重点内容整理

第一章

1、分布式系统的定义

1. **定义**：分布式计算系统是由多个相互连接的计算机组成的一个整体，这些计算机在一组系统软件（分布式操作系统或中间件）环境下，合作执行一个共同的或不同的任务，最少依赖于集中的控制过程、数据和硬件。
2. 硬件上：
    - 包含多个通用部件，可以动态基础上被指定给予各个特定的任务；
    - 这些资源部件在物理上是分布的，并经过一个通信网络相互作用
3. 控制和数据上
    - 有一个高级操作系统，对各个分布的资源进行统一和整体的控制
    - 系统对用户是透明的
    - 所有的资源都必须高度自治地工作而又相互配合，系统内不存在控制层次

2、分布式系统的要求，开放性、可扩展性、异构型、透明性，每种特性掌握含义，主要表现，能举例说明

1. **开放性**：
    - 含义：指能否用各种方法进行扩展和重新实现
    - 主要表现：通信标准与协议、服务通常通过接口、互操作性、可移植性、灵活组合的
2. **可扩展性**：
    - 含义：一个系统在资源数量和用户数量增加时仍能有效工作
    - 主要表现：规模、地域、管理上可拓展
3. **异构型**：
    - 含义：Internet允许用户访问在异构计算机和网络上的服务
    - 主要表现：网络、计算机硬件、操作系统、程序设计语言、不同开发商的实现
4. **透明性**：
    - 含义：向用户和应用程序隐藏了分布式计算系统部件的差异
    - 主要表现：访问、位置、并发、失效、复制、迁移、性能、规模透明性
5. 安全性：

3、中间件的概念，提供的服务等

1. **中间件概念**：中间件是介于应用系统和系统软件之间的一类软件，它使系统软件所提供的基础服务，衔接网络上应用系统的各个部分或不同应用，能够达到资源共享、功能共享的目的。
2. **提供的服务**：
    - 命名服务：允许实体被共享和查询
    - 作业调度：处理机时间分配、作业指派等
    - 高级通信服务
    - 资源管理
    - 数据持久化
    - 分布式事务
    - 分布式文档系统

- 安全服务

第二章

1、实体的概念及其名字、地址等的具体含义。

1. **实体的概念**：实体在一个计算机系统中指范围广泛的事务，包括计算机主机、外围设备，进程、数据、文件、数据库、服务、服务器和用户等。
2. 名字：实体的名字是一个用户可读的、便于记忆的字符串
3. 地址：访问点在分布式系统中是一个特殊实体，它的名字称为地址
4. 标识符：是用来表示某个实体的一个符号

2、名字服务器的组成，名字解析的几种方法。

1. **名字服务器的组成**：名字服务器主要是由数据库和名字解析软件组成的。数据库实现实体名和地址的绑定，及其他信息，称为名字解析上下文。

- 名字服务器操作：包括上下文管理、查询操作和行政管理。
- 名字解析：根据名字解析，利用数据库中的上下文信息对名字进行解析。
- 缓存：缓存名字查询和解析结果
- 多副本管理：副本修改和附本一致性维护
- 通信：包括与客户端的名字代理通信和名字服务器之间的通信
- 数据库：存放名字解析上下文或其子域

2. **名字解析的几种方法**
    - 迭代名字解析
    - 递归名字解析
    - 比较：递归名字解析的主要缺陷是要求每台名字服务器具有较高的性能，但是可以减少长距离通信

3、白页服务、黄页服务、绿页服务的区别

- 名字服务（**白页服务**）：名字数据库是命名实体与其地址绑定的集合。**名字服务是根据实体的名字查找它的地址。**
- 目录服务（**黄页服务**）：目录数据库命名实体与其一个或多个属性绑定的集合。**属性包括属性类型和一个或多个属性值。**可以根据实体的名字查找实体的属性，也可以根据实体的一个或多个属性及其值查找实体的名称。如X.500目录服务
- 合约服务（**绿页服务**）：一种增强的目录服务，它通过技术规范定位一个命名实体，如Web服务

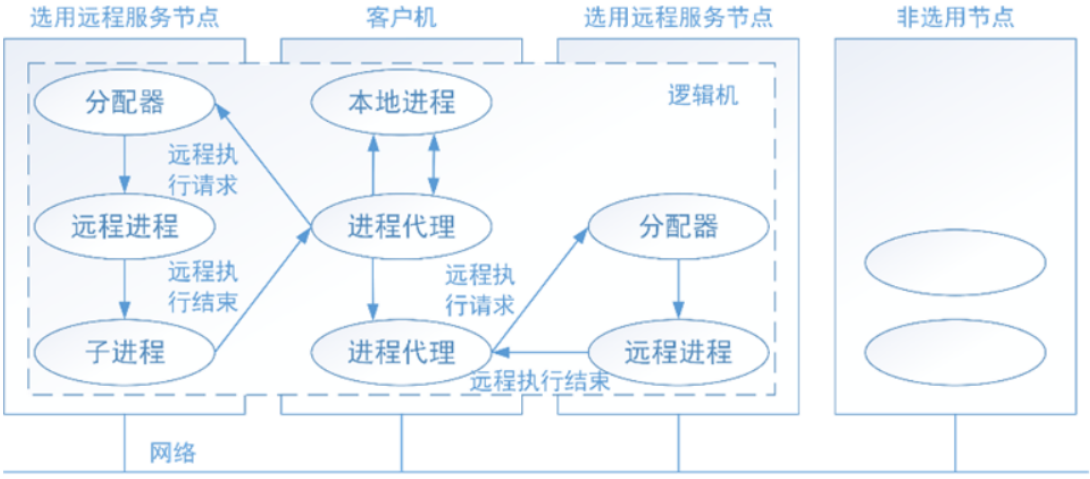
4、目录服务、LDAP

1. 目录服务：
    - 目录服务是一种特殊类型的名字服务，除了根据实体名查找它的属性，用户可以基于属性描述来查找实体，而不用完整的实体名。
    - X.500目录访问时在国际电信联盟主持下，目的是为目录系统提供一个国际标准。
2.
    - 目录服务组件/目录服务的组成
        - 目录信息库DIB(Directory Information Base)
        - 目录系统代理DSA(Directory System Agents)
        - 目录用户代理DUA(Directory User Agents)
        - 目录管理域DMD(Directory Management Domain)
    - 目录服务操作
        - 目录查询（响应）：为客户提供目录查询服务，向客户提供目录信息
        - 镜像操作绑定：设置多副本，提高目录服务的性能与可用性
        - 层次操作绑定：在DSA之间建立层次关系
    - 目录服务协议
        - X.500目录服务有4个协议：
        - 目录访问协议DAP，DUA用来与DSA通信。
        - 目录系统协议DSP，是两个DSA之间的操作协议，在DSA之间传递查询请求和响应。
        - 目录信息镜像协议DISP，是DSA用来将信息从镜像提供者传送给镜像使用者。
        - 目录操作绑定管理协议DOP，DSA用来层次操作绑定管理和镜像管理。
3. LDAP(LightWeight Directory Access Protocol)轻量级目录访问协议
    - DAP是DUA和DSA之间的请求/响应协议，包括目录查询和目录修改。
    - LDAP是用户用来访问目录服务的一个协议，目的是代替DAP访问X.500目录
    - 相比DAP，LDAP向用户提供目录服务时避免了大量的开销，操作集做了简化。

第三章

1、逻辑机模型

1. 逻辑机模型：模型中有两个部件：客户节点和远程服务节点。



    - 远程进程必须能访问驻留在源计算机上的文件系统
    - 远程进程能接受逻辑机内任何进程发来的信号，也能将信号提供给逻辑机内任何进程
    - 进程组保持在逻辑机内
    - 基于树型的进程父子关系在逻辑机内必须得以保持

