。
39. 在消息队列系统中，队列由队列管理器来管理，它与发送或接收消息的应用程序直接交互。
40. 应用层多播的基本思想是结点组织成一个覆盖网络，然后用它来传播信息给其成员。一个重要的因素是网络路由器不在组成员中。
41. 在覆盖网络构建时，主要有两种方法，一种是结点本身直接组织成树；另一种是结点组织成一个网状网络。
42. 分布式系统中，有三种不同的命名系统，它分别是无层次命名；结构化命名和基于属性的命名。
43. 在无层次命名中，通常有广播和多播、转发指针、基于宿主位置、分布式散列表、分层结构等方法实现实体定位。
44. 基于属性的命名系统实现的方式有两种。一种是分层实现，使得目录项集合形成了分层的目录信息树。而另一种是非集中式实现，它是采用映射到分布式散列表的方式。
45. 一次将所有的消息以相同的顺序传送给每个接收的多播操作称为全序多播。Lamport 时间戳可以用于以完全分布式的方式实现。
46. 向量时钟能捕获因果关系。创建向量时钟是让每个进程 Pi 维护一个向量 VCi 来完成。
47. 互斥集中式算法的优点是易于实现、很公平、保证了顺序一致性。而缺点是协作者是单个故障点，如果它崩溃了，整个系统可能瘫痪。
48. 在分布式锁中，惊群效应指的是，在有多个请求等待获取锁的时候，一旦占有锁的线程释放之后，如果所有等待的方都同时被唤醒，尝试抢占锁。但是这样的情况会造成比较

大的开销，那么在实现分布式锁的时候，应该尽量避免惊群效应的产生。

49. 分布式系统中，传统的选举算法有两种，一是欺负选举算法；二是环选举算法。
50. 分布式互斥算法的优点是不会发生死锁与饿死现象，也不存在单个故障点。
51. 拜占庭问题（Byzantine Problem）讨论的是允许存在少数节点作恶（消息可能被伪造）场景下的一致性达成问题。拜占庭容错（Byzantine Fault Tolerant，BFT）算法讨论的是在拜占庭情况下对系统如何达成共识。
52. 服务器集群在逻辑上由三层组成，第一层是逻辑交换机；第二层是应用/计算服务；第三层是文件/数据库系统。
53. IDL 编译器的输出包括三个文件，它们是头文件、客户存根和服务器存根。
54. 在面向消息的通信中，通常分为面向消息的瞬时通信和持久通信两种机制。
55. 分布式系统中的互斥算法有四种类型，一是集中式算法、二是非集中式算法、三是分布式算法、四是令牌环算法。
56. 网络协议有三要素组成，时序是对事件实现顺序的详细说明；语义是指需要发出何种控制信息以及要完成的动作与作出的响应；语法是指用户数据与控制信息的结构与格式。
57. 在名称解析的实现中，通常采用两种方法，一是迭代名称解析；二是递归名称解析。
58. 用户第一次请求时，负载均衡器将用户的请求转发到了 A 服务器上，如果负载均衡器设置了粘性 Session 的话，那么用户以后的每次请求都会转发到 A 服务器上。
59. 在逻辑时钟算法中，Lamport 定义了一个称作“先发生”的关系，表达式 a→b 表示 a 在 b 之前发生。先发生关系是一个传递关系。
60. 在原子多播里，消息排序通常有 4 种不同的排序方法，它们分别是：不排序的多播、FIFO 顺序的多播、按因果关系排序多播和全序多播。
61. 虚拟化的典型类型:基础设施虚拟化、系统虚拟化、软件虚拟化。
62. 虚拟化的目的：对象脱离原有环境、在计算机上被表示、通过计算机控制按需获取。
63. 分布式加密系统通常有三种类型，一是对称加密系统（DES）；二是公钥加密系统（RSA）、三是散列函数（MDS）系统。
64. 云体系结构的开发有如下三层：基础设施层、平台层和应用程序层。这三个开发层使用云中分配的经虚拟化和标准化的硬件与软件资源实现。
65. 部署基础设施层来支持 IaaS 服务。基础设施层是为支持 PaaS 服务构建云平台层的基础。平台层是为 SaaS 应用而实现应用层的基础。
66. SaaS 是一种基于互联网提供软件服务的应用模式。
67. 对提供者而言，云计算可以三种部署模式，即 共有云、 、 私有云和混合云 。
68. 分布式 是公有云计算基础架构的基石。
69. 当前，几乎所有的知名 IT 提供商、互联网提供商，甚至电信运营商都在向云计算进军，都在提供相关的云服务。但归纳起来，当前云提供者可以分为三大类，即 SaaS 提供商、和 PaaS、IaaS 提供商。

二、非填空题

1. 中间件在分布式系统中扮演着什么角色？

答：中间件主要是为了增强分布式系统的透明性（这正是网络操作系统所缺乏的），换言之，中间件的目标是分布式系统的单系统视图。

2. 什么是开放的分布式系统？开放性带来哪些好处？

答：开放的分布式系统根据明确定义的规则来提供服务。开放系统能够很容易地与其它系统

协作，同时也允许应用移植到同一个系统的不同实现中。

3. 分布式系统的类型。

(1)分布式计算系统(分为群集计算系统和网格计算系统)

(2)分布式信息系统(分为事务处理系统和企业应用集成)

(3)分布式普适系统(如家庭系统、电子健保系统、传感器网络)

4. 计算机系统的硬件异构性、软件异构性主要表现在哪几方面？

计算机系统的硬件异构性主要有三个方面的表现，即：

①计算机的指令系统不同。这意味着一种机器上的程序模块不能在另一种不兼容的机器上执行，很显然，一种机器上的可执行代码程序不能在另一种不兼容的机器上执行。

②数据表示方法不同。例如不同类型的计算机虽然都是按字节编址的，但是高字节和低字节的规定可能恰好相反。浮点数的表示方法也常常不一样。

③机器的配置不同。尽管机器的类型可能相同，其硬件配置也可以互不兼容。

计算机系统的软件异构性包括操作系统异构性和程序设计语言异构性。

操作系统异构性的三个主要表现方面为：

①操作系统所提供的功能可能大不相同。例如，不同的操作系统至少提供了不同的命令集。

②操作系统所提供的系统调用在语法、语义和功能方面也不相同。

③文件系统不同。

程序设计语言的异构性表现在不同的程序设计语言用不同方法在文件中存储数据。

5. 对服务器进程中的线程数目进行限制有意义吗？

答：有。原因有两个：

线程需要内存来设置他们的私有堆栈。因此，线程太多可能导致消耗过多的存储器。

更严重的情况是，对于一个操作系统，独立的线程是以无序的方式在运行。在虚拟存储器系统中，构建一个相对稳定的工作环境可能比较困难，从而导致许多的页错误和过多的 I/O 操作，结果可能导致系统性能的下降。

6. 请描述在客户和服务器进程间使用套接字时如何进行无连接通信？

答：同时在客户端和服务器上创建一个套接字，但只有服务器套接字绑定到本地终结点。然后，服务器可以随后做一个阻塞读取调用用以等待接收从任何客户端传入的数据。同样，在创建套接字后，客户端仅仅做一个阻塞调用以向服务器写入数据. 这是没有必要关闭连接的。

7. 移动计算和普适计算的区别

普适计算可包括移动计算，但普适计算不是移动计算，前者更强调环境驱动性。从技术上来说，这就要求普适计算对环境信息具有高度的可感知性，人机交互更自然化，设备和网络的自动配置和自适应能力更强，所以普适计算的研究涵盖中间件、移动计算、人机交互、嵌入式技术、传感器、网络技术等领域。普适计算要解决的问题包括：扩展性、异构性、不同构件的集成、上下文感知和不可见性(Invisibility)。其中不可见性对普适计算来说是至关重要的，因为它要求系统无需用户干预或只需要最少干预，也就是要求系统具有自动和动态的配置机制。

8. 简述分布式系统中异构性的解决方案。

- *中间件---是指一个软件层，它提供了一个编程抽象，同时屏蔽了底层网络、硬件、操作系统和编程语言的异构性。* *softbus*
  - *公共对象请求代理（Common Object Request Broker, CORBA）*
  - *JAVA 远程调用方法（Remote Method Invocation， RMI）*
  - *大多数中间件基于互联网协议实现，这些协议屏蔽了底层网络的差异，但是中间件要解决操作系统和硬件的不同。*

- *移动代码*---是指从能从一台计算机发送到另一台计算机，并在目的计算机上运行的代码（例如：Java applet）
- *虚拟机*方法提供了一种代码可以在任何计算机上运行的方法：某种语言的编译器生成一台虚拟机代码而不是某种硬件代码，虚拟机通过解释的方式来执行它。

9. 简要描绘全局唯一标识符的一个有效实现

答：这些标识符可以在以下方式中可以局部产生:将产生标识符的机器所在的网络地址，附加上当地时间, 沿用一个伪随机数. 虽然，在理论上, 另一台机器也很有可能产生相同的数字, 这种机会微乎其微。

2、由于分布计算系统包含多个（可能是不同种类的）分散的、自治的处理资源，要想把它们组织成一个整体，最有效地完成一个共同的任务，做到这一点比起传统的集中式的单机系统要困难得多，需要解决很多新问题。这些问题主要表现在哪些方面？

参考答案：

①资源的多重性带来的问题。由于处理资源的多重性，分布计算系统可能产生的差错类型和次数都比集中式单机系统多。最明显的一个例子是部分失效问题：系统中某一个处理资源出现故障而其他计算机尚不知道，但单机系统任何一部分出现故障时将停止整个计算。另一个例子是多副本信息一致性问题。可见，资源多重性使得差错处理和恢复问题变得很复杂。资源多重性还给系统资源管理带来新的困难。

②资源的分散性带来的问题。在分布计算系统中，系统资源在地理上是分散的。由于进程之间的通信采用的是报文传递的方式进行的，通信将产生不可预测的、有时是巨大的延迟，特别是在远程网络所组成的分布计算系统中更是这样。例如使用卫星通信会产生 270 毫秒的延迟。在分布计算系统中，系统的状态信息分布在各个分散的节点上。分布式的状态信息和不可预知的报文延迟使得系统的控制和同步问题变得很复杂，要想及时地、完整地搜集到系统各方面的信息是很困难的，从而使处理机进行最佳调度相当困难。

③系统的异构性带来的问题。在异构性分布计算系统中，由于各种不同资源（特别是计算机和网络）的数据表示和编码、控制方式等均不相同，这样一来就产生了翻译、命名、保护和共享等新问题。

由于上述原因，分布计算系统的研制，特别是软件的验证、调试、测量和维护问题变得很复杂。这些正是分布计算系统研制者要解决的主要问题。

10.

11. 分布式系统具有透明性时，系统有什么优点。

系统具有透明性时有以下一些优点：

①使软件的研制变得容易，因为访问资源的方法只有一种，软件的功能与其位置无关。

②系统的某些资源变动时不影响或较少影响应用软件。

③系统的资源冗余（硬件冗余和软件冗余）使操作更可靠，可用性更好。透明性使得在实现这种冗余的时候，各种冗余资源的互相替换变得容易。

④在资源操作方面，当把一个操作从一个地方移到若干地方时没有什么影响。

12. 什么是幂等操作？说说幂等的含义以及其真正的价值。判断集合并、文件记录的插入、缓冲区的读取、HTTP GET 方法、POST 和 PUT 是否是幂等的。

某个操作可以重复多次而无害出。也就是重复执行多次与执行一次的效果是相当的。

幂等性是分布式系统设计中十分重要的概念，假设有一个从账户取钱的 API：

boolean withdraw(account_id, amount)

它实现的功能是从 account_id 对应的账户中扣除 amount 数额的钱；如果扣除成功则返回 true，账户余额减少 amount；如果扣除失败则返回 false，账户余额不变。

因为生产环境的网络状况，用户操作的情况非常复杂，比如 API 的请求已经被服务器

端正确处理，但返回结果由于网络原因导致客户端无法得知处理结果，可能会使用户认为上一次操作失败了，然后刷新页面，
这就导致了 withdraw 被调用两次，账户也被多扣了一次钱。
这个问题的解决方案一是采用分布式事务，但是分布式事务一方面架构太重量级，容易被绑在特定的中间件上，不利于异构系统的集成；另一方面分布式事务虽然能保证事务的 ACID 性质，而但却无法提供性能和
可用性的保证。
另一种更轻量级的解决方案是幂等设计。我们可以通过一些技巧把 withdraw API 变成幂等的，比如：
int create_ticket()
boolean idempotent_withdraw(ticket_id, account_id, amount)
create_ticket 实现的功能是获取一个服务器端生成的唯一的 ticket_id，它用于标识后续的操作。
idempotent_withdraw 和 withdraw 的区别在于关联了一个 ticket_id，一个 ticket_id 表示的操作至多只会被处理一次，每次调用都将返回第一次调用时的处理结果。这样，idempotent_withdraw 就符合幂等性了，客户端就可以放心地多次调用。

HTTP GET 方法用于获取资源，不应有副作用，所以是幂等的。POST 方法不具备幂等性。PUT 方法具有幂等性。

13. 什么是间接通信？简述间接通信的基本策略。

通过第三个实体，允许在发送者和接收者之间的深度解耦合，即发送者不需要知道正在发送给谁（空间解耦合），发送者和接收者不需要同时存在（时间解耦合）

a) 组通信
   组通信涉及消息传递给若干接收者，是支持一对多通信的多方通信泛型
b) 发布—订阅系统
   大量生产者（或发布者）为大量的消费者（订阅者）发布他们感兴趣的信息项
   关键特征：中间服务---用于确保由生产者生成的信息被路由到需要这个信息的消费者
c) 消息队列
   提供点对点服务
   生产者发送消息到指定队列，消费者能从队列中接收消息，或者被通知队列有消息到达
d) 元组空间
   进程可以把任意的结构化数据项（元组）放到一个持久元组空间，其他进程可以指定感兴趣的模式，从而可以在元组空间读取或者删除元组。
e) 分布式共享内存（Distributed Shared Memory，DSM）
   系统提供一种抽象，用于支持在不同共享物理内存的进程之间共享数据

14. 发布—订阅系统的几种已定义的常见模式：

a) 基于渠道
   发布者发布事件到命名的渠道，订阅者订阅其中一个已命名的渠道，并接收所有发送到那个渠道的事件
b) 基于主题
   假设每个通知中包含多个域，其中一个域表示主题。订阅是根据感兴趣的主题来定

义的。

c) 基于内容

基于内容的方法是基于主题方法的一般化，它允许订阅表达式具有一个事件通知上的多个域。基于内容的过滤器是事件属性值的约束组合定义的查询。

d) 基于类型

订阅根据事件类型来定义，匹配根据给定的过滤器的类型或者子类型来定义。

e) 基于概念的订阅模型

过滤器可以根据事件的语义和语法进行表述

15. 列出出版订购系统中事件路由的常见技术？

泛洪（Flooding）

最简单的方法是基于泛洪，也就是向网络中的所有节点发送事件通知，在订阅者端执行适当的匹配

另外一种方案，用泛洪发送订阅到所有可能的发布者，在发布端执行匹配，匹配成功的事件通过点对点通信被直接发送到相关的订阅者

过滤

代理网络中采用过滤（filtering）是很多方法所采用的原则，这种方法被称为*基于过滤的路由*

代理通过一个有路径到达有效订阅者的网络转发通知，实现机制：

通过先向潜在的发布者传播订阅信息

然后在每个代理上存储相关状态

每个结点必须维护

邻居列表（该列表包含了该节点在代理网络中所有相连接的邻居）

订阅列表（该列表包含了由该节点为之服务的所有直接连接的订阅者）

路由表（维护该路径上的邻居和有效订阅列表）

广告

上述纯基于过滤的方法会由于订阅的传播而产生大量的网络流，订阅本质上采用了泛洪的方法向所有可能的发布者推送

在广告系统中，通过与订阅传播类似的方式，向订阅者传播广告，这样流量负担可以减少

汇聚（rendezvous）

控制订阅传播的方法

将所有可能的事件集合看做一个事件空间，将事件空间的责任划分到网络中的代理集合上

汇聚结点是负责一个给定的事件空间的子集的代理节点

16. 用组通信方式时，举例说明消息顺序的重要性，并说明解决方法说明。

答：要使组通信易于理解和使用，有两种性质是不可缺少的，首先是原子广播原语，它确保了一条消息要么被所有组内成员收到，要么没有一个成员能收到。其次是消息的顺序。例如：有四台机器每台机器有一个进程，进程 1、2、3、4 属于同一个进程组，进程 0 与进程 4 同时想给该组发送一条消息，当两个进程竞相访问 LAN 时，在网络中消息传送的顺序是无法确定的，可能是 0->1, 4->0,4->1,4->3,0->3,0->4。这样进程 1 先收到 0 再收到 4，进程 3 先收到进程 4 在收到 0，则 1 与 3 之间可能会出现不一致。

解决方法：1）全局时间顺序，保证立即发送所有消息并让他们保持发送顺序，该方法能将消息精确的按照发送顺序传递到目的地。2）一致时间顺序，若有两条消息 A 和 B，以很少的时间间隔发送，系统先取其中一个作为第一个发送给所有组内成员，然后再取下一个发送给组内成员，这种方法保证组内成员按照统一的顺序收到了消息，但是这个顺序可能并不是发送消息的顺序。

17. 在深度为 k 的分层定位服务中，当移动实体改变它的位置时，最多需要更新多少条位置记录？

答：改变位置可以看作是插入和删除操作的组合。插入操作要求至少 k+1 条记录变动，同样地，删除操作也要求改变 k+1 个记录，根的记录被这两个操作分享，导致 2k+1 条记录被更新。

18. 分层定位服务中的根结点可能是一个潜在的瓶颈。如何能有效地避免这个问题？

答：一项重要观察发现我们只使用随机位的字符串作为标识符，这样，我们很容易就划分标识符空间并且为每一个部分分配一个独立根结点。划分的根结点以及通路将遍布网络。

19. 要使用 Lamport 时间戳实现全序多播，是不是每个消息都必须要被严格地确认？

答：不需要，任何类型的消息，只要它的时间戳大于所接收到的消息的时间戳，就可以被加入消息队列，使用 Lamport 时间戳实现全序多播。

20. IBM MQSeries 以及许多其他消息队列系统中的路由表是人工配置的。描述一种自动完成配置工作的简单方法。

答：最简单的是现实使用一个集中的组件，该组件维护消息队列系统的拓扑结构。它使用一种已知的路由算法来计算各个队列管理器之间的最佳路由，然后为每一个队列管理器生成路由表，这些表可以由各个管理器分别下载。这种方法适合于队列管理器相对较少但是特别分散的消息队列系统。

21. 许多分布式算法需要使用协调进程。简单讨论一下，这样的算法实际上可以在什么程度上被看作为分布式的？

答：在集中式的算法中，常常是固定的进程充当协调者。分布来源于其他进程在不同的机器上运行的事实。在分布式算法中，没有固定的协调者，协调者从组成部分算法的进程中选出。事实是协调者能使算法更具分布性。

22. 有三个进程 P1，P2 和 P3 并发工作。进程 P1 需用资源 S3 和 S1；进程 P2 需用资源 S1 和 S2；进程 P3 需用资源 S2 和 S3。回答：

(1)若对资源分配不加限制，会发生什么情况?为什么?