2、远程执行的过程，能以REXEC为例说明具体过程

（1）用户本地运行环境的传播和重建。

（2）本地信号（signal）和stdin转发。

（3）远程stdout和stderr转发。

（4）由本地作业控制实现对远程作业进程控制。

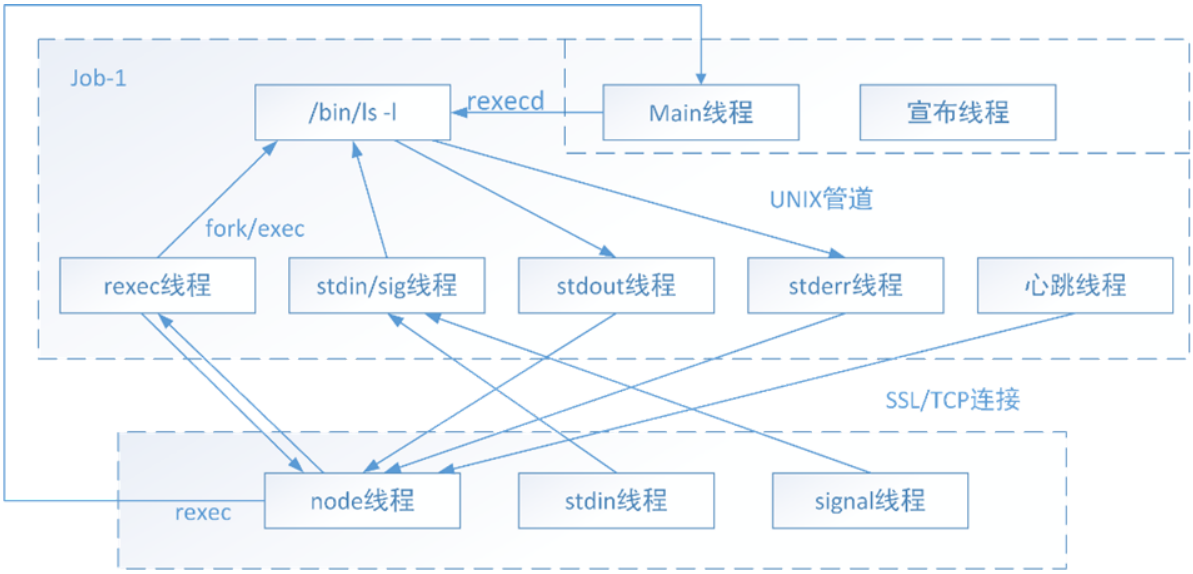


图 2 远程服务节点上建立环境和运行客户进程

1） 用户本地运行环境由rexec中的node线程打包发送，由rexecd中的rexec线程在用户作业派生（Forking）之后和执行之前接收，并在远程服务节点上重建。

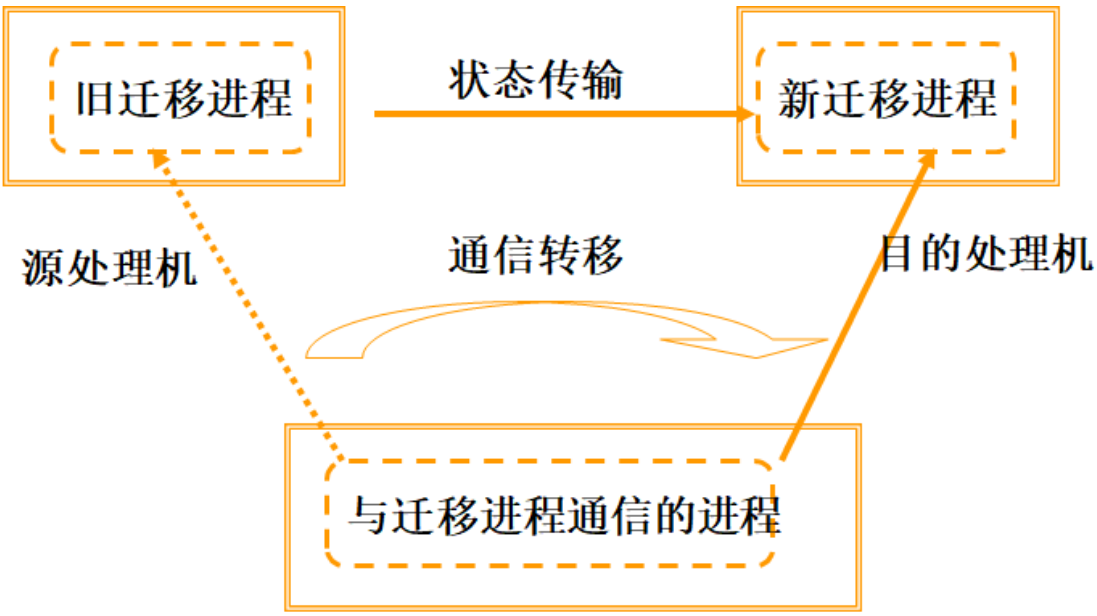
2） 本地信号（signal）和stdin是由rexec的signal线程和stdin线程转发到远程服务节点rexecd的stdin/sig线程，然后使用signal和UNIX管道将它们分发给该服务节点上的远程应用程序。

3） 远程stdout和stderr转发由rexecd的stdout线程和stderr线程通过UNIX管道将远程服务节点的stdout和stderr转发到rexec的node线程。

4） 由本地作业控制实现对远程作业进程的控制，远程执行透明性。

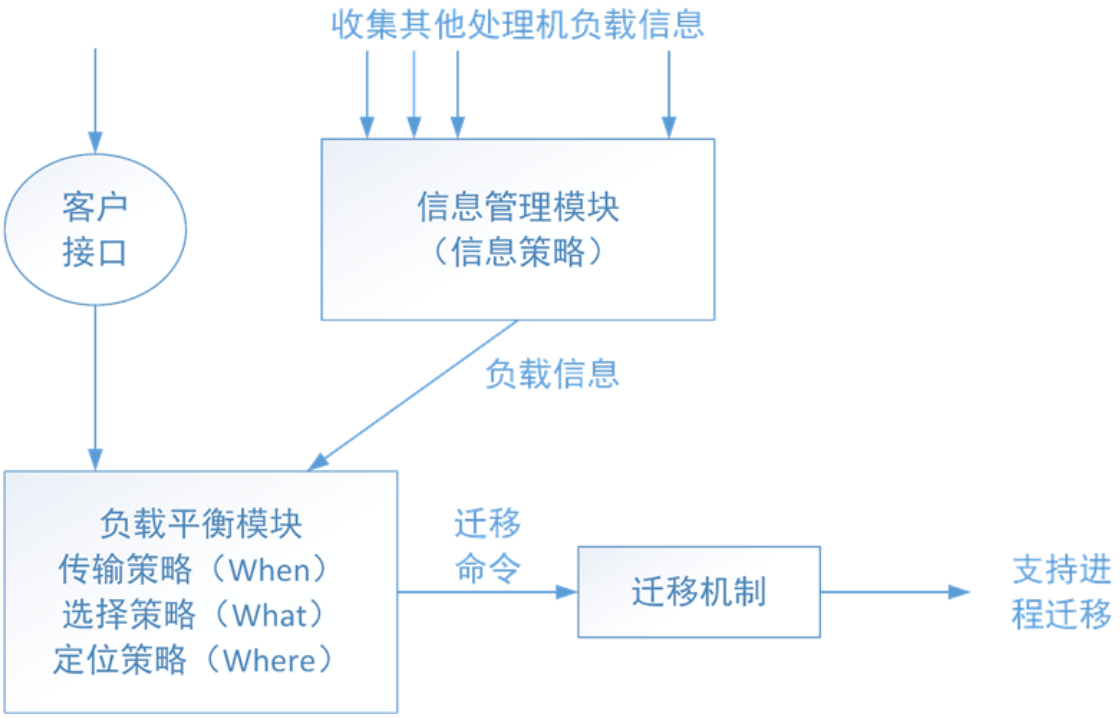
3、进程迁移的过程，动态负载平衡如何实现。

1. 进程迁移的概念：将一个正在运行的进程挂起，它的状态从源处理机节点转移到目标处理机节点，并在目标处理机上恢复该进程运行。
2. 进程迁移的步骤



1. **迁移协商。**由重负载源处理机询问目标处理机是否可以接受迁移进程。
2. **创建恢复进程。**得到目标处理机肯定答复后，在目标处理机上创建恢复进程。
3. **中断被迁移进程运行。**源处理机上的被迁进程旧实例会被中断运行，让进程停滞在一个稳定的状态下。
4. **收集源处理机上被迁进程的状态。**
5. **传输被迁进程状态。**将被迁进程状态传输到目标处理机。
6. **恢复被迁进程状态。**目标处理机上的恢复进程负责恢复被迁进程状态，重建进程实例。
7. **通告被迁进程的新位置。**通知系统内其他进程被迁进程的新位置，并重建迁移中断前的通信连接。
8. **被迁进程恢复运行。**
9. **操作转发。**利用转发计指保证进程可以在远程处理机执行。