(2)为保证进程正确工作，应采用怎样的资源分配策略?为什么?

(1)多个进程动态地共享系统的资源可能会产生死锁现象。死锁的产生，必须同时满足四个条件，第一个是互斥条件，即一个资源每次只能由一个进程占用；第二个为等待条件，即一个进程请求资源不能满足时，它必须等待，但它仍继续保持已得到的所有其它资源；第三个是非出让条件，任何一个进程不能抢占另一个进程已经获得且未释放的资源；第四个为循环等待条件，系统中存在若干个循环等待的进程，即其中每一个进程分别等待它前一个进程所持有的资源。防止死锁的机构只须确保上述四个条件之一不出现，则系统就不会发生死锁。

只要资源分配策略能保证进程不出现循环等待，则系统就不会发生死锁。

(2) 银行家算法分配资源的原则是：系统掌握每个进程对资源的最大需求量，当进程要求申请资源时，系统就测试该进程尚需资源的最大量，如果系统中现存的资源数大于或等于该进程尚需的最大量时，则就满足进程的当前申请。这样可以保证至少有一个进程可能得到全部资源而执行到结束，然后归还它所占用的全部资源供其它进程使用。银行家算法破坏了产生死锁的第四个条件，即不可能产生循环等待，从而可以避免死锁的发生。

防止进程发生循环等待的另一种资源分配策略是按序分配算法，其基本思想如下：把系统中所有的资源排一个顺序，例如系统共有 m 个资源，用 ri 表示第 i 个资源，那么这 m 个资源是：

r1, r2, r3 ……, rm

规定任何进程不得在占用资源 ri（1<i≤m）后再申请 rj（j<i≤m），或者说，如果里程需要资源 rj，那么它必须在申请 ri 之前申请（j<i）。可以证明，按这种策略分配资源时破坏了循环等待

条件，故能防止发生死锁

23. 在分布式系统中，为什么需要代码迁移？代码迁移可以分为 Sender-initiated 和 Receiver-initiated，解释其中的含义，并举例说明。

如果把进程由负载较重的机器上转移到负载较轻的机器上去，就可以提升系统的整体性能。 Sender-initiated 是在发送者启动的迁移，代码当前驻留在哪台机器上或者正在哪台机器上执行，就由该机器来启动迁移。通过因特网向 Web 数据库服务器发送搜索程序以在该服务器上进行查询就是发送者启动迁移的一个例子。 Receiver-initiated 是在接收者启动的迁移，代码迁移的主动权掌握在目标机器手中。Java 小程序就是接收者启动的一个例子。

24. 代码迁移的动机有哪些？

代码迁移指的是将程序（或执行中的程序）传递到其它计算机。(基本思想：把进程由负载较重的机器上转移到负载较轻的机器上去，就可以提升系统的整体性能)

迁移动机：

（1）实现负载均衡：将进程从负载重的系统迁移到负载轻的系统，从而改善整体性能。（2）改善通信性能：交互密集的进程可迁移到同一个节点执行以减少通信开销，当进程要处理的数据量较大时，最好将进程迁移到数据所在的节点。

（3）可用性：需长期运行的进程可能因为当前运行机器要关闭而需要迁移。（4）使用特殊功能：可以充分利用特定节点上独有的硬件或软件功能。（5）灵活性：客户首先获取必需的软件，然后调用服务器。

25. 简述三模冗余的基本思想，并举例说明三模冗余能否处理 Byzabtine 故障。

答：三模冗余是使用物理冗余来提供容错的技术，是使用主动复制方法的容错。在电子电路中有设备 A、B、C，然后每个设备复制三次，结果就是每级电路都设置了三个表决器，每个表决器有三个输入和一个输出，若两个或者三个输入相同，输出则等于输入，若三个输入各不相同，输出就是不定值，这种设计就是 TMR。

若处理机是 Byzabtine 类型的，出错的处理机仍然工作并发出错误的随机的应答，那么至少需要 2k+1 个处理机才能达到 k 级容错。最坏情况下 k 个失效的处理机偶然（甚至有意）地产生相同的应答，然而剩下的 k+1 个未出错的处理机也将产生相同的应答，因此客户机可以根据大多数的应答得到正确结果。

三模冗余在每组中有一个部件出现 Byzabtine 故障时可以处理，而一组中有两个甚至三个同时出现 Byzabtine 故障则不能处理。

26. 说明 RPC 的主要思想及 RPC 调用的主要步骤。

答：主要思想是允许程序去调用位于其他机器上的过程。当位于机器 A 的一个进程调用机器 B 上的某个过程时，机器 A 上的过程被挂起，被调用的过程在机器 B 上执行。调用者讲消息放在参数表中传送给被调用者，结果作为过程的返回值返回给调用者。消息的传送与 I/O 操作对于编程人员是不可见的。

主要步骤如下：1）客户过程以普通方式调用相应的客户存根；2）客户存根建立消息并激活内核陷阱；3）内核将消息发送到远程内核；4）远程内核将消息发送到服务器存根；5）服务器存根取出消息中的参数后调用服务器过程；6）服务器完成工作后将结果返回至服务器存根；7）服务器存根将它们打包并激活内核陷阱；8）远程内核将消息发送会客户内核；9）客户内核将消息提交给客户存根；10）客户存根从消息中取出结果返回给客户。

27. 在 RPC 调用时，如果服务器或客户机崩溃了，各有哪些解决方法。

答：如果是服务器崩溃了，用户无法区分服务器是在执行前还是执行后崩溃，解决方案如下：1）至少一次语义，指等待服务器重新启动，然后重发请求。这种方法要求不断重试直至客户收到应答消息。它保证 RPC 至少执行一次。2）之多一次语义，指立即放弃并报告失效。它确保 RPC 至多执行一次，但也可能根本没有执行；3）不作保证；4）精确一次语义；

如果是客户机崩溃了，存在孤儿问题（客户已发送请求，在应答到来之前崩溃了，此时已经激活服务器中的过程并获得结果，但是没有客户在等待结果）解决方案如下：1）根除，在客户存根发送 RPC 消息前先

做日志（用来恢复崩溃），系统重新启动后，检查日志，发现孤儿存在并将其杀死；2）再生，把时间分成有序的纪元，当客户端重启时，向所有机器广播一个消息通知一个新纪元的到来，并结束所有的远程计算；3）温和再生，服务器接收到新纪元广播时，检查自己是否有远程计算，只有那些找不到所有者的远程计算终止。4）过期，每个 RPC 都分配一个标准时间 T 来完成任务，如果超时没有完成则显示分配一个数额。

28. 在 RPC 中，如果客户机在发送请求后在服务器应答消息到来之前崩溃了，将会发生什么问题？如何解决？

解答：发生现象：客户机在发送请求后在服务器应答消息到来之前崩溃，其已经激活了服务器的相应计算，而客户没有等待它的结果，将遗留“计算孤儿”。

清除“孤儿”方法：

a) 根绝(extermination)法:客户存根发送 RPC 前在日志文件中记录将要执行的 RPC，若客户重启则依据日志作准确清除远程计算。
b) 再生（reincarnation） 法：划分时间为序号纪元（时间戳）， 客户重起则广播新纪元开始，所有远程计算被终止。
c) 温和再生（gentle reincarnation）法：改进“再生”法，由服务器检查远程计算有无调用者，若无则远程计算被终止。
d) 过期（expinration） 法：每个 rpc 执行前给定时间段 T，rpc 到期未完成的必须再申请新的 T 。服务器将清除没有再申请新的 T 的 rpc。

29. 一个影响 RPC 执行时间的问题是消息的拷贝问题，试说明在那些环节需要拷贝，并说明减少拷贝次数的方法。

答：需要消息拷贝的环节：在发送端，消息从客户存根拷贝到客户内核缓冲区，再从客户内核缓冲区拷到客户接口芯片缓冲区（网卡），然后消息被拷贝到接收端的服务器接口芯片缓冲区，之后拷贝到服务器内核缓冲区，最后到达服务器存根（共 5 次）拷贝。此外，有时还需要拷贝参数数组。

减少拷贝次数的方法：分散-集中方法（汇集发），具有分散-集中能力的网络芯片可以减少拷贝次数，他通过拼接 2 个或者多个内存缓冲区来组装报文。在发送端，由客户内核缓冲区生成报文消息头。由客户存根生成报文消息体，当发送时，由网络芯片组装报文。同样地，接收端将接收来的报文分解成消息体和消息头，并放入相应的缓冲区。

30. RPC 通信与通常的自定义协议的 Socket 通信有哪些区别和联系

RPC 是建立在 Socket 之上的，相对于 Socket，RPC 实现了以下功能，

首先，通讯方面，主要是通过在客户端和服务器之间建立 TCP 连接，远程过程调用的所有交换的数据都在这个连接里传输。连接可以是按需连接，调用结束后就断掉，也可以是长连接，多个远程过程调用共享同一个连接。

第二，寻址方面，比如 A 服务器上的应用怎么告诉底层的 RPC 框架，如何连接到 B 服务器（如主机或 IP 地址）以及特定的端口，方法的名称名称是什么，这样才能完成调用。比如基于 Web 服务协议栈的 RPC，就要提供一个 endpoint URI，或者是从 UDDI 服务上查找。如果是 RMI 调用的话，还需要一个 RMI Registry 来注册服务的地址。

第三，当 A 服务器上的应用发起远程过程调用时，方法的参数需要通过底层的网络协议如 TCP 传递到 B 服务器，由于网络协议是基于二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化（Serialize）或编组（marshal），通过寻址和传输将序列化的二进制发送给 B 服务器。

第四，B 服务器收到请求后，需要对参数进行反序列化（序列化的逆操作），恢复为内存中的表达方式，然后找到对应的方法（寻址的一部分）进行本地调用，然后得到返回值。

第五，返回值还要发送回服务器 A 上的应用，也要经过序列化的方式发送，服务器 A 接到后，再反序列化，恢复为内存中的表达方式，交给 A 服务器上的应用去处理

RPC 方法的基本原则是--以模块调用的简单性忽略了通讯的具体细节,以便程序员不用关心 C/S 之间的通讯

协议，集中精力实现业务过程. 这就决定了 RPC 生成的通讯不可能对每种应用都有恰当的处理方法. 与 Socket 方法相比,传输相同的有效数据,RPC 占用更多的网络带宽和系统资源。

31. 给出一种用来让客户端绑定到暂时远程对象的对象应用的实现示例。

答：使用 Java 实现的类如下：

public class Object_reference {

InetAddress server3address; // network address of object's server int server3endpoint; // endpoint to whic h server is listening int object3identifier; // identifier for this object

URL client3code; // (remote) file containing client-side stub byte[] init3data; // possible additional initialization data }

Object_reference 类至少需要包含对象所属的服务器的传输层地址。在具体实现中，使用了一个 URL 来标识包含了所有必需的客户端代码文件，用一个字节数组来保存进一步初始化后的代码。另一种实现可以直接保存客户端代码而不是一个 URL。这种方法将代理对象作为引用传递，Java RMI 采用了这种做法。

32. 简述远程方法调用（Remote Method Invocation，RMI）的基本通信原理。

答:：远程方法调用(RMI)的基本通信原理:

客户端与服务器端内在通过套接字通信

服务器端

1、创建远程服务对象

2、接收请求、执行并返回结果(Skeleton)

1）解码（读取）远程方法的参数； 2）调用实际远程对象实现的方法；

3）将结果（返回值或异常）返回给调用程序。

客户端

1、建立与服务器的连接

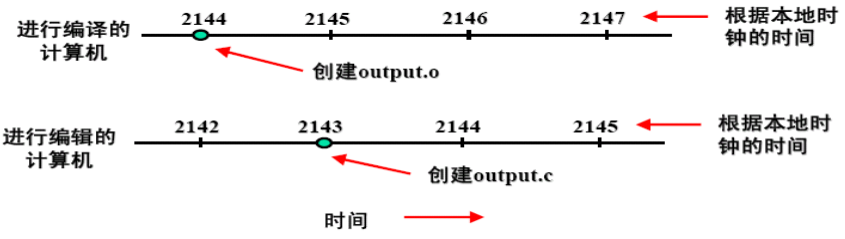
2、发送请求、接收返回结果(Stub)

1）初始化连接； 2）编码并发送参数； 3）等待方法调用结果；

4）解码（读取）返回值或返回的异常； 5）将值返回给调用程序。

33. 试举例说明没有统一时钟的分布式系统会发生什么问题？

解答：当每台机器有它自己的时钟时，一个发生于另一事件之后的事件可能会被标记为一个比另一个事件更早的时间。例：



34. 什么是 RPC？试简述 RPC 的执行步骤。

答：RPC 是 remote procedure call（远程过程调用）的简称。RPC 思想是使远程的过程调用就像在本地的过程一样，调用者不应该意识到此调用的过程是在其他机器上实行的。

RPC 的执行步骤：
(1) 客户过程以普通方式调用相应的客户存根；
(2) 客户存根建立消息，打包并激活内核陷阱；
(3) 内核将消息发送到远程内核；
(4) 远程内核将消息发送到服务器存根；
(5) 服务器存根将消息解包，取出其中参数后调用服务器过程；
(6) 服务器完成工作或将结果返回服务器存根；
(7) 服务器存根将它打包并激活内核陷阱；
(8) 远程内核将消息发送至客户内核；
(9) 客户内核将消息交给客户存根；
(10) 客户存根将消息解包，从中取出结果返回给客户；

35. 分布式系统中的体系结构样式有那几种？并简述之。

1)分层体系结构：组件组成了不同的层，其中 Li 层的组件可以调用下面的层 Li-1。其中的关键因素是，其控制系统是从一层到别一层的：请求是从上往下，而请求结果是从下向上。

2）基于对象的体系结构：是一种很松散的组织结构。基本上每一个对象都对应一个组件，这些组件是通过过程调用机制来连接的。分层和面向对象的体系结构仍然是大型软件系统最重要的样式。

3)以数据为中心的体系结构，其发展思想：进程通信需要通过一个公用（被动或主动的）仓库。可以分成两个关键的部分:展现当前状态的中央数据结构和一个对数据进行操作的独立组件的集合

4）基于事件的体系结构，进程基本上是通过事件的传播来通信的，事件传播还可以有选择地携带数据。基本思想是：进程发布事件，然后，中间件将确保那些订阅了这些事件的进程才接收它们，优点是进程是松散耦合的。

36. 分布式令牌环算法存在令牌丢失的问题，如果令牌丢失，会导致算法失败，请将该算法改进一下，使该算法既能检测到令牌丢失，也能进行补救。

令牌环的故障处理功能主要体现在对令牌和数据帧的维护上。令牌本身就是比特串，绕环传递过程中也可能受干扰而出错，以至造成环路上无令牌循环的差错；另外，当某站点发送数据帧后，由于故障而无法将所发的数据帧从网上撤消时，又会造成网上数据帧持续循环的差错。令牌丢失和数据帧无法撤消，是环网上最严重的两种差错，可以通过在环路上指定一个站点作为主动令牌管理站，以此来解决这些问题。

主动令牌管理站通过一种超过机制来检测令牌丢失的情况，该超时值比最长的帧为完全遍历环路所需的时间还要长一些。如果在该时段内没有检测到令牌，便认为令牌已经丢失，管理站将清除环路上的数据碎片，并发出一个令牌。

为了检测到一个持续循环的数据帧，管理站在经过的任何一个数据帧上置其监控位为 1，如果管理站检测到一个经过的数据帧的监控拉的已经置为 1，便知道有某个站未能清除自己发出的数据帧，管理站将清除环路的残余数据，并发出一个令牌。

37. 简述分布式系统设计所面临的问题及遇到的挑战。

1)：难以合理设计分配策略,在集中式系统中，所有的资源都由系统系统管理和分配，但在

分布式系统中，资源属于局部工作站或个人计算机，所以调度的灵活性不如集中式系统，资源的物理分布可能与服务器的分布不匹配,某些资源可能空闲,而另外一些资源可能超载。

2)：部分失效问题：由于分布式系统通常是由若干部分组成的，各个部分由于各种各样的原因可能发生故障，如硬件故障。如果一个分布式系统不对这些故障对这些问题进行有效的处理，系统某个组成部分的故障可能导致整个系统的瘫痪。