3. 动态负载平衡：系统中重负载处理机转移一部分负载到轻负载的处理机上运行，使得整个集群系统中的所有处理机的负载趋向均衡，从而提高系统的整体效率。



- 信息管理模块：负载信息管理模块主要决定和负载平衡相关的信息策略

- 负载平衡模块：负载平衡模块决定如何用进程迁移实现负载调正的目的，依据负载信息管理模块提供的负载信息做出迁移决定。

第四章

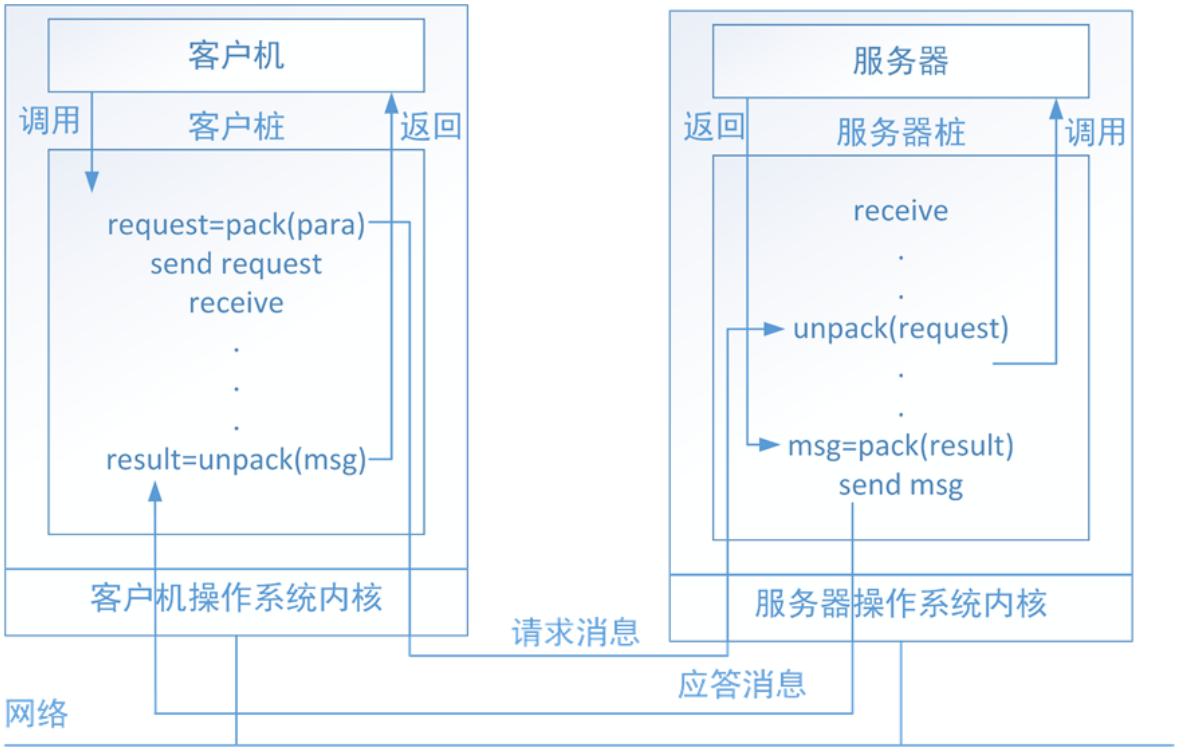
1、通信原语，消息传送的几种模式及其对比。

1. 消息传送模式中有两个基本原语：发送原语send和接受原语receive
2. 三种消息传送模式及对比
    - 同步消息传送
        - 进程P执行send，Q必须等待，直到执行receive
        - 进程Q执行receive后必须等待进程P发送消息
    - 阻塞发送/接受
        - 进程到达send时执行一次阻塞发送，无需等待对应的接受，在消息从S安全写入发送缓冲区前，发送进程不能返回
        - 进程到receive时执行一次阻塞接受，无需等待对应的发送，在消息从缓冲区接受到R之前，接受进程不会返回
        - 系统需要提供临时的缓冲区
    - 非阻塞发送/接受(消息可能被改写)
        - 进程到send执行一次非阻塞发送，无需等待对应接受，只要通知操作系统有一个消息要发送，发送进程就可以返回
        - 进程到receive时执行一次非阻塞接受，无需等待对应的发送，只要通知操作系统有一个消息要接收，接收进程就可以返回
        - 系统需要提供临时的缓冲区
    -

通信事件	同步消息传送	阻塞发送/接受	非阻塞发送/接受
发送启动条件	发送接收原语双方均达到	发送原语达到	发送原语达到
发送返回指示	消息被接受	消息已发送	消息发送启动
语义	清晰	居中间	易出错
缓冲消息	不需要	需要	需要
状态检查	不需要	不需要	需要
等待开销	高	居中间	低
通信与计算重叠	不能	能	能

2、RPC的概念，远程调用的过程，参数的传递等。

1. **RPC的概念**：(Remote Procedure Call)远程过程调用，是一种进程间通信方式。它允许程序调用另一个地址空间（通常是共享网络的另一台机器上）的过程或者函数。
2. **远程调用的过程**

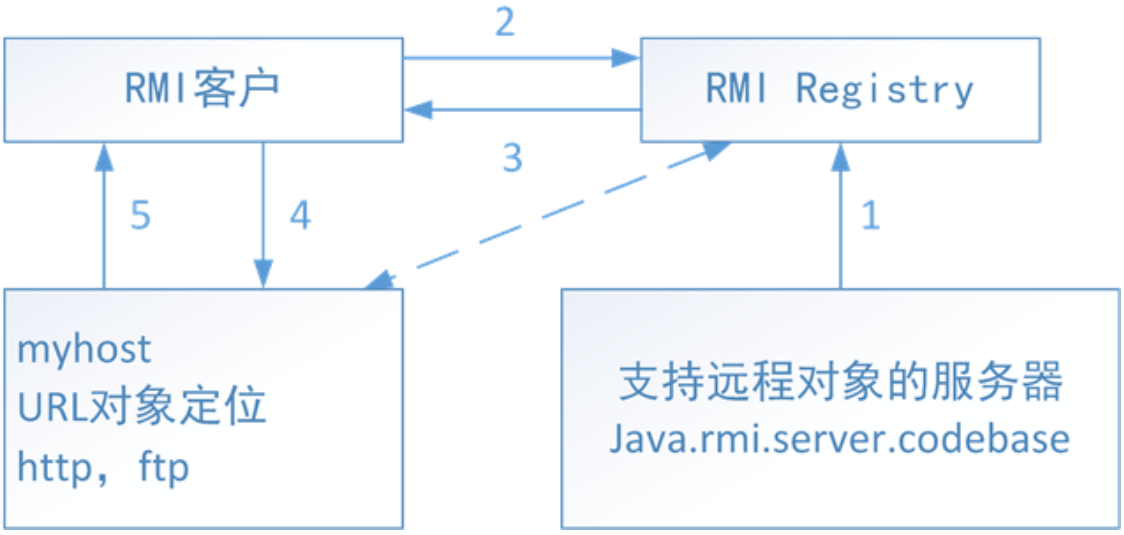


- client应用程序正常调用client stub
- client stub构造消息，通过trap进入操作系统内核
- 内核将消息发送到server的操作系统内核
- server内核将消息交给server stub
- server stub将消息解包，用相应参数调用服务例程
- 服务例程进行计算，将结果返回server stub
- server stub构造消息，通过trap进入内核
- 内核将消息发送到client的核心
- client内核接受消息并将它交给相应stub
- stub解包，将调用结果返回应用程序

3. 参数传递
    - 值参数传递：简单参数，客户将这些参数的值直接传递给服务器
    - 引用参数传递：引用参数是指针。
    - 参数的规范形式
    - 桩的生成：在完整地定义了RPC协议之后，协议实现客户桩和服务器桩

3、RMI的概念，调用过程，参数传递等。

1. RMI的概念：(Remote Method Invocation)远程方法调用，是Java特有的分布式计算技术。本质上是通过Java编程语言扩展了常规的过程调用，在网络上不仅可以传送对象数据，而且可以传送对象代码。
2.



1. 远程对象注册与名字绑定
2. 客户按名字查找远程对象
3. 注册器返回远程对象接口
4. 客户从codebase请求stub类
5. http服务器返回远程对象的stub类

3. 参数传递

第五章

1、逻辑时钟的概念，Lamport时间戳、向量时间戳。

1. 逻辑时钟的概念：计算机内部各时钟一致，而不是是否与真实时间接近。更进一步，只需跟踪事件的顺序
2. Lamport时间戳：
    1. 阐明时钟同步时可能的，指出时钟同步不需要绝对同步
    2. 如果两个进程无关，时钟无需同步，不会产生问题
    3. 重要的不是所有进程在时间上完全一致，而是事件发生顺序上达成一致
3. 向量时间戳：

    向量时间戳让每个进程维护一个向量V

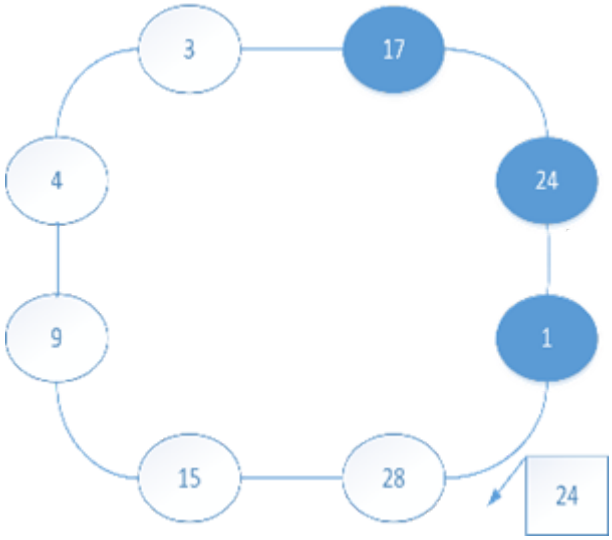
    - 向量V是一个数组元素，数组元素个数是系统的进程数，元素是升序的整数
    - $V_i[i]$是目前为止进程P_i发生的事件的数量
    - $V_i[j]=k$，表示进程P_i知道进程P_j中已经发生了k个事件

2、常用选举算法。

1.环选举算法

基于环的选举算法，适合安排成逻辑环的进程集合。假定选举过程中不会出现失效，通信是异步的。算法的目的是选举一个进程作为协调者，**这个进程具有最高的编号。**

这个选举算法的代价是要转发3N-1次消息。



① 初始，每个进程标记为选举未参与者。

② 选举发起者将自己标记为选举参与者，将自己的标识符置于选举消息中，并顺时针传送给下一个邻居（进程）。

③ 当进程收到一个选举消息，它将选举消息中的标识符与它自己的标识符进行比较，如果消息中的标识符大，进程转发这个选举消息到它的下一个邻居，并将自己标记为选举参与者；

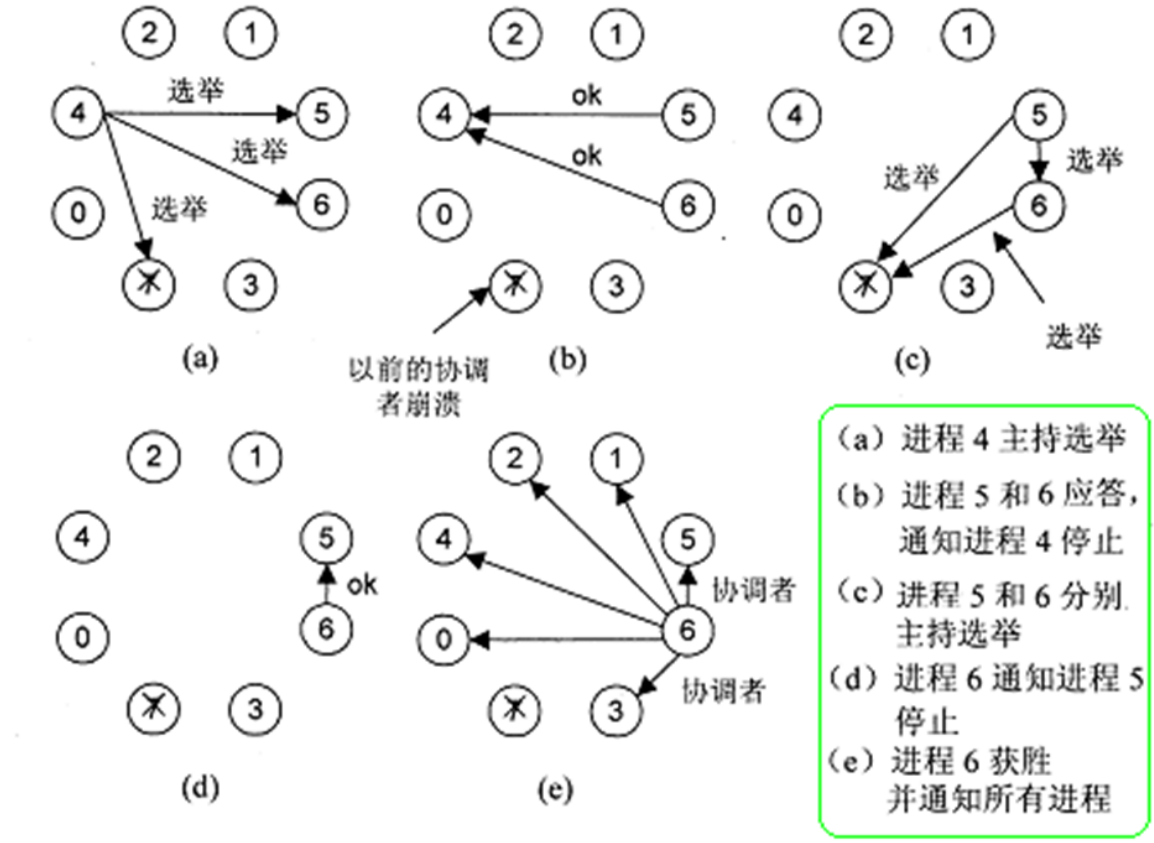
④ 如果消息中的标识符小且接收进程不是选举参与者，接收进程用它的标识符替换消息中的标识符，并将选举消息转发到下一个邻居,自己也成了选举参与者；

⑤ 如果接收进程已经是选举参与者，它只转发选举消息。直到接收进程的标识符与选举消息中携带的进程标识符相等为止。

⑥ 标识符中标识符最高者便是新的协调者。

2.欺负算法

允许进程在选举过程中失效，通过超时机制来检测进程失效。启动选举的进程只与编号高于它的进程通信。



将进程进行排序

① P向高的进程发送选举消息E(lection)

② 如果没有响应，P选举获胜

③ 如果有进程Q响应应答消息OK，则P结束，Q接管选举并继续下去。

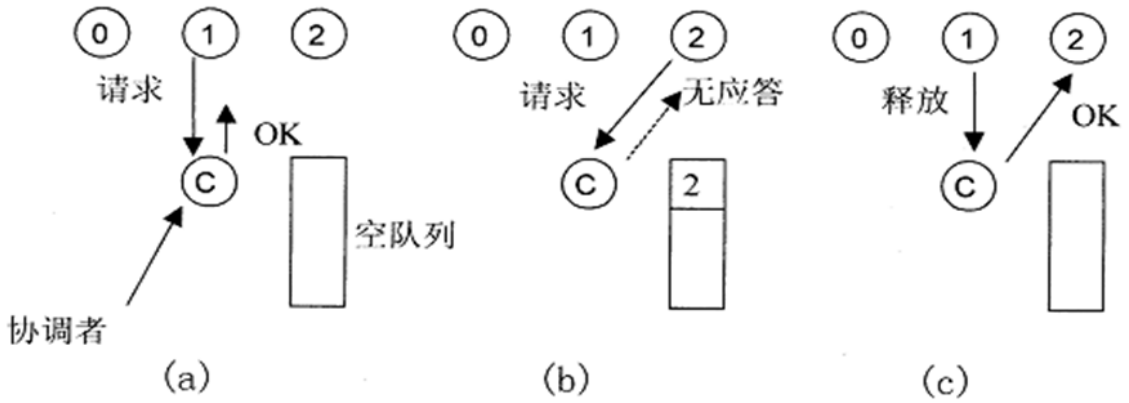
④ 选举获胜者向其他进程发送协调者消息C(orrdinator)