3）性能和可靠性过分依赖于网络：由于分布式系统是建立在网络之上的，而网络本身是不可靠的，可能经常发生故障，网络故障可能导致整个系统的终止；另外，网络超负荷会导致性能下降，增加系统的响应时间。

4）缺乏统一控制：一个分布式系统的控制通常是一个典型的分散式控制，没有统一的中心控制。因此，分布式系统通常需要相应的同步机制来协调系统中各个部分的工作；

5）安全保密性问题：为了获得可扩展性，分布式系统中的许多软件接口都提供给用户，这样的开放结构对于开发人员非常有价值，担同时也为破坏者打开了方便之门

38. 由于分布计算系统包含多个（可能是不同种类的）分散的、自治的处理资源，要想把它们组织成一个整体，最有效地完成一个共同的任务，做到这一点比起传统的集中式的单机系统要困难得多，需要解决很多新问题。这些问题主要表现在哪些方面？

①资源的多重性带来的问题。由于处理资源的多重性，分布计算系统可能产生的差错类型和次数都比集中式单机系统多。最明显的一个例子是部分失效问题：系统中某一个处理资源出现故障而其他计算机尚不知道，但单机系统任何一部分出现故障时将停止整个计算。另一个例子是多副本信息一致性问题。可见,资源多重性使得差错处理和恢复问题变得很复杂。资源多重性还给系统资源管理带来新的困难。

②资源的分散性带来的问题。在分布计算系统中，系统资源在地理上是分散的。由于进程之间的通信采用的是报文传递的方式进行的,通信将产生不可预测的、有时是巨大的延迟，特别是在远程网络所组成的分布计算系统中更是这样。例如使用卫星通信会产生 270 毫秒的延迟。在分布计算系统中，系统的状态信息分布在各个分散的节点上。分布式的状态信息和不可预知的报文延迟使得系统的控制和同步问题变得很复杂，要想及时地、完整地搜集到系统各方面的信息是很困难的，从而使处理机进行最佳调度相当困难。

③系统的异构性带来的问题。在异构性分布计算系统中，由于各种不同资源（特别是计算机和网络）的数据表示和编码、控制方式等均不相同，这样一来就产生了翻译、命名、保护和共享等新问题。

由于上述原因，分布计算系统的研制，特别是软件的验证、调试、测量和维护问题变得很复杂。这些正是分布计算系统研制者要解决的主要问题。

39. 挑战：设计与实现一个对用户来说是透明的且具有容错能力的分布式系统。

挑战：1. Heterogeneity 异构性，包括网络、硬件、操作系统、编程语言、Implementation 组件，<中间件的作用就是隐藏这些异构，并提供一致的计算模式（模块）>，<虚拟机，编译器成虚拟机使用的 code>;2. Openness 开放性:提供 Services, Syntax, Semantics，合适的接口定义需要兼顾完整性和中立性，同时互操作性和便携性也是重要的，分布式系统需要 flexible,即易于配置且把策略和结构分开来获得 flexible;3. Security 安全性：安全性包括有效性，机密性，完整性三个要素，拒绝服务攻击和移动代码安全是两个安全方面的挑战。4. Scalability 可扩展性（可测性），即在资源和用户增加的时候保持效率，size、geographically、administratively 的 scaleble。要求分散（分布式）算法，没有

机器有系统状态的完整信息，机器做决定仅仅是取决于本地信息，一个机器的错误不会毁掉整个算法，没有统一时序。5. Failure Handling 错误处理。6. Concurrency 协力，一致（并发？） 7. Transparency 透明度

40. 说明客户/服务器模式的主要思想，并说明在采用了阻塞的、有缓存的、可靠的发送和接收原语的情况下，系统是如何工作的。

主要思想：构造一种操作系统，它由一组协同进程组成，这组进程称作服务器。为用户提供服务的进程称作客户，客户和服务器都运行在相同的微内核中。 进程调用 send 原语，它指定了目的地以及发送到该目的地的缓冲区数据。消息被发送时，发送的进程被阻塞。直到消息传送完毕，其后的指令才能继续执行。之后对接收信息感兴趣的进程可以让内核为之建立一个邮箱，并指定一个地址以便于寻找网络信包。所有具有该地址的输入消息被放入邮箱中，调用 receive 时只要从邮箱中取出一条消息，邮箱为空时阻塞。可靠的原语有三种：①重新定义非可靠原语。②要求接收机器的内核给发送机器的内核发送一个确认消息。只有当收到这个确认消息后，发送内核释放用户进程。③客户机在发送消息阻塞后，服务器的内核不发送确认消息，而是将应答作为确认消息。

41. 对于接收消息 Receive 原语，为什么需要缓存，缓存的作用是什么？

答：如果不适用缓存，服务器接收来的消息会被丢弃或者存在诸如服务器需要存储和管理早到来的消息这样的问题。缓存的作用就是用来统一管理消息的：它定义了一种叫邮箱的数据结构，接收客户端请求的进程通知内核创建邮箱存储消息，并且指定了访问地址。当 Receive 原语调用是，系统内核就会提取消息并知道如何处理它。

42. 一个最完备的分布式体系由以下模块组成。请说明各模块的功能？

①分布式处理系统必须有能力在短时间内动态地组合成面向不同服务对象的系统。对用户来说系统是透明的，用户只需指定系统干什么而不必指出哪个部件可以提供这一服务。系统各组成部分是自主的，但不是无政府状态，而是遵循某个主计划由高级操作系统进行协调工作。在一个计算机网中有多台主机不一定都是分布式处理。如果这样的系统不具备动态组合及任务再指派的能力，那么它们仍然是集中式处理。

②分布式查询可以访问来自多种异类数据源的数据，而这些数据可存储在相同或不同的计算机上。

③分布式数据库系统由分布于多个计算机结点上的若干个数据库系统组成,它提供有效的存取手段来操纵这些结点上的子数据库。分布式数据库在使用上可视为一个完整的数据库,而实际上它是分布在地理分散的各个结点上。当然,分布在各个结点上的子数据库在逻辑上是相关的。

④分布式缓存支持一些基本配置：重复（replicated）、分配（partitioned）和分层（tiered）。重复（Replication）用于提高缓存数据的可用性。在这种情况下，数据将重复缓存在分布式系统的多台成员机器上，这样只要有一个成员发生故障，其他成员便可以继续处理

该数据的提供。另一方面，分配（Partitioning）是一种用于实现高可伸缩性的技巧。通过将数据分配存放在许多机器上，内存缓存的大小加随着机器的增加而呈线性增长。结合分配和重复这两种机制创建出的缓存可同时具备大容量和高可伸缩的特性。
⑤分布式文件系统具有执行远程文件存取的能力,并以透明方式对分布在网络上的文件进行管理和存取。
⑥ 分布式网络通信能够连接跨越了多台计算机的应用程序各节点。
⑦分布式监控管理可以有效避免最上层服务器因顾及不暇而出现管理疏漏的现象。
⑧用于编写运行于分布式计算机系统上的分布式程序。一个分布式程序由若干个可以独立执行的程序模块组成,它们分布于一个分布式处理系统的多台计算机上被同时执行。它与集中式的程序设计语言相比有三个特点：分布性、通信性和稳健性。
⑨分布式系统的执行存在着许多非稳定性的因素。分布式算法起到分布性和并发性的作用，这一点不同于集中式算法。

43. 为什么分布式系统需要用到 ID 生成系统

在复杂分布式系统中，往往需要对大量的数据和消息进行唯一标识。如在美团点评的金融、支付、餐饮、酒店、猫眼电影等产品的系统中，数据日渐增长，对数据库的分库分表后需要有一个唯一 ID 来标识一条数据或消息，数据库的自增 ID 显然不能满足需求；特别一点的如订单、骑手、优惠券也都需要有唯一 ID 做标识。此时一个能够生成全局唯一 ID 的系统是非常必要的。

概括下来，业务系统对 ID 号的要求有哪些呢？

ID 生成系统的需求

1.全局唯一性：不能出现重复的 ID，最基本的要求。

2.趋势递增：MySQL InnoDB 引擎使用的是聚集索引，由于多数 RDBMS 使用 B-tree 的数据结构来存储索引数据，在主键的选择上面我们应尽量使用有序的主键保证写入性能。

3.单调递增：保证下一个 ID 一定大于上一个 ID。

4.信息安全：如果 ID 是连续递增的，恶意用户就可以很容易的窥见订单号的规则，从而猜出下一个订单号，如果是竞争对手，就可以直接知道我们一天的订单量。所以在某些场景下，需要 ID 无规则。

第 3、4 两个需求是互斥的，无法同时满足。

同时，在大型分布式网站架构中，除了需要满足 ID 生成自身的需求外，还需要 ID 生成系统可用性极高。

44. 应用哪些技术可以使得一个分布式系统具有可伸缩性？

答：实现分布式可伸缩性，基本的三种技术为：

1、减少通信延迟，即使用异步通信方式，使得发送方发送请求后不必阻塞以等待答复，而是处理其他本地任务。

2、分层，即将一个组件分解为几个小层。一个好的例子是 DNS 域名系统，它将域名分为三层，均衡了系统负载。

3、复制冗余，它能使得资源更容易就近获取，并且它能使资源分布于整个系统，均衡了负载。

45. 请介绍一下分布式两阶段提交协议？

阶段一：开始向事务涉及到的全部资源发送提交前信息。此时，事务涉及到的资源还有最后一次机会来异常结束事务。如果任意一个资源决定异常结束事务，则整个事务取消，不会进行资源的更新。否则，事务将正常执行，除非发生灾难性的失败。为了防止会发生灾难性的失败，所有资源的更新都会写入到日志中。这些日志是永久性的，因此，这些日志会幸免遇难并且在失败之后可以重新对所有资源进行更新。 * 阶段二：只在阶段一没有异常结束的时候才会发生。此时，所有能被定位和单独控制的资源管理器都将开始执行真正的数据更新。 在分布式事

务两阶段提交协议中，有一个主事务管理器负责充当分布式事务协调器的角色。事务协调器负责整个事务并使之与网络中的其他事务管理器协同工作。 为了实现分布式事务，必须使用一种协议在分布式事务的各个参与者之间传递事务上下文信息，IIOP 便是这种协议。这就要求不同开发商开发的事务参与者必须支持一种标准协议，才能实现分布式的事务。

46. 描述可能发生在因特网上的几类主要的安全威胁(对进程的威胁、对信道的威胁、拒绝服务)可能的发生情况。

Threats to processes: without authentication of principals and servers, many threats exist. An enemy could access other user's files or mailboxes, or set up 'spoof' servers. E.g. a server could be set up to 'spoof' a bank's service and receive details of user's financial transactions.

Threats to communication channels: IP spoofing - sending requests to servers with a false source address, man-in-the-middle attacks.

Denial of service: flooding a publicly-available service with irrelevant messages.

47. 给出服务器实现的概要，说明创建新线程执行每个客户请求的服务器如何使用操作 getRequest、sendReply 操作。说明服务器如何从请求消息中复制请求 requestID 到应答消息以及它如何获得客户 IP 地址和端口。

```
class Server{
  private int serverPort = 8888;
  public static int messageLength = 1000;
  DatagramSocket mySocket;

  public Server(){
     mySocket = new DatagramSocket(serverPort);
     while(true){
          byte [] request = getRequest();
          Worker w = new Worker(request);
     }


  }
  public byte [] getRequest(){
     //as above}
  public void sendReply(byte[]reply, InetAddress clientHost, int clientPort){
     // as above}
}
class Worker extends Thread {
     InetAddress clientHost;
     int clientPort;
     int requestId;
     byte [] request;
     public Worker(request){
     // extract fields of message into instance variables
   }
```

```
public void run(){
   try{
        req = request.unmarshal();
        byte [] args = req.getArgs();
        //unmarshall args, execute operation,
        // get results marshalled as array of bytes in result
        RequestMessage rm = new RequestMessage( requestId, result);
        reply = rm.marshal();
        sendReply(reply, clientHost, clientPort );
   }catch {... }
  }
}
```

48. 讨论在 TCP/IP 连接之上实现请求一应答协议时可以获得的调用语义，该调用语义要确保数据按发送顺序到达，既不丢失也不重复。考虑导致连接中断的所有条件。

A process is informed that a connection is broken:

- when one of the processes exits or closes the connection.
- when the network is congested or fails altogether

Therefore a client process cannot distinguish between network failure and failure of the server.

Provided that the connection continues to exist, no messages are lost, therefore, every request will receive a corresponding reply, in which case the client knows that the method was executed exactly once.

However, if the server process crashes, the client will be informed that the connection is broken and the client will know that the method was executed either once (if the server crashed after executing it) or not at all (if the server crashed before executing it).

But, if the network fails the client will also be informed that the connection is broken. This may have happened either during the transmission of the request message or during the transmission of the reply message. As before the method was executed either once or not at all.

Therefore we have at-most-once call semantics.

49. 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？

分为以下 4 个步骤：

1.当发送一个 URL 请求时，不管这个 URL 是 Web 页面的 URL 还是 Web 页面上每个资源的 URL，浏览器都会开启一个线程来处理这个请求，同时在远程 DNS 服务器上启动一个 DNS 查询。这能使浏览器获得请求对应的 IP 地址。

2.浏览器与远程 Web 服务器通过 TCP 三次握手协商来建立一个 TCP/IP 连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。

3.一旦 TCP/IP 连接建立，浏览器会通过该连接向远程服务器发送 HTTP 的 GET 请求。远程服务器找到资源并使用 HTTP 响应返回该资源，值为 200 的 HTTP 响应状态表示一个正确的响应。

4.此时，Web 服务器提供资源服务，客户端开始下载资源。

50. 为什么要用 ajax

Ajax 应用程序的优势在于：
1. 通过异步模式，提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax 引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。

51. 简述 ajax 的过程

1. 创建 XMLHttpRequest 对象,也就是创建一个异步调用对象
2. 创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息
3. 设置响应 HTTP 请求状态变化的函数
4. 发送 HTTP 请求
5. 获取异步调用返回的数据
6. 使用 JavaScript 和 DOM 实现局部刷新

52. 客户向服务器发出远程过程调用。客户花 5ms 时间计算每个请求的参数，服务器花 10ms 时间处理每个请求。木地操作系统处理每次发送和接收操作的时间是 0.5ms,网络传递每个请求或者应答消息的时间是 3ms.编码或者解码每个消息花 0.5ms 时间。

计算下列两种条件下客户创建和返回消息所花费的时间:

(1)如果它是单线程的

(2)如果它有两个线程，这两个线程能在一个处理器土并发地处理请求

你可以忽略上下文切换时间。如果客户和服务器处理器是线程化的，需要异步 RPC 吗?

i) time per call = calc. args + marshal args + OS send time + message transmission +

OS receive time + unmarshall args + execute server procedure

+ marshall results + OS send time + message transmission +

OS receive time + unmarshal args

= 5 + 4*marshal/unmarshal + 4*OS send/receive + 2*message transmission + execute server procedure

= 5+ 4*0.5 + 4*0.5 + +2*3 + 10 ms = 5+2+2+6+10 =25ms.

Time for two calls = 50 ms.

ii) threaded calls:

client does calc. args + marshal args + OS send time (call 1) = 5+.5=.5 = 6

then calc args + marshal args + OS send time (call 2) = 6

= 12 ms then waits for reply from first call

server gets first call after

message transmission + OS receive time + unmarshal args = 6+ 3+.5+.5

= 10 ms, takes 10+1 to execute, marshal, send at 21 ms

server receives 2nd call before this, but works on it after 21 ms taking

10+1, sends it at 32 ms from start

client receives it 3+1 = 4 ms later i.e. at 36 ms

(message transmission + OS receive time + unmarshal args) later

Time for 2 calls = 36 ms.