3、几种常用的互斥算法，能灵活运用，并根据需求改进。

- 集中式算法
- 分布式算法（基于时间戳算法）
- 令牌环算法（基于令牌算法）
- 基于事件优先权算法
- 共享K个相同资源算法

1. 集中式互斥算法

- 选一个进程作为协调者
- 无论什么时候进程要访问资源，都需要向协调者发送请求信息并获得允许
- 如果当前资源可访问，就发送允许信息



2.分布式算法（基于时间戳）

Lamport算法

- 需满足的要求
  - 拥有资源的进程首先要放弃该资源，其他进程才能使用它
  - 对资源的请求是按他们提出的次序予以批准的
  - 被批准访问资源的进程最终能释放该资源
- 算法的内容：

1. **1.Pi 资源请求消息Request（Ti ： P i）发送**
2. **2.Pj收到Request（T i：P i），按T顺序置于其消息队列，如果没有资源请求或请求时间晚于收到消息的时间戳，回应Reply（T j：P j）**
3. **3.Pi被批准使用临界资源条件：有请求，且Ti最小（消息全定序）；Pi接收了所有晚于Ti的消息（包括应答）**
4. **4.Pi释放临界资源，删除（Ti ： P i），发送Release（T j+1：Pi）**
5. **5.Pj收到Release后，删除（Ti ： P i）**

- **算法的问题**：集中式算法的单点故障被它的N点故障所取代，（当一个进程崩溃时，不能对请求做出应答，且需要花费更多的带宽
- 解决办法：收到消息后，无论赞成还是反对都返回一个应答消息，可以使用超时机制来确定是否崩溃

Ricart_Agrawala算法

- 1.Pi 资源请求消息Request（T_i ： P_i）发送
- 2.Pj收到Request（T_i：P_i），按T顺序置于其消息队列
  - 如果没有资源请求或请求时间晚于收到消息的时间戳，回应Reply（T_j：P_j）；否则推迟返回应答消息
  - 进程从临界区退出，向需要请请求资源的进程补发应答消息
- 请求进程从竞争进程得到应答小Reply（T_j：P_j），便可进入临界区

2（N -1）个消息（Ricart）

3（N-1）个消息（Lamport）

3.令牌环算法（基于令牌算法）

- 构造一个逻辑环
- 获得令牌后才可以决定是否访问资源
- 不可以用同一令牌立即再访问
- 问题：检测令牌丢失困难

4.基于事件优先权算法

完全可靠网络算法

- 请求队列P、Q
  - Q队列放置其他节点送来的请求（接收令牌）
  - P队列放置其他节点来不及处理的随令牌转来的请求
- 算法过程
  - *i* 希望进入，发送*Request（i，P（i））*，并将*（i，P（i））*存入接收进程的Q队列
  - *j* 退出 根据P、Q队列情况判断（标注最高优先权进程，合并队列）
  - *j* 将令牌和新的P队列发送到所标注的最高优先权进程

不可靠网络算法

- 算法基本与可靠网络互斥算法一致
- 进程将令牌和P队列发送指定进程，并启动超时机制。
- 在超时范围内，如果没有收到指定进程的应答，进程从队列删除该进程，同时将令牌发给下一个进程。

共享K个相同资源的算法

- K个相同资源副本被N个竞争进程共享？
- Raymond算法
  - N-1个竞争进程中，少于K-1个进程在临界区中
  - 则另一个进程可以进入临界区
  - 进入临界区，需要得到N-1-（K-1）=N-K个应答消息

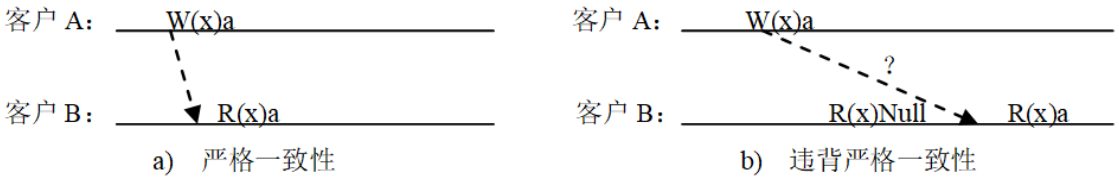
第八章

1、几种一致性模型，每种模型的具体表述，能绘图说明。

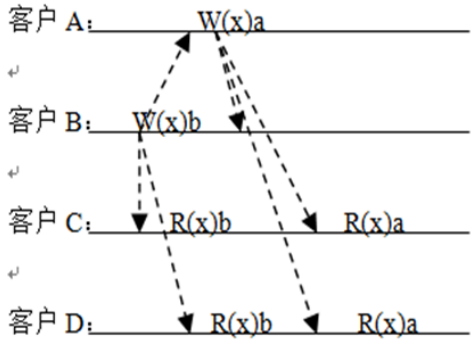
一致性模型是数据存储与访问数据存储的进程之间的一种契约，遵守契约，则数据存储正确工作

数据为中心的一致性模型

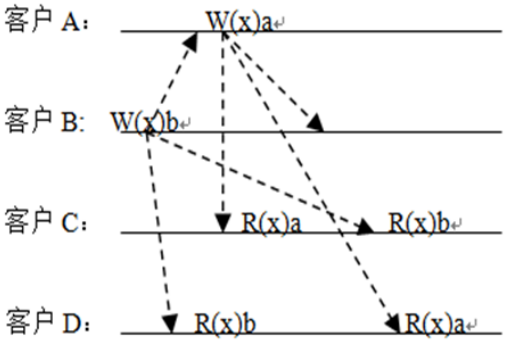
- 严格一致性



  - 最强的一致性模型
  - 对数据项的读操作返回的值应是该数据项最近写入的值
  - 绝对全局时钟和即时传播
  - 在分布式数据存储中不可能实现
- 顺序一致性



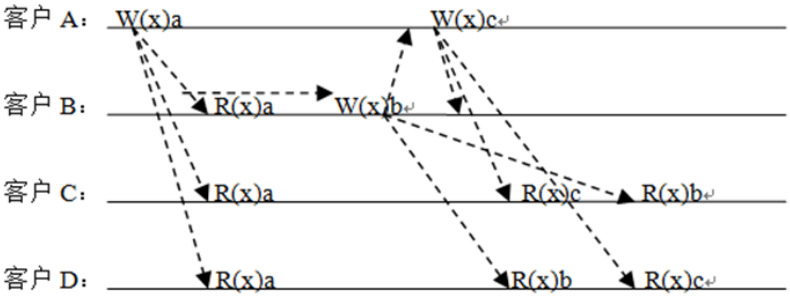
a) 顺序一致性



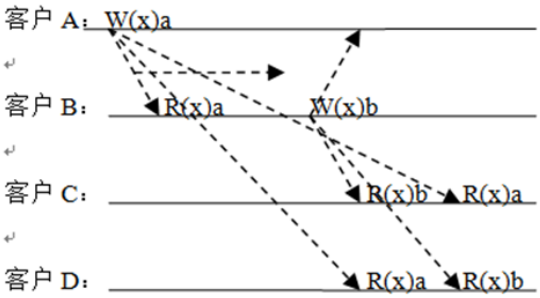
b) 违背顺序一致性

- 放弃了时间定序的要求
- 任何对数据存储的一组操作执行效果是相同
- 所有客户以同样的次序看到所有写操作的全局定序

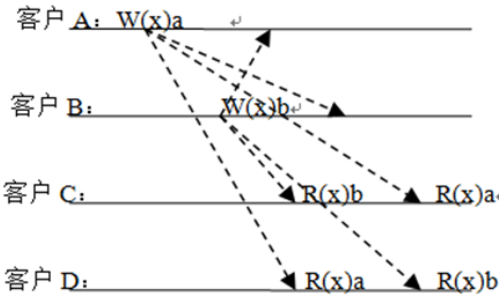
- 因果一致性



a) 有效因果一致性定序

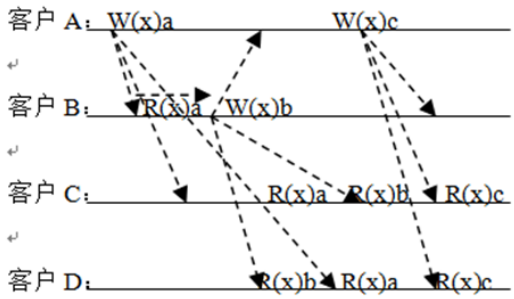


b) 违背因果一致性定序

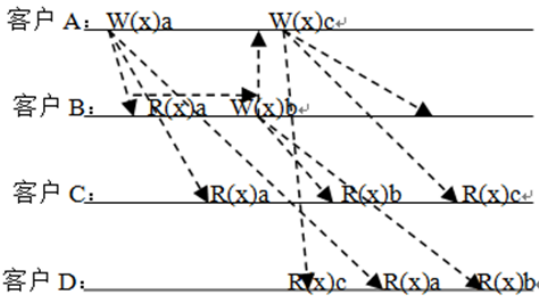


c) 有效因果一致性定序

- FIFO一致性



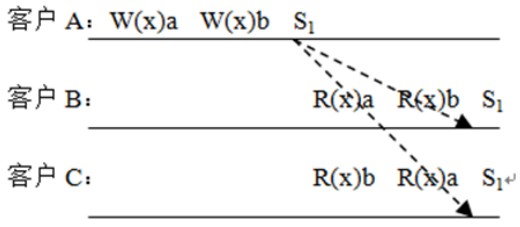
a) 有效 FIFO 一致性定序



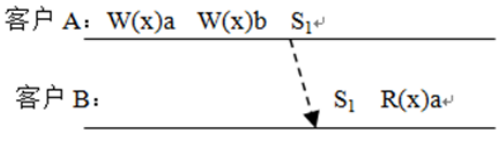
b) 无效 FIFO 一致性定序

- 一个客户的写操作定序在所有副本上是相同的
- 也称为管道RAM模型，BCD都看到A先执行W(x)a, W(x)c

- 弱一致性



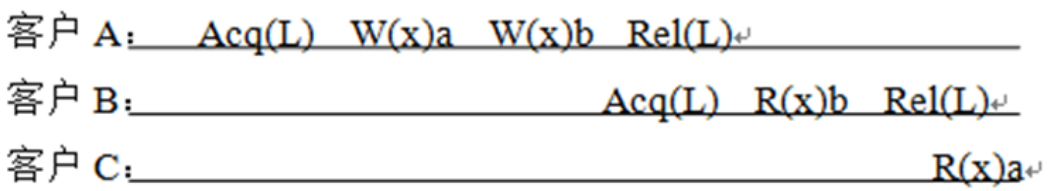
a) 弱一致性定序



b) 非弱一致性定序

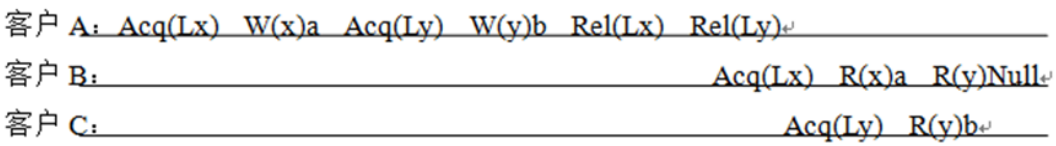
- 采用按一个操作组，而不是单个操作进行一致性定序，通过同步变量S，来同步数据存储的所有副本

- 释放一致性



  - 获取操作、释放操作

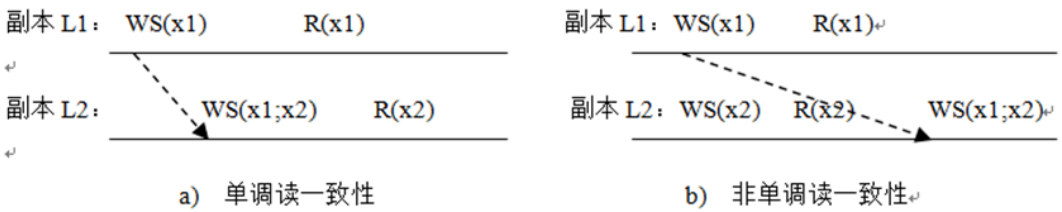
- 入口一致性



  - 数据项一次操作与同步变量相关联优惠

客户为中心的一致性模型

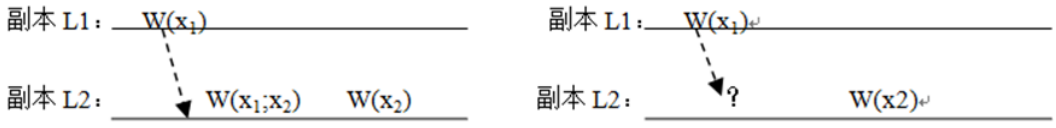
- 单调读



a) 单调读一致性　　b) 非单调读一致性

  - 如果一个进程读数据项x的值，该进程的任何后续对x的读操作总是返回前一次读同样的值或更加新的值

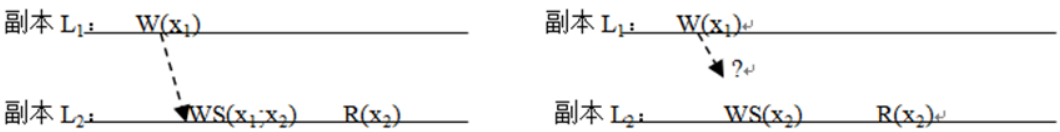
- 单调写



a) 单调写一致性　　b) 非单调写一致性

  - 一个进程对数据项x执行写操作，必须在该进程对x执行任何后续写操作之前完成

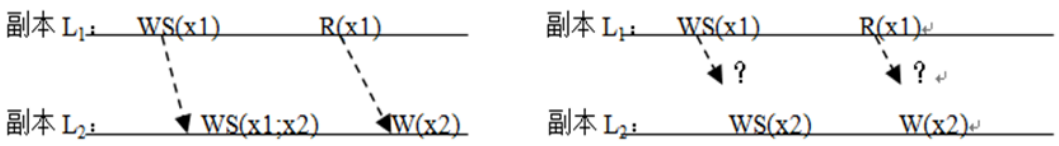
- 写后读



a) 写后读一致性　　b) 非写后读一致性

  - 一个进程对数据项x执行一次写操作的结果，总是会被该进程对数据项x的后续读操作所看见，无论读操作发生在哪个副本上

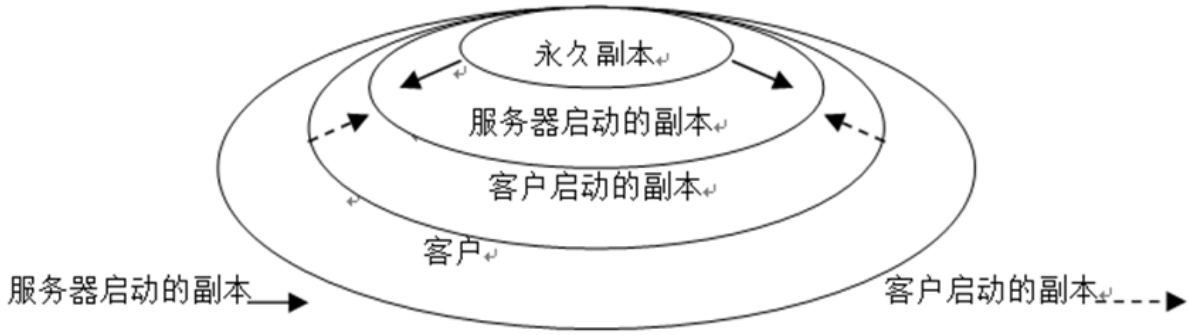
- 读后写



a) 读后写一致性　　b) 非读后写一致性

  - 一个进程对数据项x的写操作是跟在同一进程对x读操作之后，保证相同的或更加新的x的值能被看见。
  - 进程对数据项x所执行的任何后续写操作总是在x的副本上执行，而该副本是用该进程最近读取的值所更新。