53. 为什么使用分布式锁？

使用分布式锁的目的，无外乎就是保证同一时间只有一个客户端可以对共享资源进行操作。

但是 Martin 指出，根据锁的用途还可以细分为以下两类

(1)允许多个客户端操作共享资源

这种情况下，对共享资源的操作一定是幂等性操作，无论你操作多少次都不会出现不同结果。在这里使用锁，无外乎就是为了避免重复操作共享资源从而提高效率。

(2)只允许一个客户端操作共享资源

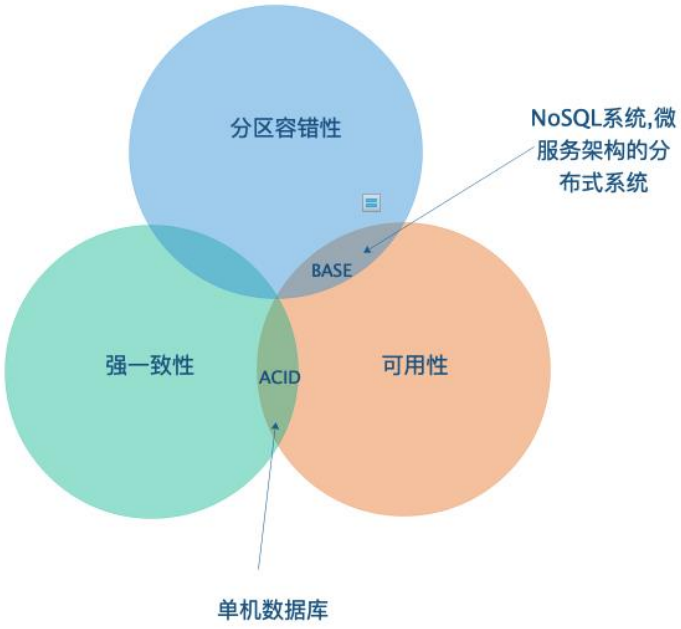
这种情况下，对共享资源的操作一般是非幂等性操作。在这种情况下，如果出现多个客户端操作共享资源，就可能意味着数据不一致，数据丢失。

54. CAP,BASE 以及 ACID 的关系

CAP 描述了对于一个分布式系统而言重要的三要素:数据一致性,可用性,分区容错性之间的制约关系,当你选择了其中的两个时,就不得不对剩下的一个做一定程度的牺牲。BASE 和 ACID 都可以看做是对 CAP 三要素进行取舍后的某种特殊情况

- BASE 强调可用性和分区容错性,放弃强一致性,这是大部分分布式系统的选择,比如 NoSQL 系统,微服务架构下的分布式系统
- ACID 是单机数据的事务特性,因为不是分布式系统无需考虑分区容错,故而是选择了可用性和强一致性后的结果。

它们之间的关系如下所示



55. 简述虚拟化的实现层次。

虚拟化技术通过在同一个硬件主机上多路复用虚拟机的方式来共享昂贵的硬件资源，虚拟化的基本思想是分离软硬件以产生更好的系统性能

引入虚拟化后，不同用户应用程序由自身的操作系统（即客户操作系统）管理，并且那些客户操作系统可以独立于主机操作系统同时运行在同一个硬件上，这通常是通过新添加一个称为虚拟化层的软件来完成，该虚拟化层称为 hypervisor 或虚拟机监视器（Virtual Machine Monitor，VMM）

- 指令集体系结构级：代码解释和动态二进制翻译

- 硬件抽象级：虚拟化一个计算机硬件资源
- 操作系统级：在单一物理服务器上创建隔离的容器和操作系统实例
- 库支持级：库接口的虚拟化
- 应用程序级：进程级虚拟化、高级语言（High Level Language，HLL）虚拟机

56. 虚拟机的两种架构的本质差别表现在那些方面？

虚拟机有两种架构：寄居架构（Hosted Architecture）和裸金属架构 ("Bare Metal" Architecture)。所谓寄居架构就是在已安装的操作系统之上安装和运行虚拟化程序 VMM，然后再利用 VMM 创建和管理虚拟机，它依赖于主机操作系统对设备的支持和物理资源的管理; 而裸金属架构就是直接在硬件上面安装虚拟化软件，即将 VMM 安装在物理服务器上，再在其上安装操作系统和应用，依赖虚拟层内核和服务器控制台进行管理。

普遍认为，裸金属架构比寄居架构高，因为裸金属架构是直接运行在硬件上的。一般寄生型适用于桌面系统，裸金属适用于服务器系统。

57. 简述什么是 SOAP，以及 SOAP 与 XML 之间的关系。

SOAP（Simple Object Access Protocol）即简单对象访问协议，是一个简单的用在 Web 上交换结构信息的 XML 协议，没有定义应用语义和传输语义，它以一种基于 XML 且与平台无关的 Web 编程方式改进了 Internet 的互操作性。SOAP 消息传递协议使用 HTTP 承载消息，而使用 XML 格式化消息。

SOAP 包括四个部分：

1，信封；2，数据的编码规则；3，RPC 调用规范；4，SOAP 绑定。 SOAP 与 XML 之间的关系是：

1，SOAP 是在 XML 基础上定义的，所有的 SOAP 消息都使用 XML 消息格式来编码，XML 是 SOAP 的底层技术规范；

2，对于 SOAP 中的简单类型，SOAP 采用了在 XML Schema 规范的数据类型部分定义的内嵌数据类型中所有类型，包括这些类型的值和词汇空间的定义；；

3，SOAP 完全继承了 XML 的开放性和描述可扩展性。

4，SOAP 还具有 XML 的其他一些优点，如可广泛应用于 web 的任何地方，使得编程更加简单，便于阅读等等。

58. 选举算法中 Bully 算法的思想

选举算法：选择一个进程作为协调者、发起者或其他特殊角色，一般选择进程号最大的进程（假设每个进程都知道其他进程的进程号，但不知道是否还在运行）它的目的是保证在选举之行后，所有进程都认可被选举的进程。

Bully 算法：

当进程 P 注意到需要选举一个进程作协调者时：

(1) 向所有进程号比它高的进程发 ELECTION 消息

(2) 如果得不到任何进程的响应，进程 P 获胜，成为协调者

(3)如果有进程号比它高的进程响应，该进程接管选举过程，进程 P 任务完成

(4)当其他进程都放弃，只剩一个进程时，该进程成为协调者

(5)一个以前被中止的进程恢复后也有选举权

59. 在分布式系统中，许多算法都需要一个进程充当协调者，因此需要协调者选举算法。试说明霸道算法的主要思想，并说明在 8 个进程的情况下号码为 3 的进程发现协调者崩溃后的选举过程。

霸道算法：当一个进程发现协调者不再响应请求时，它发起选举。进程 P 选举过程如下：

①□ P 向所有号码比它大的进程发送选举消息

② 若无人响应，P 获胜成为协调者。

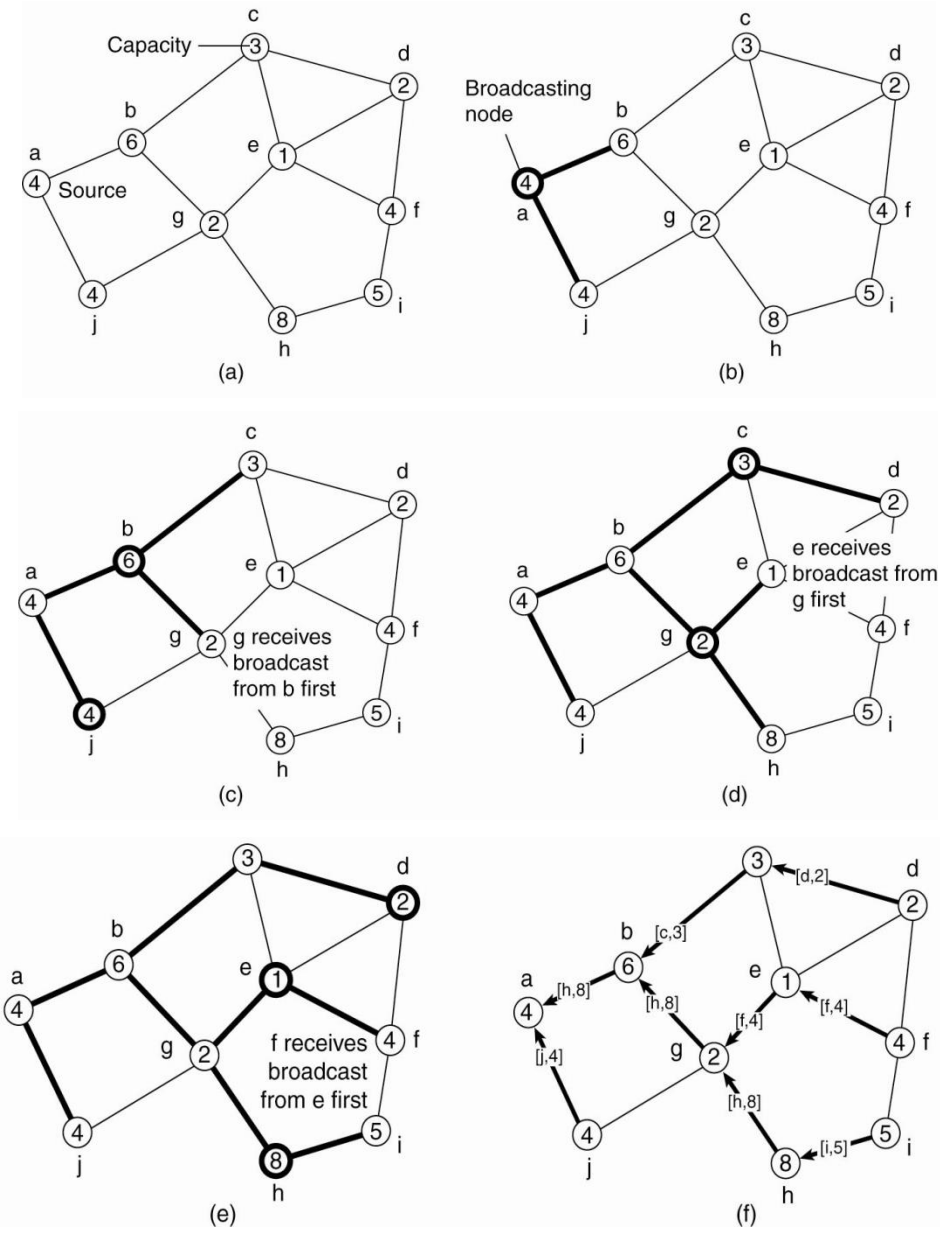
③ 若有号码比它大的进程响应，响应者接管，P 的工作完成。

假设 8 个进程为进程 0 到进程 7，原协调者为进程 7

进程 3 发现协调者崩溃，进程 3 主持选举，进程 4 进程 5 和进程 6 应答，通知进程 3 停止，进程 4 进程 5 进程 6 分别主持选举，进程 6 通知进程 5 和进程 4 停止，进程 6 获胜并通知所有进程。

60. 无线环境下的选举算法

- 不能保证其消息传送是可靠的以及网络拓扑结构不改变
- 源节点向其相邻节点发送 ELECTION 消息开始一个选举
- 当节点第一次收到 ELECTION 消息时，会将发送者作为其父节点，然后将 ELECTION 消息发给其相邻节点（父节点除外）。等到其他节点的确认消息都收到后，再向父节点确认（消息中包含资源容量）。
- 而从某个节点再次收到 ELECTION 消息时，只是确认。

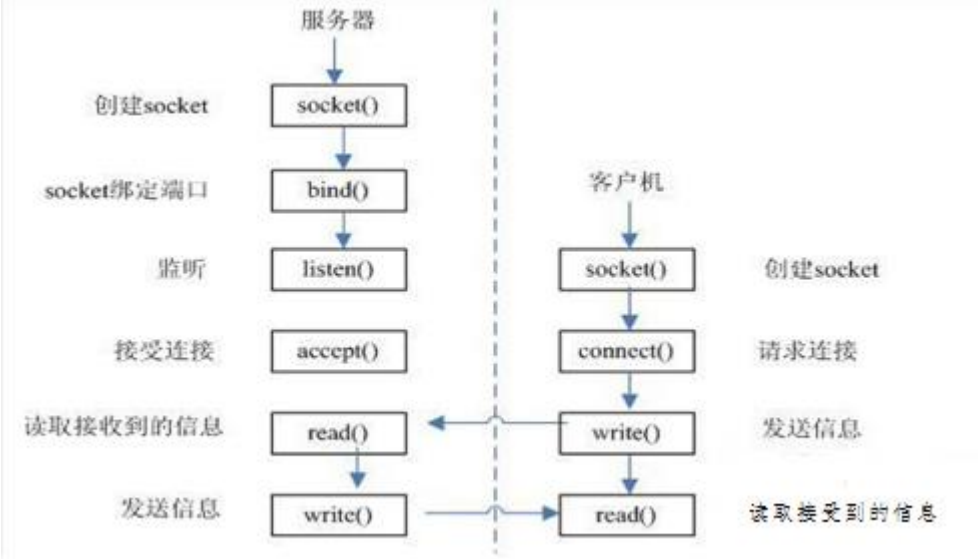


61. 描述一下客户和服务器之间使用套接字的无连接通信是如何进行的？

答：首先服务器和客户端都要创建一个套接字，并遵循 UDP 协议，服务器将其所在的 IP 地址以及一个端口号绑定到套接字，完成绑定后，服务器就能接收来自客户端的 UDP 数据包了。同样，客户端在创建套接字后，能够向服务器发送 UDP 包进行通信，通信过程中，服务器和客户端之间是不用建立连接的。

62. 试描述客户和服务器之间使用套接字的面向连接的通信是如何进行的？

答：为了实现服务器与客户机的通信，服务器和客户机都必须建立套接字。服务器与客户机的工作原理可以用下面的过程来描述。 (1)服务器先用 socket 函数来建立一个套接字，用这个套接字完成通信的监听。 (2)用 bind 函数来绑定一个端口号和 IP 地址。因为本地计算机可能有多个网址和 IP，每一个 IP 和端口有多个端口。需要指定一个 IP 和端口进行监听。 (3)服务器调用 listen 函数，使服务器的这个端口和 IP 处于监听状态，等待客户机的连接。(4)客户机用 socket 函数建立一个套接字，设定远程 IP 和端口。(5)客户机调用 connect 函数连接远程计算机指定的端口。 (6)服务器用 accept 函数来接受远程计算机的连接，建立起与客户机之间的通信。(7)建立连接以后，客户机用 write 函数向 socket 中写入数据。也可以用 read 函数读取服务器发送来的数据。 (8)服务器用 read 函数读取客户机发送来的数据，也可以用 write 函数来发送数据。(9)完成通信以后，用 close 函数关闭 socket 连接。 客户机与服务器建立面向连接的套接字进行通信，请求与响应过程可用图来表示。



63. 简述 TCP 和 UDP 协议在通信中的区别

TCP 是面向连接的可靠的协议，适用于传输大批量的文件，检查是否正常传输。而 UDP 是面向非连接的不可靠的协议，适用于传输一次性小批量的文件，不对传输数据报进行检查。

TCP 需要先建立连接才能通话；而 UDP 不需要，实时性要高点。

TCP 可以形象比喻为打电话的过程；UDP 可以比喻为发短信的过程。

TCP 不能发送广播和组播，只能单播；UDP 可以广播和组播。

64. 分布式系统的类型。

(1)分布式计算系统(分为群集计算系统和网格计算系统) (2)分布式信息系统(分为事务处理系统和企业应用集成) (3)分布式普适系统(如家庭系统、电子健保系统、传感器网络)

65. 简述面向服务的体系结构（SOA）。

面向服务的体系结构，是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独

立于实现服务的硬件平台、操作系统和编程语言。
3 个主要角色是：
1、服务请求者：服务请求者是一个应用程序、一个软件模块或需要一个服务的另一个服务。它发起对注册中心中的服务的查询，通过传输绑定服务，并且执行服务功能。服务请求者根据接口契约来执行服务。
2、服务提供者：服务提供者是一个可通过网络寻址的实体，它接受和执行来自请求者的请求。它将自己的服务和接口契约发布到服务注册中心，以便服务请求者可以发现和访问该服务。
3、服务注册中心：服务注册中心是服务发现的支持者。它包含一个可用服务的存储库，并允许感兴趣的服务请求者查找服务提供者接口。
3 个主要的服务是：
发布：为了使服务可访问．需要发布服务描述以使服务请求者可以发现和调用它。
查询：服务请求者定位服务．方法是查询服务注册中心来找到满足其标准的服务。
绑定和调用：在检索完服务描述之后，服务请求者继续根据服务描述中的信息来调用服务。

66. 选举算法中环算法的思想

不使用令牌，按进程号排序，每个进程都知道自己的后继者，当进程 P 注意到需要选举一个进程作协调者时：

(1) 创建一条包含该进程号的 ELECTION 消息，发给后继进程 (2)后继进程再将自己的进程号加入 ELECTION 消息，依次类推 (3)最后回到进程 P，它再发送一条 COORDINATOR 消息到环上，包含新选出的协调者进程（进程号最大者）和所有在线进程，这消息在循环一周后被删除，随后每个进程都恢复原来的工作。

67. 请求驱动式令牌传递方法中，若 pi 发出 request 消息后久未获得 Token，该怎么处理？若引 入时戳，该算法应做何修改？

答： 在请求驱动式令牌传递方法中，或 pi 发出的 request 消息后久未获得 Token，应该决定是站点故障还是 Token 丢失，需要有对应逻辑环重构方法和 Token 生成方法。 可以引入时时戳增加算法的强健性，具体如下: (1)当 request 消息后久未获得令牌，则向其它进程发询问消息；若其它进程无反对消息到达，则重新生成令牌，否则继续等待。 (2)若接收到询问消息的进程是令牌的持有者，或已发出一样 Request 消息，且自己 Request 消息的时戳先于询问进程 Request 消息的时戳，则立即发回一条反对消息。(3)令牌持有者传递令牌时，若发现接收者故障，需要调用逻辑环重构算法进行环重构，再重新选择接收者。

68. 设计一个能支持分布式无用单元回收和在本地与远程对象引用之间转化的远程对象表。给出一个例子，包括在不同地址上的几个远程对象和代理，以阐述该表的使用。说明当一个调用导致创建新代理时这个表的变化。说明当一个代理不能用时这个表的变化。

local reference	*remote reference*	*holders*

The table will have three columns containing the local reference and the remote reference of a remote object and the virtual machines that currently have proxies for that remote object. There will be one row in the table for each remote object exported at the site and one row for each proxy held at the site.

To illustrate its use, suppose that there are 3 sites with the following exported remote objects:

S1: A1, A2, A3 S2: B1, B2; S3: C1;

and that proxies for A1 are held at S2 and S3; a proxy for B1 is held at S3.

Then the tables hold the following information:.

at S1			*at S2*			*at S3*		
local	*remote*	*holders*	*local*	*remote*	*holders*	*local*	*remote*	*holders*
a1	*A1*	*S2, S3*	*b1*	*B1*	*S3*	*c1*	*C1*	
a2	*A2*		*b2*	*B2*		*a1*	*A1proxy*	
a3	*A3*		*a1*	*A1proxy*		*b1*	*B1proxy*	

Now suppose that C1(at S3) invokes a method in B1 causing it to return a reference to B2. The table at S2 adds the holder S3 to the entry for B2 and the table at S3 adds a new entry for the proxy of B2.

Suppose that the proxy for A1 at S3 becomes unreachable. S3 sends a message to S1 and the holder S3 is removed from A1. The proxy for A1 is removed from the table at S3.

69. Jones 正在运行一组进程 p1,p2,...，PN。每个进程 pi 包含一个变量 vi。她希望判定所有变量 v1,v2，...，vn 在执行中是否相等。

(1) Jones 的进程在同步系统中运行。她使用一个监控器进程判定变量是否相等。应用进程何时应该与监控器进程通信，它们的消息应该包含什么?