2、分发协议，几种不同副本的概念，作用等。



永久副本

概念：永久副本是由数据存储的拥有者创建和管理，且作为数据拥有者的数据固定存储。

作用：满足容错要求。

服务器启动的副本

概念：服务器启动的副本是应数据存储拥有者的请求根据永久副本而创建的，常常放置在另外的服务器中。

作用：增强系统的性能。

客户启动的副本

概念：这类副本是客户创建的副本，即客户启动的副本。

作用：提高系统的性能，缩短访问数据的时间。

3、更新传播的几种方式及其对比。

传播更新通知：无效化协议

- 只传播一个简短的数据无效通知，不包含其他信息。
- 写操作对读操作的比率很高时，传播效果好。
- 几乎不占用网络带宽，当写操作对读操作的比率高时效果好

传播更新数据：读操作对写操作的比率很高时

- 在副本间传送被修改过的数据。
- 读操作对写操作的比率很高时，传播效果好。

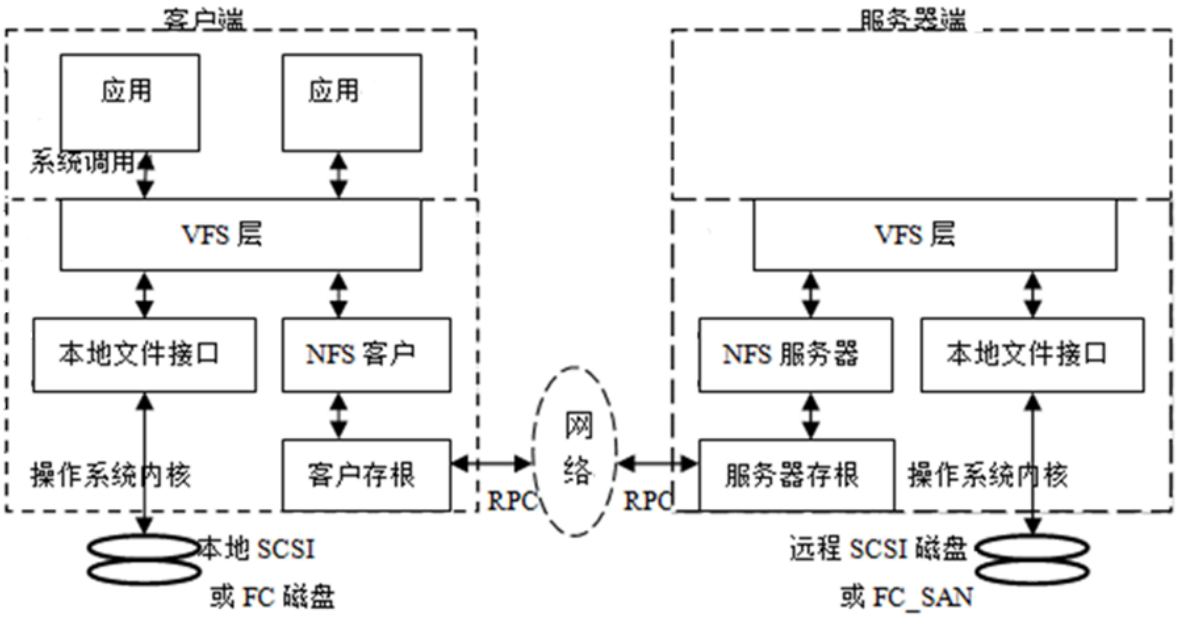
传播更新操作：主动复制

不传播被修改过的数据，而是告诉各副本应该执行的操作。要求每个副本有一个进程来执行更新操作，主动地保持各副本关联数据的一致性。更新操作所关联的参数较少时，所占带宽较小。

第九章

1、网络文件系统的相关内容。

NFS(Network File System)分层结构体系



名字空间

- NFS客户能够在自己本地文件系统中安装（挂载）一个远程文件系统或其一部分
- 当服务器允许客户使用其一个目录及该目录项时，称服务器输出（export）该目录
- 一个文件的名字依赖于客户组织其本地名字空间的方式
- 一台NFS服务器自身可以安装来自其他服务器的输出目录，但不能再将这些目录输出给客户

文件句柄

① 持久文件句柄(Persistent file handle)：

服务器为文件系统对象生命期规定的一个固定值

[文件系统标识符 | 文件i-node号 | i-node生成数]

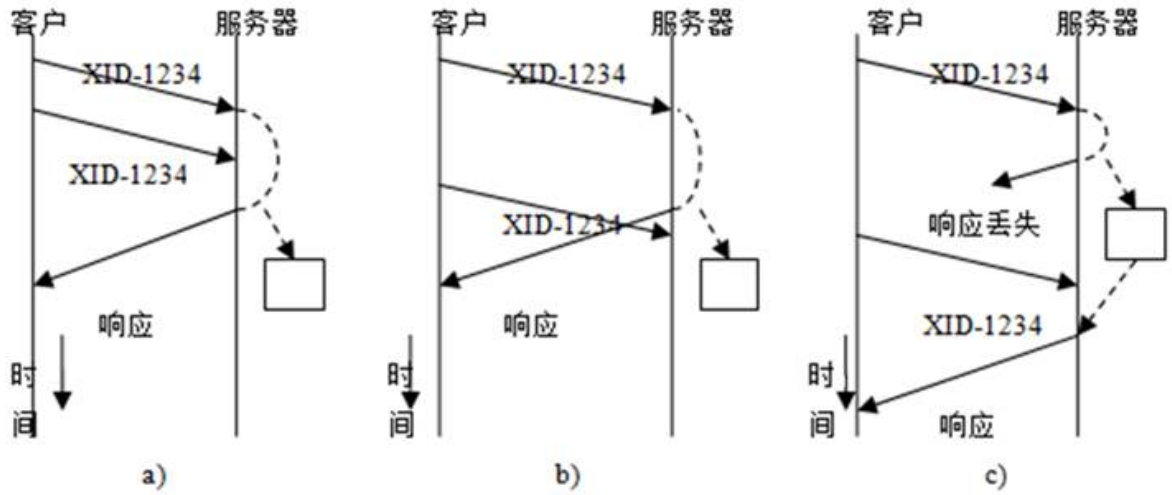
② 可变（挥发）文件句柄(Volatile file handle)：

挥发文件句柄对客户是不透明的，服务器在不同的时间点上可令其无效，它可能包括下面的信息。

[挥发位 = 1 | 服务器自举时间 | 槽 | 生成数]