(2)解释语句:可能的(v1=v2=...vn) .Jones 如何能判定该语句在她的执行中成立。

(i) communicate new value when local variable v_i changes;

with this value send: current time of day *C(e)* and vector timestamp *V(e)* of the event of the change, *e*.

(ii) *possibly* (...): there is a consistent, potentially simultaneous global state in which the given predicate is true.

Monitor process takes potentially simultaneous events which correspond to a consistent state, and checks predicate $v_1 = v_2 = \ldots = v_N$.

Simultaneous: estimate simultaneity using bound on clock synchronization and upper limit on message propagation time, comparing values of *C* (see p. 415).

Consistent state: check vector timestamps of all pairs of potentially simultaneous events e_i, e_j: check $V(e_i)[i] \geq V(e_j)[i]$.

70. 修改用于互斥的中央服务器算法，使之能够处理任何客户〔在任何状态)的崩溃故障，假设服务器是正确的，并且有一个可靠的故障检测器。讨论这个系统是否能够容错。如果拥有令牌的客户被错误地怀疑为出了故障，会发生什么样的情况?

The server uses the reliable failure detector to determine whether any client has crashed. If the client has been granted the token then the server acts as if the client had returned the token. In case it subsequently receives the token from the client (which may have sent it before crashing), it ignores it.

The resultant system is not fault-tolerant. If a token-holding client crashed then the application-specific data protected by the critical section (whose consistency is at stake) may be in an unknown state at the point when another client starts to access it.

If a client that possesses the token is wrongly suspected to have failed then there is a danger that two processes will

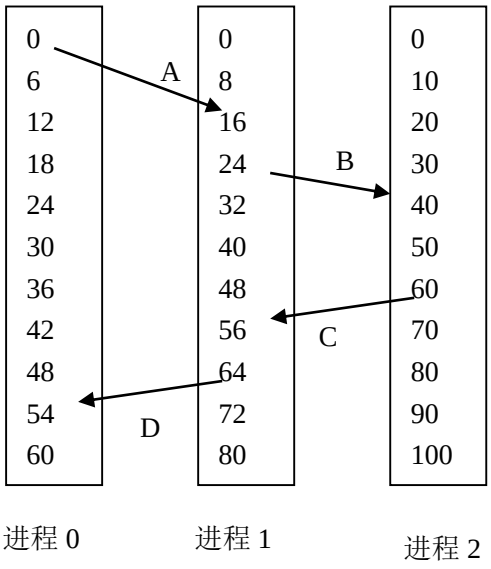
be allowed to execute in the critical section concurrently.

71. 实现分布式系统同步的复杂性表现在哪几个方面？说明先发生关系，并说明在 LAMPORT 算法中怎样给事件分配时间。

答：分布式算法有如下性质：1）相关信息分散在多台机器上；2）进程决策仅依赖于本地信息；3）系统中单点故障应避免；4）没有公用时钟和其他精确的全局时间资源存在。前三点说明在一处收集所有信息并对他们进程处理是不可接受的，左后一点说明在分布式系统获得时间上的一致并不是容易的。

LAMPORT 算法的解决方案是直接使用先发生关系，每条消息都携带发送者的时钟以指出其发送的时间，当消息到达时，接受者的时钟比消息发送者时钟小，就立即将自己的时钟调到比发送者的时间大 1 或更多的值，我们给出一种测量时间的方法，使得对每一事件 a，在所有进程中都认可给它一个时间值 C(a)，在给事件分配时间时要遵循一下规则：1）在同一进程中 a 发生在 b 之前则 C(a)<C(b)；2）若 a 和 b 分别代表发送消息和接收消息，则 C(a)<C(b)；3）对所有事件 a 和 b，C(a)≠C(b)

72. 有三个进程分别运行在不同的机器上，每个机器都有自己的时钟并以不同且不变的速率工作（进程 1 的时钟嘀嗒了 6 下时，进程 2 的时钟嘀嗒了 8 下，而进程 3 的时钟嘀嗒了 10 下）。举例说明进程之间消息传递中违反先发生关系的情况，并说明如何用 Lamport 方法解决。



73. How Web Services work?

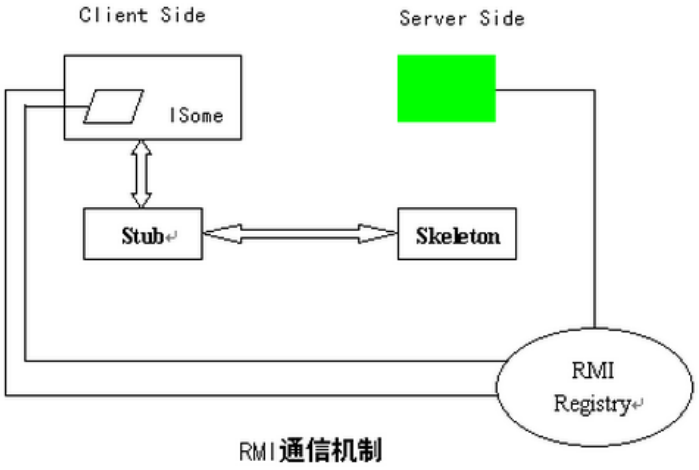
1 客户发现服务器

2 客户通过建立 TCP 连接来绑定服务器 3 客户建立 SOAP 请求

4 SOAP 路由器路由请求给合适的服务器 5 服务器把请求解包,然后处理,并返回结果 6 结果通过相反的路径返回到客户端

74. 简述远程方法调用（Remote Method Invocation，RMI）的基本通信原理。

答:：远程方法调用(RMI)的基本通信原理:



RMI通信机制

客户端与服务器端内在通过套接字通信

服务器端

1、创建远程服务对象

2、接收请求、执行并返回结果(Skeleton)

1）解码（读取）远程方法的参数； 2）调用实际远程对象实现的方法；

3）将结果（返回值或异常）返回给调用程序。

客户端

1、建立与服务器的连接

2、发送请求、接收返回结果(Stub)

1）初始化连接； 2）编码并发送参数； 3）等待方法调用结果；

4）解码（读取）返回值或返回的异常； 5）将值返回给调用程序。

75. 服务器虚拟化的关键技术？

①计算虚拟化

- CPU 虚拟化
- 计算负载的动态分配
- 能耗管理

②存储虚拟化

- 内存虚拟化
- 磁盘存储动态分配

③设备与 I/O 虚拟化

- 软件方式实现
- 统一、标准化的接口
- 操作指令转译

④实时迁移技术

- 将整个虚拟机的运行状态完整、快速地从原宿主机的硬件平台转移到新的宿主机硬件平台。

76. 举例说明 Lamport 等人提出的算法是如何解决 Byzantine 将军问题的。

答：Lamport 等人设计了一种递归算法可在特定条件下解决这一问题。例如：N = 4（有四个将军），M

= 1（其中有一个叛徒），对这样的参数，参数运行四步。

第一步，每个将军发送可靠的消息给其他所有的将军，声明自己真实的军队人数，忠诚的将军声明的是真值，叛徒则可能对其他每个将军都撒一个不同的谎。如图 a；

第二步，把第一步声明的结果组成向量形式，如图 b；

第三步，每个将军把图 b 中各自的向量传递给其他每一个将军，这里叛徒再一次撒谎，使用了 12 个新值。A—J。如图 c；

第四步，每个将军检查所有新接收向量的每一个元素，若某个值占多数则把该值放入结果向量中，

将军 1(G1) 1K
将军 2(G2) 2K
将军 4(G4) 4K
将军 3(G3) X,Y,Z

a

G1 = (1K, 2K, X, 4K)
G2 = (1K, 2K, Y, 4K)
G3 = (1K, 2K, 3K, 4K)
G4 = (1K, 2K, Z, 4K)

b

G1 = [(1K,2K,X,4K) (1K,2K,Y,4K) (A,B,C,D) (1K,2K,Z,4K)]
G2 = [(1K,2K,X,4K) (1K,2K,Y,4K) (E,F,G,H) (1K,2K,Z,4K)]
G3 = [(1K,2K,X,4K) (1K,2K,Y,4K) (1K,2K,3K,4K) (1K,2K,Z,4K)]
G1 = [(1K,2K,X,4K) (1K,2K,Y,4K) (I,J,K,L) (1K,2K,Z,4K)]

c

G1 = (1K, 2K, UNKNOW, 4K)
G2 = (1K, 2K, UNKNOW, 4K)
G4 = (1K, 2K, UNKNOW, 4K)
G2 = (1K, 2K, 3K, 4K)

d

77. 云计算技术体系结构可以分为哪几层？

答：分为 4 层，物理资源层；资源池层；管理中间件层；SOA 构建层

物理资源层，包括计算机、存储器、网络设施、数据库和软件等。资源池层，将大量相同类型的资源构成同构或接近同构的资源池，构建资源池更多是物理资源的集成和管理工作。

管理中间件层，负责对云计算的资源进行管理，并对众多应用任务进行调度，使资源能够有效、安全地为应用提供服务。

SOA 构建层，将云计算能力封装成标准的 Web Services 服务，并纳入到 SOA 体系进行管理和使用，包括服务注册、查找、访问和构建服务工作流等。

（管理中间件和资源池层是云计算技术的最关键部分，SOA 构建层的功能更多依靠外部设施提供）

78. 简述云计算服务的三个层次

答：云计算层次与云服务层次如图1所示。

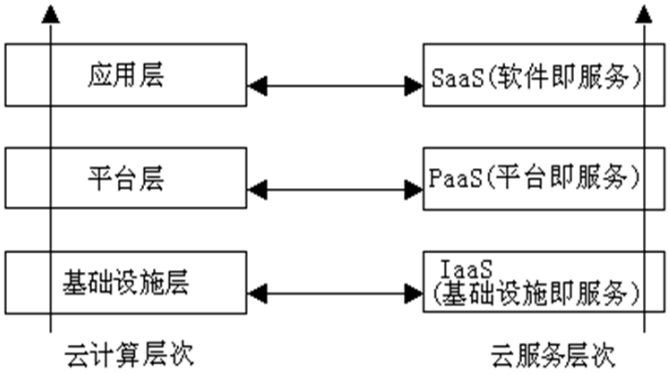


图 1　云计算层次与云服务层次

云计算可以认为包括以下几个层次的服务：基础设施级服务（IaaS）、平台级服务（PaaS）和软件级服务（SaaS）。这里所谓的层次，是分层体系架构意义上的“层次”。IaaS，PaaS，SaaS 分别在基础设施层、软件开放运行平台层、应用软件层实现。

IaaS(Infrastructure-as-a- Service)：基础设施级服务，消费者通过 Internet 可以从完善的计算机基础设施获得服务。IaaS 是把数据中心、基础设施等硬件资源通过 Web 分配给用户的商业模式。基础设施栈包括操作系统访问、防火墙、路由和负载平衡。示例产品：Flexiscale 和 Amazon EC2。

PaaS(Platform-as-a- Service)：平台级服务。PaaS 实际上是指将软件研发的平台作为一种服务，以 SaaS 的模式提交给用户。因此，PaaS 也是 SaaS 模式的一种应用。但是，PaaS 的出现可以加快 SaaS 的发展，尤其是加快 SaaS 应用的开发速度。PaaS 服务使得软件开发人员可以不购买服务器等设备环境的情况下开发新的应用程序。提供平台给系统管理员和开发人员，令其可以基于平台构建、测试及部署定制应用程序。也降低了管理系统的成本。典型服务包括 Storage、Database、Scalability。示例产品：Google App Engine、AWS:S3、Microsoft Azure。

SaaS(Software-as-a- Service)：软件级服务。它是一种通过 Internet 提供软件的模式，用户无需购买软件，而是向提供商租用基于 Web 的软件，来管理企业经营活动。SaaS 模式大大降低了软件尤其是大型软件的使用成本，并且由于软件是托管在服务商的服务器上，减少了客户的管理维护成本，可靠性也更高。示例产品：Google Docs、CRM、Financial Planning、Human Resources、Word Processing 等。

79. 简述云计算与分布式计算的关系

答：分布式计算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的结果。

分布式计算依赖于分布式系统，分布式系统由通过网络连接的多台计算机组成。网络把大量分布在不同地理位置的计算机连接在一起，每台计算机都拥有独立的处理器及内存。这些计

算机互相协作，共同完成一个目标或者计算任务。

分布式计算是一个很大的范畴，在当今的网络时代，不是分布式计算的应用已经很少了。因此，云计算也是分布式计算的一种。

80. 简述私有云、公用云和混合云的基本概念

答：私有云也叫做专用云，是由单个客户所拥有、按需提供的基础设施，该客户控制哪些应用程序在哪里运行，拥有自己的服务器、网络和磁盘，并且可以决定允许哪些用户使用基础设施。

公用云（公有云）是由第三方运行的云，第三方可以把来自许多不同客户的作业混合一起部署在云内的服务器、存储系统和其它基础设施上。因此，用户不知道运行其作业的同一台服务器、网络或磁盘上还有哪些用户。

混合云把公用云模式与私有云模式结合在一起，客户可通过一种可控的方式对云资源实现部分拥有、部分与他人共享。

81. 举例描述*aaS 的概念。

云计算按照其提供的“产品”或者是用户获得资源的类型，大致可以分为一些几种类别：

1）IaaS，全称 Infrastructure as a Service，基础设施即服务。将多台服务器组成的“云端”计算资源和存储，作为计量服务提供给用户。它将内存、I/O、存储和计算能力整合成一个虚拟的资源池向业界用户提供存储资源和虚拟化服务器等服务。如 Amazon EC2/S3。

2）PaaS，全称 Platform as a Service，平台即服务，把服务器平台或者开发环境作为一种服务提供的商业模式，以 SaaS 的模式提交给用户。用户在服务提供商的基础架构上开发程序并通过网络传送给其他用户（最终用户）。如 Force.com， Google App Engine，Microsoft Windows Azure。

3）SaaS，全称 Software as a Service，软件即服务，是基于互联网提供软件服务的软件应用模式。将应用软件统一部署于服务器（集群），通过网络向用户提供软件。用户根据实际需求定制或者租用应用软件。消除了企业或者机构购买、构建和维护基础设施和应用程序的投入。如 Saleforce online CRM。

4）DaaS，全称 Data as a Service，数据即服务，是继 SaaS，PaaS 之后又一个新的服务概念。

5）MaaS，全称 M2M as a Service，M2M 即服务，M2M 是将数据从一台终端传送到另一台终端，也就是就是机器与机器（Machine to Machine）的对话，是物联网四大支撑技术之一。

6）TaaS，全称 everyTHING As A Service，虚拟化云计算技术,SOA 等技术的结合实现物联网的泛在即服务。

82. 服务器虚拟化的特性？

①多实例

- 一个物理服务器上可以运行多个虚拟服务器
- 支持多个客户操作系统
- 物理系统资源以可控的方式分配给虚拟机

②隔离性

- 虚拟机之间完全隔离
- 一个虚拟机的崩溃不会对其他虚拟机造成影响
- 虚拟机之间的数据相对独立，不会泄露
- 虚拟机之间如果需要互相访问，方式等同于独立物理服务器之间的互相访问

③封装性

- 硬件无关
- 对外表现为单一的逻辑实体
- 一个虚拟机可以方便的在不同硬件之间复制、移动

- 将不同访问方式的硬件封装成统一标准化的虚拟硬件设备，保证了虚拟机的兼容性

④高性能

- 可通过扩展获得“无限”的性能
- 虚拟化抽象层需要一定管理开销

83. 服务器虚拟化的关键技术？

①计算虚拟化

- CPU 虚拟化
- 计算负载的动态分配
- 能耗管理

②存储虚拟化

- 内存虚拟化
- 磁盘存储动态分配

③设备与 I/O 虚拟化

- 软件方式实现
- 统一、标准化的接口
- 操作指令转译

④实时迁移技术

- 将整个虚拟机的运行状态完整、快速地从原宿主机的硬件平台转移到新的宿主机硬件平台。

84. Multitenant technique

Multitenant technique refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple tenants. With a multitenant architecture, a software application is designed to virtually partition its data and configuration, and each client organization works with a customized virtual application instance. Therefore, it is one of the essential attributes of Cloud Computing.

According to this understanding, basically Amazon Web Services (AWS), Force.com, Google App Engine, and Microsoft Azure are all examples of multitenant technique, as they all have multi-tenancy at some level in the system architecture stack.

多租户（就是能让一个单独的应用实例可以为多个组织服务，而且保持良好的隔离性和安全性，并且通过这种技术，能有效地降低应用的购置和维护成本）:i.是一种软件架构技术，它是在探讨与实现如何于多用户的环境下共用相同的系统或程序组件，并且仍可确保各用户间数据的隔离性。ii.实现重点，在于不同租户间应用程序竟的隔离以及数据的隔离,以维持不同租户间应用程序不会相互干扰，同时数据的保密性也够强。1)程序部分：通过进程或是支持多应用程序同时运行的装载环境来做进程间的隔离，或是在同一个服务程序进程内以运行就绪的方式隔离。2)数据部分：通过不同的机制将不同租户的数据隔离，Force 是采用中介数据的技术来切割，微软 MSDN 的技术文件则是展示了使用结构描述的方式隔离。

85. 简述什么是 SOAP，以及 SOAP 与 XML 之间的关系。

SOAP（Simple Object Access Protocol）即简单对象访问协议，是一个简单的用在 Web 上交换结构信息的 XML 协议，没有定义应用语义和传输语义，它以一种基于 XML 且与平台无关的 Web 编程方式改进了 Internet 的互操作性。SOAP 消息传递协议使用 HTTP 承载消息，而使用 XML 格式化消息。

SOAP 包括四个部分：

1，信封；2，数据的编码规则；3，RPC 调用规范；4，SOAP 绑定。 SOAP 与 XML 之间的关系是：

1，SOAP 是在 XML 基础上定义的，所有的 SOAP 消息都使用 XML 消息格式来编码，XML 是 SOAP 的底层技术规范；

2，对于 SOAP 中的简单类型，SOAP 采用了在 XML Schema 规范的数据类型部分定义的内嵌数据类型中所有类型，包括这些类型的值和词汇空间的定义；；

3，SOAP 完全继承了 XML 的开放性和描述可扩展性。

4，SOAP 还具有 XML 的其他一些优点，如可广泛应用于 web 的任何地方，使得编程更加简单，便于阅读等等。

86. 在面向消息的通信中，什么是持久通信和暂时通信？试举例说明。

答：持久通信：发送者发送消息后不需要再保持运行状态，接收者在发送者发送消息时也不需要处于运行状态。典型例子：电子邮件系统。传输的消息在提交之后由通信系统存储，直到将其交付给接收者。工作方式类似于驿马快递制度。

暂时通信：通信系统只在发送和接收消息的应用程序运行期间存储消息。典型例子：所有传输层通信服务，存储转发式路由器。

87. 讨论下列操作是否是幂等的:

• 按电梯的请求按钮。

.写数据到文件。

.将数据追加到文件。

操作不应该与任何状态相关是不是幂等的必要条件?

The operation to write data to a file can be defined (i) as in Unix where each write is applied at the read-write pointer, in which case the operation is not idempotent; or (ii) as in several file servers where the write operation is applied to a specified sequence of locations, in which case, the operation is idempotent because it can be repeated any number of times with the same effect. The operation to append data to a file is not idempotent, because the file is extended each time this operation is performed.

The question of the relationship between idempotence and server state requires some careful clarification.

It is a necessary condition of idempotence that the effect of an operation is independent of previous operations. Effects can be conveyed from one operation to the next by means of a server state such as a read-write pointer or a bank balance. Therefore it is a necessary condition of idempotence that the effects of an operation should not depend on server state. Note however, that the idempotent file write operation does change the state of a file.

88. 如何解决幂等性问题

- 全局唯一 ID

根据业务生成一个全局唯一 ID,在调用接口时会传入该 ID,接口提供方会从相应的存储系统比如 Redis 中去检索这个全局 ID 是否存在,如果存在则说明该操作已经执行过了,将拒绝本次服务请求;否则将相应该服务请求并将全局 ID 存入存储系统中,之后包含相同业务 ID 参数的请求将被拒绝。

- 去重表

这种方法适用于在业务中有唯一标识的插入场景。比如在支付场景中,一个订单只会支付一次,可以建立一张去重表,将订单 ID 作为唯一索引。把支付并且写入支付单据到去重表放入一个事务中,这样当出现重复支付时,数据库就会抛出唯一约束异常,操作就会回滚。这样保证了订单只会被支付一次。

- 多版本并发控制

适合对更新请求作幂等性控制,比如要更新商品的名字,这是就可以在更新的接口中增加一个版本号来做

幂等性控制

boolean updateGoodsName(int id,String newName,int version);

数据库更新的 SQL 语句如下

update goods set name=#{newName},version=#{version} where id=#{id} and version<${version}

- 状态机控制

适合在有状态机流转的情况下,比如订单的创建和付款,订单的创建肯定是在付款之前。这是可以添加一个 int 类型的字段来表示订单状态,创建为 0,付款成功为 100,付款失败为 99,则对订单状态的更新就可以这样表示

update order set status=#{status} where id=#{id} and status<#{status}

- 插入或更新

在 MySQL 数据库中，如果在 insert 语句后面带上 ON DUPLICATE KEY UPDATE 子句，而要插入的行与表中现有记录的惟一索引或主键中产生重复值,则对旧行进行更新;否则执行新纪录的插入。

我们可以利用该特性防止记录的重复插入,比如 good_id 和 category_id 构成唯一索引,则重复执行多次该 SQL,数据库中也只会有一条记录。

```
insert into goods_category (goods_id,category_id,create_time,update_time)
    values(#{goodsId},#{categoryId},now(),now())
    on DUPLICATE KEY UPDATE
    update_time=now()
```

89. 举例说明时间触发和事件触发的区别。

答：事件触发是指，当一个重要的外部事件触发时，它被传感器察觉到，并导致与传感器相连的 cpu 得到一个中断请求。

时间触发是指，在每隔固定的时间 t 后产生一次时钟中断，对选定的传感器进行采样，并且驱动（特定的）执行机构。

举例，考虑一个 100 层楼的电梯控制器设计。假定电梯正在 60 层安静的等待顾客，有人在一层按下按钮。就在 100 毫秒后，另一人在 100 层按下按钮。在事件触发系统中，第一次按钮产生一个中断，将使电梯启动下行，就在他做出下行决定后，第二个按下按钮的事件到来，因此第二个事件被记录下来以作将来的参考，但电梯还是继续下行。

若考虑时间触发系统，没 500 毫秒采样一次。若两次按下按钮都在一次采样周期中出现，控制器就不得不进行决定，例如按最近用户优先原则，此时电梯将上行。

由以上例子可以看出,事件触发的设计在低负载时会更快响应,但在高负载时可能崩溃。时间触发相反,仅适用于相对静态的环境。

90. 使用上载/下载模式的文件服务器系统与使用远程访问模式的文件系统之间有什么区别？

答：文件服务分为两种类型：上载/下载模式和远程访问模式。

上载/下载模式。只提供两种主要的操作（读文件和写文件），读操作将整个文件从文件服务器传输到请求客户端，写操作则刚好相反，文件系统运行在客户端。优点是系统概念简单，应用程序取得需要的文件，并在本地使用它。当程序结束时，将所有修改的文件或新创建的文件写回去，不需要管理复杂的文件服务接口，文件传输效率高。缺点是客户端需要足够的存储空间，当需要部分文件是需要传输整个文件。

远程访问模式。提供了大量的操作，如打开、关闭文件，读写部分文件等等。文件系统在服务器端运行。优点是客户不需要大量存储空间，当需要部分文件是不需要传输整个文件。

91. 说明保持客户高速缓存一致性的四种算法。

答：1）直接写，当缓存中的文件被更新后，新的值在缓存在保存，而且同时发送到服务器，而当另外的进程访问文件时，读到的是最新值，但是存在一个问题，其他进程在更新之前读到的文件内容可能是

过期的，那么在每次用到文件时需要从服务器中读取文件版本进行比较，查看是否过期，但是每次都要在服务器和客户端之间通信，这样就体现不出缓存的作用了。

2）延迟写，操作不立即发送给服务器，而是延迟一段时间，也就减少了网络消息，当进程读取文件时，依赖于时间。具体读到的是那次操作的结果不确定，语义模糊。

3）关闭写，操作只有在文件关闭后才写回服务器，配合 session 语义。

4）集中控制，就是在文件服务器上保存了进程对文件的操作方式等信息，类似于锁机制的管理，避免写操作的文件被其他进程操作，但是当修改的操作结束时，会将操作结束消息通知服务器，操作的结果也就立即会送到服务器

92. 试分别解释严格一致性、顺序一致性、因果一致性、PRAM 一致性等几种以数据为中心的一致性模型的含义。下图中的事件序列对上述哪几种一致性模型是有效的？

解答：

P1	W（X）1	W（X）3
P2	R（X）1 W（X）2	
P3	R（X）1	R（X）3 R（X）2
P4	R（X）1	R（X）2 R（X）3

严格一致性模型：所有共享访问事件都有绝对时间顺序；

顺序一致性模型：所有进程都以相同的顺序检测到所有的共享访问事件；

因果一致性模型：所有进程都以相同的顺序检测到所有因果联系的事件；

PRAM 一致性模型：所有的进程按照预定的顺序检测到来自一个处理器的写操作，来自其他处理器的写操作不必以相同的顺序出现；

图中的事件序列对因果一致性、PRAM 一致性是有效的。

93. 列出 URL 的 3 个主要成分，叙述如何表示它们的边界，并用例子说明每个成分。在什么程度上 URL 是位置透明的?

http://www.dcs.qmw.ac.uk/research/distrib/index.html

- The protocol to use. the part before the colon, in the example the protocol to use is http ("HyperText Transport Protocol").
- The part between// and/is the Domain name of the Web server host www.dcs.qmw.ac.uk.
- The remainder refers to information on that host named within the top level directory used by that Web server research/distrib/book.html.

The hostname www is location independent so we have location transparency in that the address of a particular computer is not included. Therefore the organisation may move the Web service to another computer.

But if the responsibility for providing a WWW-based information service moves to another organisation, the URL would need to be changed.

94. 简述基于中间件技术的分布式计算系统的组成结构。

参考答案：

对于基于中间件的分布计算系统，它的组成结构可以用图 1 表示。

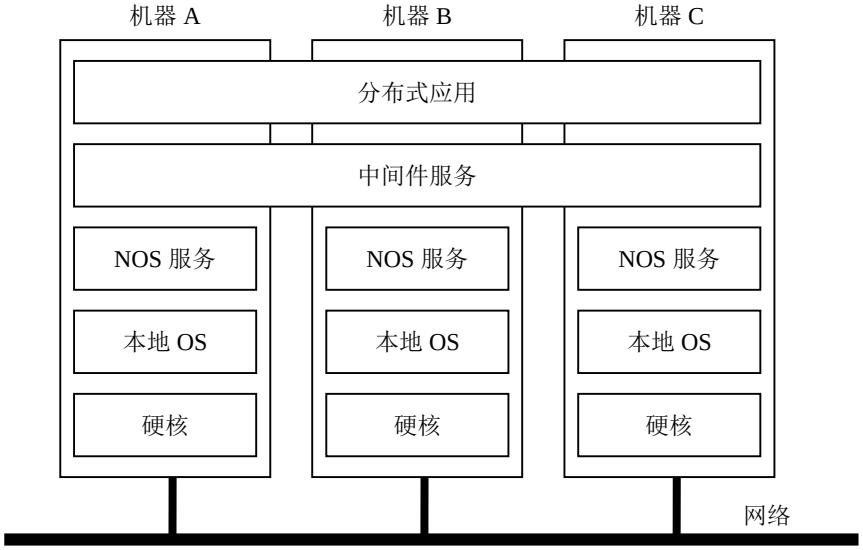
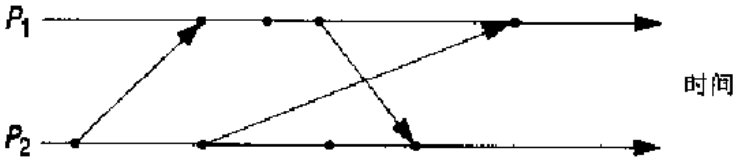


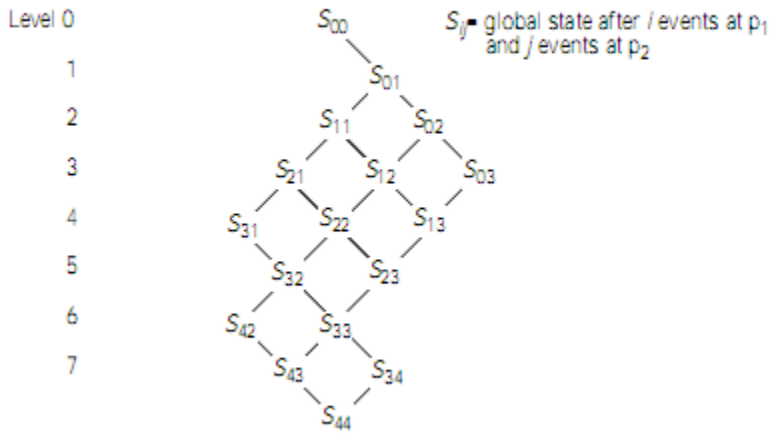
图 1 基于中间件的分布计算系统的组成结构

许多分布式应用程序直接使用网络操作系统提供的编程接口。例如，应用程序的通信功能通过网络操作系统提供的 socket 机制来实现，使得不同机器上的进程之间能够相互传送报文。另外，应用程序通过本地文件系统提供的接口使用文件。这种直接利用网络操作系统提供的服务编制的应用程序很难具有很高的透明度。为了解决这个问题，可以在应用程序和网络操作系统之间增加一个软件层次，提供一个更高的抽象层次。这样的一个软件层次称为中间件，它位于应用程序和网络操作系统之间，如图 1 所示。

95. 下图给出在两个进程 P1 和 P2 中发生的事件。进程之间的箭头表示消息传递。从初始状态〔0,0)开始，画出并标注一致状态(p1 的状态，p2 的状态)的网格。



The lattice of global states for the execution of Figure of exercise 10.15



96. 在多计算机系统中的 256 个 CPU 组成了一个 16 X 16 的网格方阵。在最坏的情况下，消息的延迟时间有多长（以跳(hop)的形式给出，跳是结点之间的逻辑距离）？

答：假设路由是最优的，最长的路由是从网格方阵的一个角落到对角的角落。那么这个路由的长度是 30 跳。如果一行或一列中的处理器彼此相连，则路由长度为 15 跳。

97. 虚拟机的两种架构的本质差别表现在那些方面？

虚拟机有两种架构：寄居架构（Hosted Architecture）和裸金属架构（“Bare Metal” Architecture)。

98. SAN 和 NAS 在存储虚拟化方面有哪些不同？

99. GAE 平台的应用服务架构与传统的 Web 应用服务架构有什么异同？

100. 说明分布式系统相对于集中式系统的优点和缺点。 从长远的角度看，推动分布式系统发展的主要动力是什么？

答：相对于集中式系统，分布式系统的优点：1）从经济上，微处理机提供了比大型主机更好的性能价格比；2）从速度上，分布式系统总的计算能力比单个大型主机更强；3）从分布上，具有固定的分布性，一些应用涉及到空间上分散的机器；4）从可靠性上，具有极强的可靠性，如果一个极强崩溃，整个系统还可以继续运行；5）从前景上，分布式操作系统的计算能力可以逐渐有所增加。

分布式系统的缺点：1）软件问题，目前分布式操作系统开发的软件太少；2）通信网络问题，一旦一个系统依赖网络，那么网络的信息丢失或饱和将会抵消我们通过建立分布式系统所获得的大部分优势；3）安全问题，数据的易于共享也容易造成对保密数据的访问。

推动分布式系统发展的主要动力：尽管分布式系统存在一些潜在的不足，但是从长远的角度看，推动分布式系统发展的主要动力是大量个人计算机的存在和人们共同工作于信息共享的需要，这种信息共享必须是以一种方便的形式进行。而不受地理或人员，数据以及机器的物理分布的影响

101. Browser/Server 体系结构与 Client/Server 体系结构相比不仅具有 Client/Server 体系结构的优点，而且又有 Client/Server 体系结构所不具备的独特优势。

答： 简单地说，GAE 平台的架构可以分为三部分：前端、Datastore 和服务群。前端是包含了 Front End、Static Files、App Server 和 App Master 的应用平台。Datastore 是基于 Bigtable 技术的分布式数据库，是 GAE 中唯一存储持久化数据的地方。服务群是供 App Server 调用的 API 服务。

传统的 web 架构自上而下可分为表现层、应用层、领域层、持久层。

开放的标准：Client/Server 所采用的通信协议往往是专用的。Browser/Server 所采用的标准，如 HTTP、HTML 等，都是开放的、非专用的，是经过标准化组织所确定的，保证了其应用的通用性和跨平台性。

较低的开发和维护成本：Client/Server 的应用必须开发出专用的客户端软件，无论是安装、配置还是升级都需要在所有的客户机上实施，极大地浪费了人力和物力。Browser/Server 的应用只需在客户端装有通用的浏览器即可，维护和升级工作都在服务器端进行，不需对客户端进行任何改变，故而大大降低了开发和维护的成本。

使用简单，界面友好：Client/Server 用户的界面是由客户端软件所决定的，其使用的方法和界面各不相同，每推广一个 Client/Server 系统都要求用户从头学起，难以使用。Browser/Server 用户的界面都统一在浏览器上，浏览器易于使用、界面友好，不须再学习使用其他的软件，一劳永逸地解决了用户的使用问题。

客户端简单：Client/Server 的客户端具有显示与处理数据的功能，对客户端的要求很高，是一个“胖”客户机。Browser/Server 的客户端不再负责数据库的存取和复杂数据计算等任务，只需要根据结果数据中指明的格式对其进行显示，充分发挥了服务器的强大作用，这样就大大地降低了对客户端的要求，客户端变得非常“瘦”。

102. 应用哪些技术可以使得一个分布式系统具有可伸缩性？

答：实现分布式可伸缩性，基本的三种技术为：

1、减少通信延迟，即使用异步通信方式，使得发送方发送请求后不必阻塞以等待答复，而是处理其他本地任务。

2、分层，即将一个组件分解为几个小层。一个好的例子是 DNS 域名系统，它将域名分为三层，均衡了系统负载。

3、复制冗余，它能使得资源更容易就近获取，并且它能使资源分布于整个系统，均衡了负载。

103. 给出一个多线程客户端的例子，并给出一种构造多线程服务器的方法。

答：多线程客户端例子，以网页浏览器为例：

浏览器在从服务器获取 HTML 文件时，同时也在显示它。因为一个 HTML 文件可能包含文本，图像，音频，视频等文件，故当一个线程获得其中一个文件并显示它时，同时还有其它线程正从服务器读取其它文件。即一个浏览器拥有多个线程与服务器进行交互。

构建多线程服务器：

使用有限状态机模型，它使用非阻塞系统调用方法，可实现并行处理多个请求。对每一个接收或发送的消息都将其处理状态存储到一个表中，由多线程对其进行处理。

104. 在代码迁移时，需要迁移代码片断、资源片断和执行片断，说明在迁移资源片断时需要考虑的主要问题。

答：迁移资源片段时，有时需要考虑改变资源片段的相关引用，以适应迁移后的使用，但是又不能改变该资源与其他进程之间的绑定关系。此外，有时还需考虑该资源片段与机器之间的绑定关系。例如有些资源片段迁移后可用，但是要花很大代价，有些则迁移后在其他机器上不可执行。

105. 什么是有状态服务器和无状态服务器，给出相应的例子，并说明有状态服务器存在的问题。

答：无状态服务器，在请求之间，服务器不保存具体客户的信息，以及与客户端交互活动的有关信息。它要求每个请求必须是独立的，必须包含全文件名和文件中的偏移量，因此消息长度较长。有状态服务器，在请求之间，服务器保存客户信息以及与客户交互活动的有关信息

106. 客户为了发送消息给服务器，它必须知道服务器的地址。试给出服务器进程编址的几种方法，并说明如何定位进程。

答：方法一。机器号加进程号，内核使用机器号将消息正确地发送到适当的机器上，用进程号决定将消息发送给哪一个进程。

方法二。进程选择随机地址，通过广播方式定位进程，进程在大范围的地址空间中随机指定自己的标识号。在支持广播式的 LAN 中，发送者广播一个特殊的定位包，其中包含目的进程地址，所有的内核查看地址是不是他们的，如果是则返回消息给出网络地址，然后发送内核缓存地址。

方法三。客户机运行时，使用 ASCII 码访问服务。客户机运行时，向名字服务器发送请求信息，名字服务器将 ASCII 服务器名映射成服务器地址，客户机收到给地址后，可以访问服务器。

107. 分布式系统的类型。

(1)分布式计算系统(分为群集计算系统和网格计算系统) (2)分布式信息系统(分为事务处理系统和企业应用集成) (3)分布式普适系统(如家庭系统、电子健保系统、传感器网络)

108. 客户端-服务器模型。

服务器（server）：实现某个特定服务的进程 客户（client）：向服务器请求服务的进程

客户端-服务器之间的一般交互：请求/回复（如下左图）

基于无连接协议的客户和服务器通信：高效，但是易受传输故障的影响（无法检测消息是否丢失也无法解释是否发生传输故障）。适合局域网。

基于连接的协议：性能相对较低，不适合局域网，适合广域网（基于可靠的 TCP/IP）。

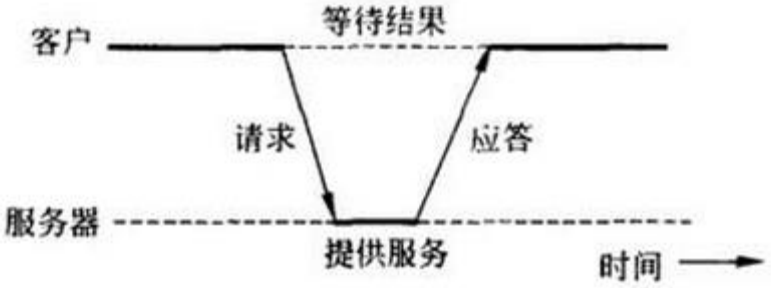


图 1.25　客户与服务器之间的一般交互

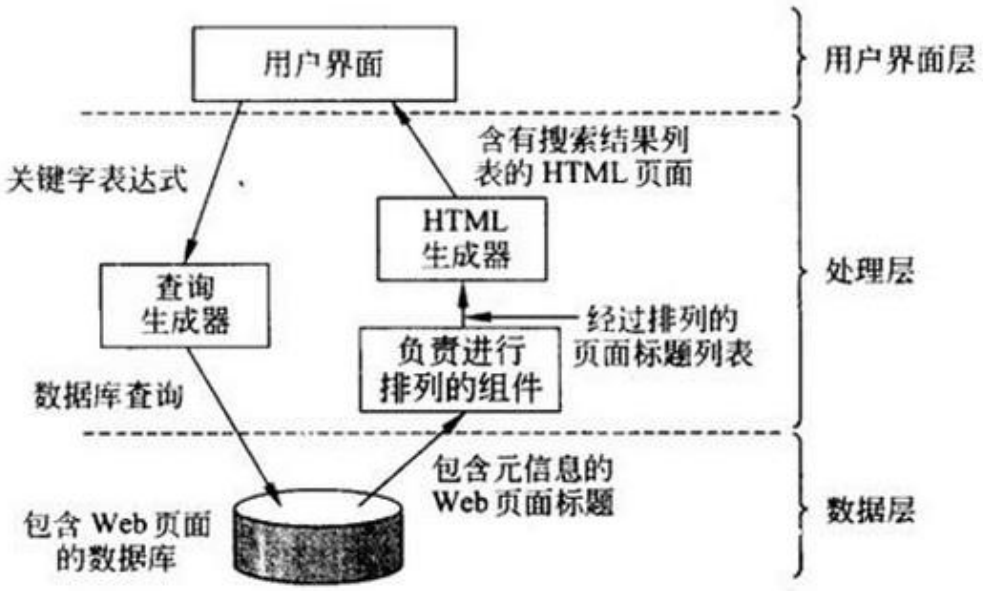


图 1.28　Internet 搜索引擎通常组织为三个不同层次

客户服务器应用程序通常组织为三个层次(如上右图)：(1)用户界面层：含有直接与用户交互所需的一切；(2)处理层：含有应用程序核心功能；(3)数据层：操作数据或文件系统，保持不同应用程序之间的数据一致性。

客户端-服务器模型可能的组织结构如下图：

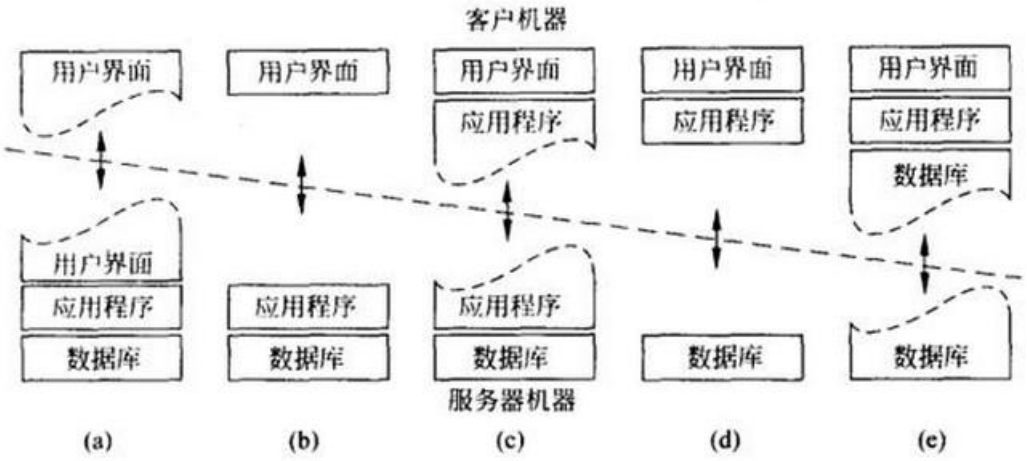


图 1.29 客户-服务器模型的可能组织结构

(a)只有与终端有关的用户接口部分位于客户机器上；(b)把整个用户接口软件放在客户端 (c)部分应用程序移到前端；(d)大多数的应用程序基本是运行在客户机上，但所有对文件或数据库项目的操作都是在服务器上；(e)同(d)，本地硬盘含有部分数据。

109. 解释透明性的含义，并举例说明不同类型的透明性。

答：对于分布式系统而言，透明性是指它呈现给用户或应用程序时，就好像是一个单独是计算机系统。具体说来，就是隐藏了多个计算机的处理过程，资源的物理分布。透明性：如果一个分布式系统能够在用户和应用程序面前呈现为单个计算机系统，这样的分布式系统就称为是透明的。

分类:1、访问透明性:隐藏数据表示形式以及访问方式的不同 2、位置透明性:隐藏数据所在位置 3、迁移透明性:隐藏资源是否已移动到另一个位置 4、重定位透明性:隐藏资源是否在使用中已移动到另一个位置 5、复制透明性:隐藏资源是否已被复制 6、并发透明性:隐藏资源是否由若干相互竞争的用户共享 7、故障透明性:隐藏资源的故障和恢复 8、持久性透明性:隐藏资源（软件）位于内存里或在磁盘上。

110. 进程和线程的比较。

进程定义为执行中的程序。未引入线程前是资源分配单位（存储器、文件）和 CPU 调度（分配）单位。引入线程后，线程成为 CPU 调度单位，而进程只作为其他资源分配单位。

线程是 CPU 调度单位，拥有线程状态、寄存器上下文和栈这些资源，同线程一样也有就绪、阻塞和执行三种基本状态。

（1）对于地址空间和其他资源（如打开文件）来说，进程间是相互独立的，同一进程的各线程间共享该进程地址空间和其他资源(某进程内的线程在其他进程不可见)。

（2）在通信上，进程间通信通过 IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信－－需要进程同步和互斥手段的辅助，以保证数据的一致性。

（3）在调度上，线程上下文切换比进程上下文切换要快得多。线程是 CPU 调度单位，而进程只作为其他资源分配单位。线程的创建时间比进程短；线程的终止时间比进程短；同进程内的线程切换时间比进程短。 因此，多线程能提高性能；线程不像进程那样彼此隔离，并受到系统自动提供的保护，因此多线程应用程序开发需要付出更多努力。

111. 多线程服务器的优点？

多线程技术不仅能够显著简化服务器代码，还能够使得应用并行技术来开发高性能的服务器变得更加容易，即使在单处理器系统上也是如此。多线程能够保留顺序处理的思路，使用阻塞性系统的系统调用，仍然能到达并行处理的目的。使用阻塞系统调用使编程更容易，并行处理能提高系统的性能。

112. 什么软件代理？举例说明其作用。

软件代理是一些独立的单元，能与其他的代理进行协作，一同执行任务。定义为对环境的变

化做出反应，并且启动这种变化的自治进程，而且可以与用户代理或其他代理协同。与进程的区别在于能够对自己执行操作，在适当的时候采取主动。

代理分类：

(1)合作代理：通过协作达到某个共同的目标：会议安排

(2)移动代理：能够在不同机器间迁移

(3)接口代理：协助最终用户使用应用程序，拥有学习能力：促成买卖

(5)信息代理：管理来自多个信息源的信息：排序、过滤和比较

a.固定信息代理：电子邮件代理

b.移动信息代理：网络漫游，搜集所需信息

113. 代码迁移的动机有哪些？

代码迁移指的是将程序（或执行中的程序）传递到其它计算机。(基本思想：把进程由负载较重的机器上转移到负载较轻的机器上去，就可以提升系统的整体性能)

迁移动机：

（1）实现负载均衡：将进程从负载重的系统迁移到负载轻的系统，从而改善整体性能。（2）改善通信性能：交互密集的进程可迁移到同一个节点执行以减少通信开销，当进程要处理的数据量较大时，最好将进程迁移到数据所在的节点。

（3）可用性：需长期运行的进程可能因为当前运行机器要关闭而需要迁移。（4）使用特殊功能：可以充分利用特定节点上独有的硬件或软件功能。（5）灵活性：客户首先获取必需的软件，然后调用服务器。

114. 代码迁移时进程对资源的绑定类型有哪些？

进程对资源的绑定类型有三类：分别是按标志符（URL）、按值和按类型。

115. 代码迁移时资源对机器的绑定类型有哪些？

资源对机器绑定类型分成：未连接（数据文件）、附着连接（数据库）和紧固连接（本地设备）三类。

资源对机器绑定

		未连接	附着连接	紧固连接
进程对资源绑定	按标志符	MV (or GR)	GR (or MV)	GR
	按值	CP (or MV, GR)	GR (or CP)	GR
	按类型	RB (or MV, CP)	RB (or GR, CP)	RB (or GR)

- MV：移动资源
- GR：建立全局系统范围内引用
- CP：复制资源的值
- RB：将进程重新绑定到本地同类型资源

116. 什么是远程过程调用？远程过程调用的步骤。

远程过程调用(Remote Procedure Call)RPC 是指本地程序调用位于其他机器上的进程，调用方通过消息的形式把参数传递到被调用方的进程，然后等待被调用方执行完后用消息的方式把结果传回调用方。具体步骤是：

（1）客户过程以正常的方式调用客户存根

（2）客户存根生成一个消息，然后调用本地操作系统 （3）客户端操作系统将消息发送给远程操作系统 （4）远程操作系统将消息交给服务器存根

（5）服务器存根将参数提取出来，然后调用服务器

（6）服务器执行要求的操作，操作完成后将结果返回给服务器存根（7）服务器存根将结果打包成一个消息，然后调用本地操作系统

（8）服务器操作系统将含有结果的消息发送回客户端操作系统 （9）客户端操作系统将消息交给客户存根

（10）客户存根将结果从消息中提取出来，返回给调用它的客户过程

117. 什么是远程对象调用？

远程对象调用指的是在本地调用位于其他机器上的对象。和远程过程调用主要的区别在于方法被调用的方式。在远程对象调用中，远程接口使每个远程方法都具有方法签名。如果一个方法在服务器上执行，但是没有相匹配的签名被添加到这个远程接口上，那么这个新方法就不能被远程对象调用的客户方所调用。在远程过程调用中，当一个请求到达远程过程调用的服务器时，这个请求就包含了一个参数集和一个文本值。

118. 消息同步通信与异步通信的区别？

异步通信特征在于发送者要把传输的消息提交之后立即执行其他的程序，这意味着该消息存储在位于发送端主机的本地缓冲区里中，或者存储在送达的第一个通信服务器上的缓冲区上中。 而对于同步通信来说，发送者在提交信息之后会被阻塞直到消息已经到达并储存在接收主机的本地缓冲区中以后也就是消息确实已经传到接收者之后，才会继续执行其他程序。

119. DNS 名称解析的方法有哪两种？各自优缺点？

在分布式系统中，由于多个名称服务器名字空间的分散性，影响了名称解析的实现。通常有两种方法实现名称解析：

- 迭代名称解析：名称解析程序将完整的名称转发给根名称服务器，根服务器尽量解析路径名，然后把结果返回给客户，然后，客户再把剩下的路径名发送给下一台名称服务器，直到名称解析完成
- 递归名称解析：与迭代解析返回第个中间结果给客户不同，名称服务器会直接把结果传递给它找到的下一台服务器，继续解析，直到完成解析

递归名称解析的优点是，与迭代名称解析相比，缓存结果要更为有效，同时，通信开销要小。主要缺点是它要求每台名称服务器有较高的性能。

120. Lamport 时间戳算法的思想

Lamport时间戳算法基本思想是：

多个进程分别在多台机器上运行，每台机器都有自己的时钟，各时钟的工作速率不同。每个消息均携带有送者时钟的发送时间。如果消息到达接收者时，接收者时钟显示的时间大于发送时间时，认为时钟是正常的，符合发送者时间先于接收者时间这一规律（规则2）。如果消息到达接收者时，接收者显示的时钟小于发送时间，接收者就将它的时钟调到一个比发送时间大1的值。

为满足全局时间的需要，可以对这个算法稍作补充，规定在每两个事件之间，时钟必须至少滴答一次。如果一个进程以相当快的速度连续发送或接收两个消息，那么它的时钟必须在这之间至少滴答一次。

此外，可以将事件发生所在的进程号附加在时间的低位后面，并用小数点分开。这样，如果进程1和进程2中的事件都发生在时刻20，那么，它们的时间分别记为20.1和20.2。

Lamport算法为我们提供了一种对系统中所有的事件进行完全排序的方法。

121. 当发现一个时钟 4s 时，它的读数是 10:27:54.0(小时:分钟:秒)。解释为什么在这时不愿将时钟设成正确的时间，并(用数字表示)给出它应该如何调整以便在 8s 后变成正确的时间。

Some applications use the current clock value to stamp events, on the assumption that clocks always advance.

We use *E* to refer to the 'errant' clock that reads 10:27:54.0 when the real time is 10:27:50. We assume that *H* advances at a perfect rate, to a first approximation, over the next 8 seconds. We adjust our software clock *S* to tick at rate chosen so that it will be correct after 8 seconds, as follows:

$S = c(E - Tskew) + Tskew$, where $Tskew$ = 10:27:54 and c is to be found.

But $S = Tskew+4$ (the correct time) when $E = Tskew+8$, so:

$Tskew+ 4 = c(Tskew + 8 - Tskew) + Tskew$, and c is 0.5. Finally:

$S = 0.5(E - Tskew) + Tskew$ (when $Tskew \quad E \quad Tskew+8$).

122. 全局状态

全局状态定义了每个进程的本地状态和正在传输中的消息

一致的全局状态：如果已经记录了一个进程 P 收到了来自进程 Q 的一条消息，那么也应该记录 Q 确实已经发送了那条消息。

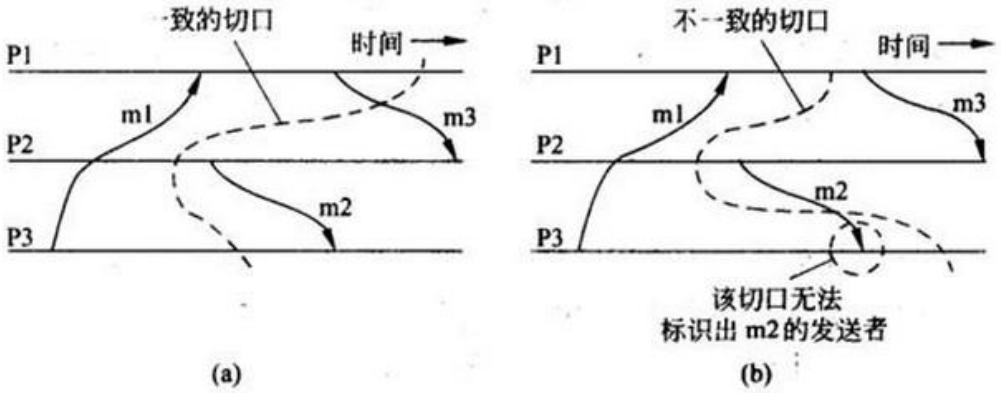
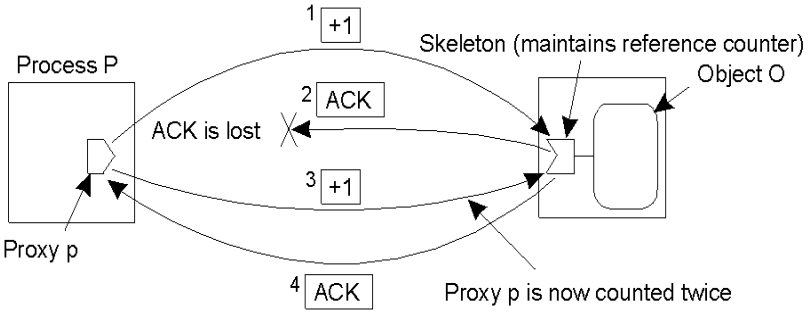


图 5.9 一致的切口和不一致的切口

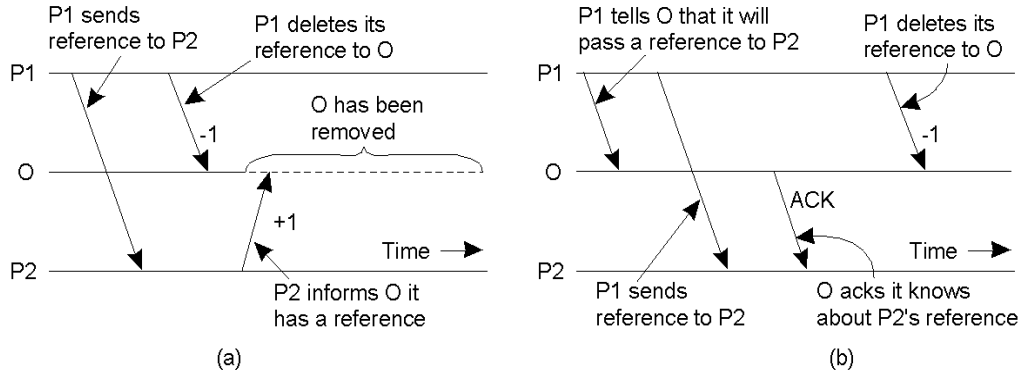
(a) 一致的切口；(b) 不一致的切口

123. 删除无引用实体的方法

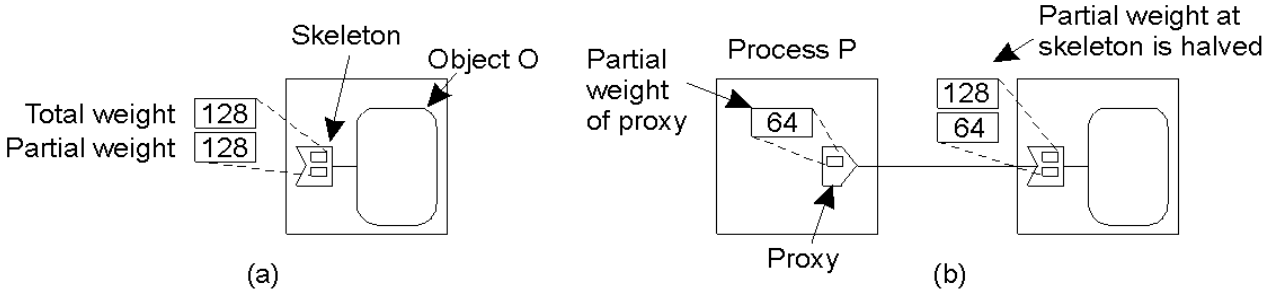
（1）**引用计数方法**：包括**简单引用计数**和**高级引用计数**方法



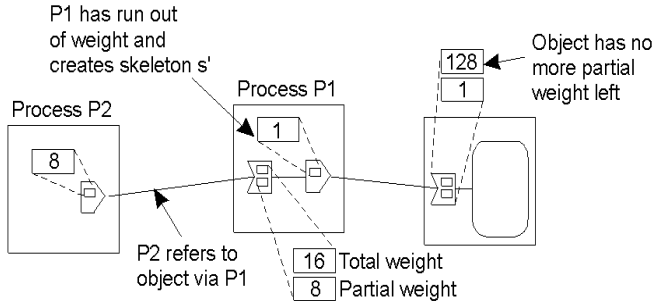
在通信不可靠的情况下维护正确的引用计数所存在的问题

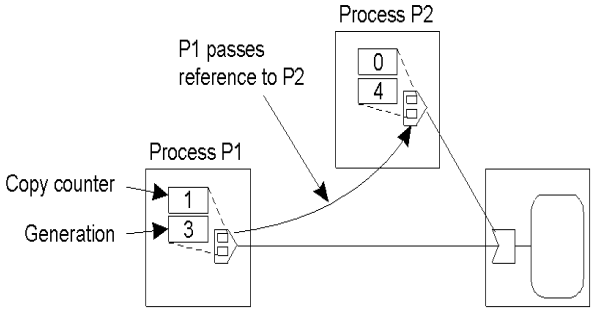


a)向其他进程复制引用计数之后再递增引用计数　b) 解决方法



a)加权引用计数中权数的初始值　　b)创建新引用时的权数值





在引用的部分权数达到 1 时创建一个间接权数　　在世代引用计数中创建和复制引用

（2）**引用列表方法**：在骨架端维护一张明确的列表，持续跟踪引用他的代理。

优点：不需要可靠通信；进程发生故障时，容易保持引用列表的一致性

缺点：引用列表的规模问题---注册的引用在有效时间内有效（分发租用）

124. 设计一个分布式垃圾回收算法，简述其原理

引用计数（Reference Counting）

分布式标记清除算法

标记-清除（Mark-Sweep）

125. 选举算法中 Bully 算法的思想

选举算法：选择一个进程作为协调者、发起者或其他特殊角色，一般选择进程号最大的进程（假设每个进程都知道其他进程的进程号，但不知道是否还在运行）它的目的是保证在选举之行后，所有进程都认可被选举的进程。 Bully 算法：

当进程 P 注意到需要选举一个进程作协调者时： (1)向所有进程号比它高的进程发 ELECTION 消息 (2)如果得不到任何进程的响应，进程 P 获胜，成为协调者

(3)如果有进程号比它高的进程响应，该进程接管选举过程，进程 P 任务完成 (4)当其他进程都放弃，只剩一个进程时，该进程成为协调者 (5)一个以前被中止的进程恢复后也有选举权

126. 选举算法中环算法的思想

不使用令牌，按进程号排序，每个进程都知道自己的后继者，当进程 P 注意到需要选举一个进程作协调者时：

(1)创建一条包含该进程号的 ELECTION 消息，发给后继进程 (2)后继进程再将自己的进程号加入 ELECTION 消息，依次类推

(3)最后回到进程 P，它再发送一条 COORDINATOR 消息到环上，包含新选出的协调者进程（进程号最大者）和所有在线进程，这消息在循环一周后被删除，随后每个进程都恢复原来的工作。

127. 分布式互斥算法

（1）当进程想进入临界区时，它向所有其他进程发一条打了时间戳的消息 Request （2）当收到所有其他进程的 Reply 消息时，就可以进入临界区了 （3）当一个进程收到一条 Request 消息时，必须返回一条 Reply 消息：

- 如该进程自己不想进入临界区，则立即发送 Reply 消息
- 如该进程想进入临界区，则把自己的 Request 消息时间戳与收到的 Request 消息时间戳相比较，
  - 如自己的晚，则立即发送 Reply 消息
  - 否则，就推迟发送 Reply 消息

128. 给出 Lamport 时间戳互斥算法的主要思想

参考答案：

Lamport 时间戳互斥算法由以下 5 条规则组成，为方便起见，认为每条规则定义的活动形成单个事件：

①一个进程 P_i 如果为了申请资源，它向其它各个进程发送具有时间戳 T_m:P_i 的申请资源的报文，并把此报文也放到自己的申请队列中；

②一个进程 P_j 如果收到具有时间戳 T_m:P_i 的申请资源的报文，它把此报文放到自己的申请队列中，并将向 P_i 发送一个带有时间戳的承认报文。如果 P_j 正在临界区或正在发送自己的申请报文，则此承认报文要等到 P_j 从临界区中退出之后或 P_j 发送完自己的申请报文之后再发送，否则立即发送；

③一个进程 P_i 如果想释放资源，它先从自己的申请队列中删除对应的 T_m:P_i 申请报文，并向所有其他进程发送具有时间戳的 P_i 释放资源的报文；

④一个进程 P_j 如果收到 P_i 释放资源的报文，它从自己的申请队列中删除 T_m:P_i 申请报文；

⑤当满足下述两个条件时，申请资源的进程 P_i 获得资源：

（a）P_i 的申请队列中有 T_m:P_i 申请报文，并且根据时间戳它排在所有其它进程发来的申请报文前面；

（b）P_i 收到所有其它进程的承认报文，其上面的时间戳值大于 T_m。

使用 Lamport 时间戳互斥算法，进入一次临界区共需要 3(n-1)个报文，n 为参加互斥的

进程数。

129. 什么叫容错性？

容错意味着系统即使发生故障也能提供服务，容错与可靠性相联系，包含以下需求：可用性（Availability）：任何给定的时刻都能及时工作 可靠性（Reliability）：系统可以无故障的持续运行

安全性（Safety）：系统偶然出现故障能正常操作而不会造成任何灾难 可维护性（Maintainability）：发生故障的系统被恢复的难易程度

130. 掩盖故障的冗余有哪几种？

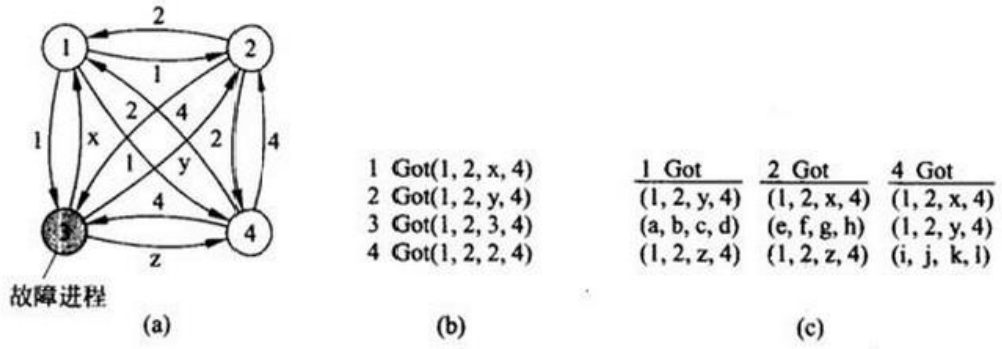
信息冗余：增加信息，如海明码校验。 时间冗余：多次执行一个动作，如事务。

物理冗余：增加额外的设备或进程。如三倍的模块冗余。

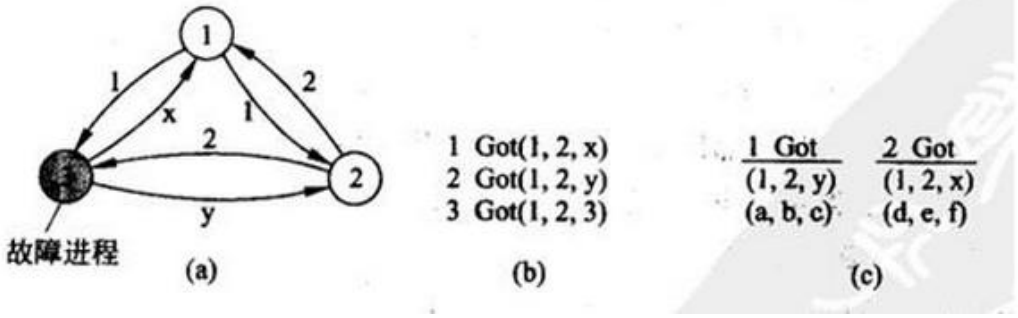
131. 拜占庭将军问题

Lamport 证明在具有 m 个故障进程的系统中，只有存在 2m+1 的正常工作的进程才能达成协议，即总进程为 3m+1。

三个忠诚将军和一个叛徒的问题，叛徒通过发送错误和矛盾来组织忠诚将军达成协议。如下图：假设现在将军们要相互告知自己的兵力，将军三是叛徒。每个人都将自己收到的向量发给其他所有的将军。



这个时候，将军们把自己收到的向量中，哪一个数出现得最多就把他纪录到自己的向量中去，因此将军 124 都可以看到相同的消息：（1,2，UNKNOWN，4）。但是如果有两个忠诚将军和一个叛徒将军，就不能判断了。如下图：



1、2 得到的信息都没有出现大多数情况，所以，该算法不能产生协定。

132. NFS 的共享预约

NFS(Network File System 网络文件系统)共享预约：一种锁定文件的隐含方法。客户打开文件时，指定它所需的访问类型（READ、WRITE 或 BOTH）,以及服务器应该拒绝的其他客户的访问类型（ NONE、READ、WRITE 或 BOTH ）。如果服务器不能满足客户的需求，那么该客户的 open 操作失败。

当前文件的拒绝状态

请求访问	NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
BOTH	Succeed	Fail	Fail	Fail

(a)

请求的文件拒绝状态

当前访问状态	NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
BOTH	Succeed	Fail	Fail	Fail

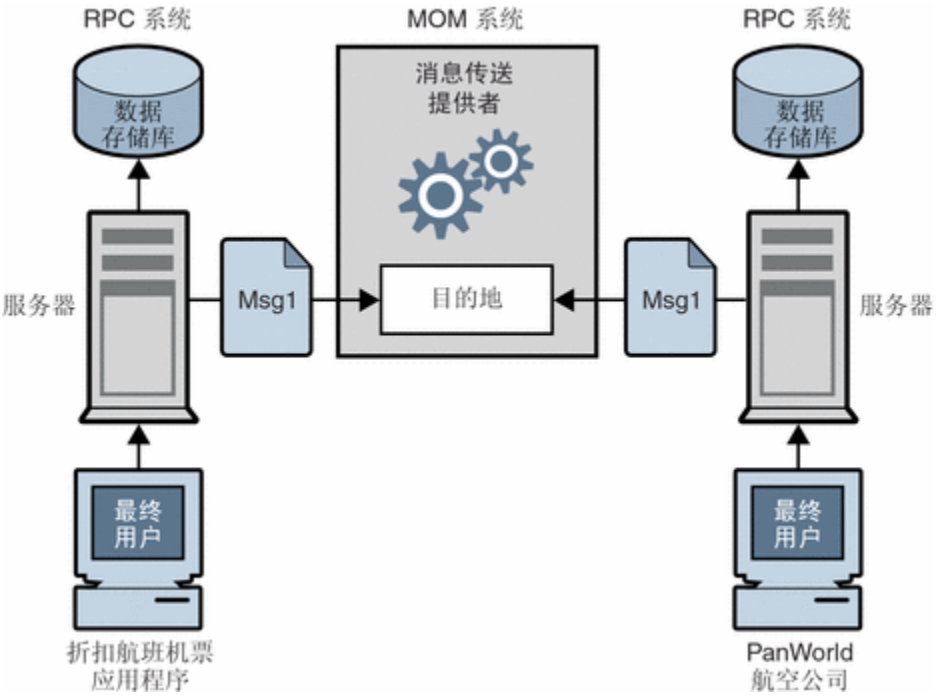
(b)

使用 NFS 共享预约实现操作的结果（新客户试图打开另一客户已成功打开的文件）。

a）当前文件的拒绝状态下客户请求共享访问的情况

b）在当前文件访问状态下，客户请求拒绝状态的情况

133. 假如开发一个高吞吐的航空机票订单服务，需要手机端、网页端、以及其他第三方接口调用，为了保证高吞吐，如何用 RPC+消息队列方式来实现，说说你的设计思想和实现思路



实现思路大致如上图，不同的客户端通过 rpc 提交订单请求，rpc 接到订单请求，封装消息发送给消息中间件。从而 rpc 订单系统可以快速响应客户端，同时解耦订单系统与后台系统。后台系统以及第三方接口系统可以作为消息队列的消费者。

134. 现要为某网上商城实现一个商品价格查询服务，该服务具有以下功能：

用户可以主动查询某个商品的价格。

用户可以订购某个商品的价格，当商品价格低于用户指定的阈值时，该服务通知订购用户当前的价格。

多个用户可同时使用该服务。

现要使用面向对象的技术，如 CORBA 技术实现该服务：请描述该服务对象和客户端程序分别需要实现的接口。接口可以采用任何一种程序设计语言描述（甚至夹杂自然语言），但要明确每个接口名、接口中的方法名、方法的返回值和参数名以及类型。

商品价格查询服务的接口：

方法一：价格查询。
Float getPrice（String goodID） throws someFailure；返回值为价格。
方法二：订购价格变化情况
Void subscribe(String goodID, float myInterestPrice, Ref myCallback) throws some someFilure
其中，myInterestPrice 为指定的价格阈值，myCallback 为实现 nicePrice（）方法的客户端回调接口对象引用。
客户端实现的接口 方法一：
Void nicePrice(String goodID, float nicePrice) throws some someFilure; 其中，nicePrice 是低于阈值的新价格。
参数类型和名字等，可在合理范围内变动。

另外，教材课后习题 1.2,1.4，2.9, 2.11, 2.18, 4.10, 4.15, 5.22, 5.23, 6.9, 12.7, 14.4, 14.7, 15.5, 15.23，14.14