##

2、NFS RPC重传的几种情形。



重传过早----------重传过晚----------响应丢失

三种RPC重传处理情况：

① 如果文件服务器还没有处理完原先的RPC请求，则简单的忽略这个重传的请求。

② 服务器刚返回响应就收到了重传请求，时间非常接近，服务器就断定重传请求和原先的响应相互交叉，服务器忽略重传请求。

③ 响应确实丢失了，服务器从缓存中找到原先的响应，作为RPC重传请求的响应发送给客户。

3、AFS/Coda实现客户端缓存一致性的方法。

客户端缓存

Coda采用上载/下载模型，在客户端总是缓存整个文件或文件集。

服务器复制

① Coda允许以文件卷为单位将文件复制到多个服务器上。拥有一份卷副本的服务器集合称为该卷的卷存储组（Volume Storage Group，VSG）

② 客户能访问到的（Accessible VSG），AVSG为空，该客户断开

③ 读是从一个AVSG成员读，更新时并行传送给AVSG所有成员。

④ 版本记录解决冲突

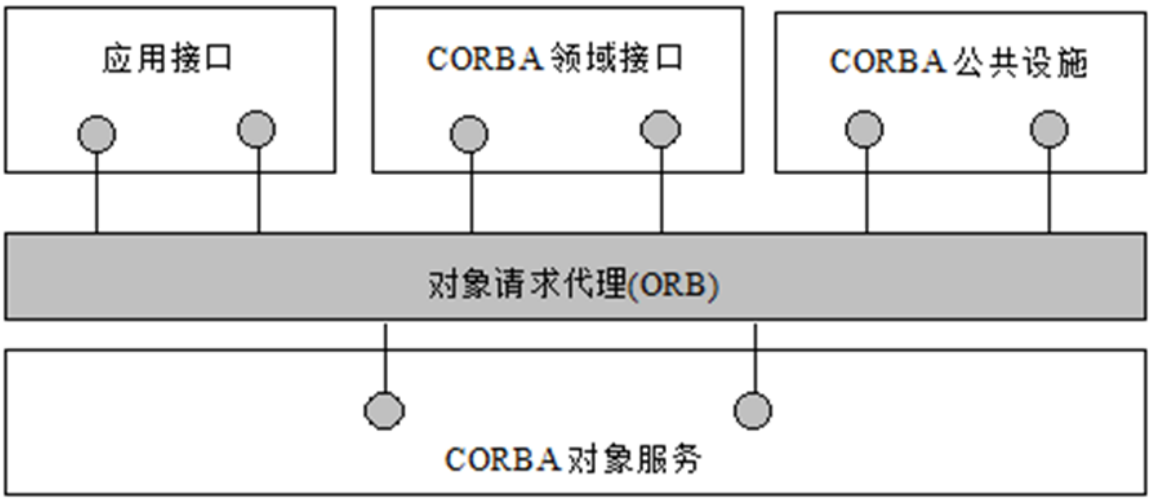
断开操作

如果对于一个卷，客户的AVSG是空的，该客户与卷之间是处在断开状态。此时，客户端不能联系任何持有该卷副本的服务器。在此情况下，大多数文件系统将不能继续工作。Coda采用一种不同的方法，客户将继续使用文件的本地副本工作，这个本地缓存的副本是客户在服务器上打开该文件时获得的。

第十一章

1、OMA参考模型，CORBA的组成，几种接口的作用。

OMA参考模型



ORB和GIOP

- ORB(Object Request Broker)对象请求代理：它提供了网络环境无关性、操作系统无关性和开发语言无关性的公共平台。
- CORBA远程对象方法调用依赖通过对象请求协议GIOP，解决异构环境下分布式对象互操作问题
- 当前普遍使用IIOP(Internet Inter_ORB Protocol)是运行在TCP/IP协议层之上的GIOP

CORBA对象服务

- CORBA的公共对象服务是独立于应用领域，为使用和实现对象而提供的基本服务集合
- 在构建分布式应用时，经常会用到这些服务

CORBA公共设施(Common Facilities)

- 提供了一组更高层的函数，包括用户界面、信息管理等方面的通用设施，为终端用户应用提供一组共享服务接口，例如组合文档系统管理和电子邮件服务等。

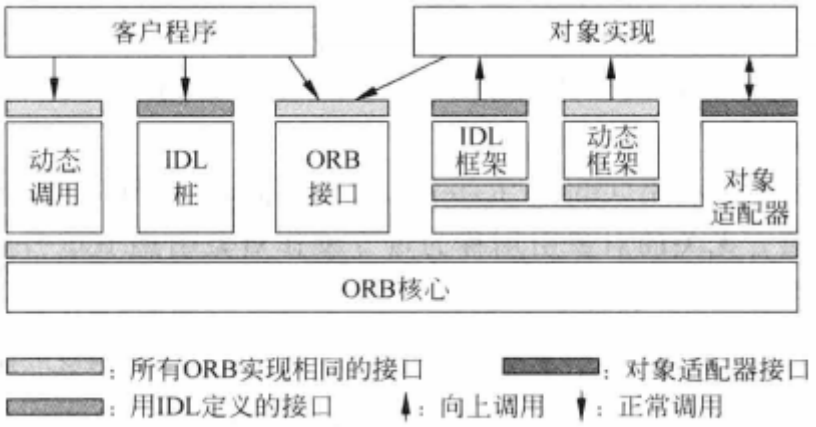
CORBA领域接口(Domain Interface)

- 与特定的应用领域有关，如为制造业、金融业、通信行业等领域服务提供接口。

应用接口(Application Interface)

- 由销售商提供的、可控制其接口的产品

CORBA的组成



- ORB 对象请求代理
    - 作为“软件总线”来连接网络上的不同对象

    - 提供对象的定位和方法调用
- OA 对象适配器-
    - 用于构造对象实现与ORB之间的接口
    - 给框架发送方法调用
    - 支持服务器对象的生命周期(对象的创建和删除)
- BOA 基本对象适配器
    - 当客户请求对象服务时，负责激活对象
- POA 可移植对象适配器

几种接口的作用

① **ORB核心**：实现对所有ORB都相同的接口。

② **对象适配器接口：**连接对象实现和ORB。

③ **接口定义语言(IDL)和静态接口**：客户用ORB携带的IDL编译器，编译对象接口IDL文件，生成特定编程语言的Stub和Sleketon代码。

④ **动态接口**：在预先不知道服务对象接口的情况下，客户通过查询或采用其他手段获得服务对象的接口描述信息，然后使用动态调用接口（DII）来构造客户请求，并发送给对象实现。在对象实现方，可用动态骨架接口(DSI)动态分发用户请求的机制，以便动态处理客户方的请求。

2、IDL，能将IDL转换为Java的接口。

1. IDL: 为了保持CORBA的商业中立性和语言中立性，必须要有一个中介，存在于像C++ CORBA服务器代码和JAVA CORBA客户机这样的实体之间，这就是IDL（接口定义语言）。IDL语言是中立的，因此可以利用任何已经作了IDL映射的程序设计语言。
2. 将IDL转换为JAVA接口

   ① 指定生成文件的路径

   idlj [–td c:_work\corbasem] calculator.idl

   ② 生成客户端的类

   idlj -fclient calculator.idl (default)

   ③ 生成服务器端的类

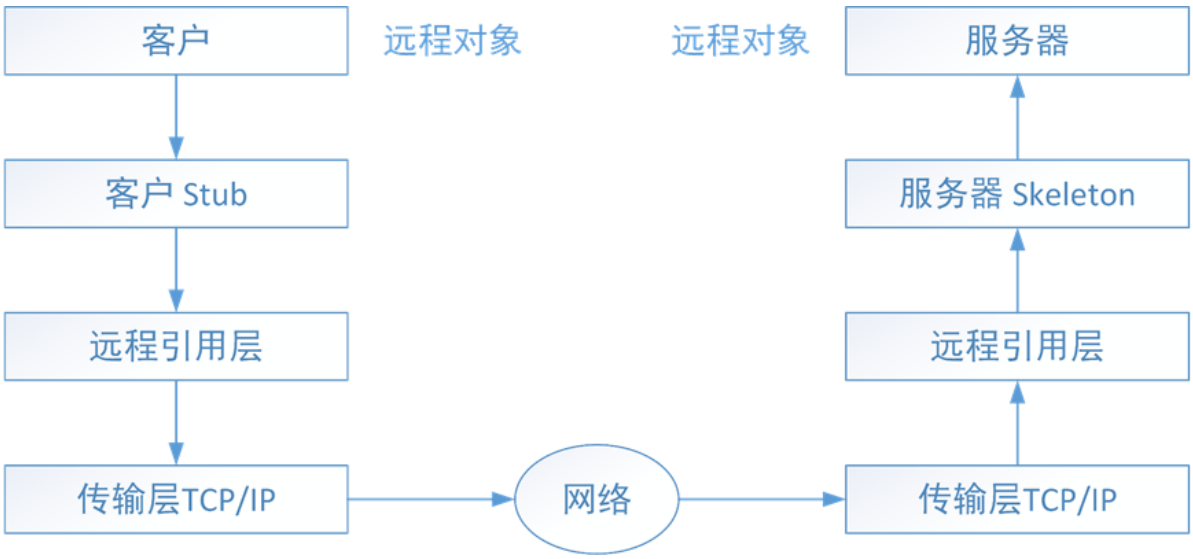
   idlj -fserver calculator.idl

   ④ 生成客户端和服务器端的类

   idlj -fall calculator.idl

3、RMI体系结构及应用开发。

RMI体系结构



客户及服务器方都包括三层抽象：桩/骨架层，远程引用层和传输层。

应用开发

分布式应用开发步骤：

① 设计和实现你的分布式应用的组件。

（定义远程接口—>实现远程对象—>实现客户端）

② 编译源程序并生成头文件。

③ 通过网络访问得到所需要的类。

④ 启动应用程序。

第十二章

1、Web服务的概念

1. Web服务是指在网络上发布的一些应用程序接口，可供网络中其他应用程序访问和调用。

2、什么是Web服务契约，服务功能描述、服务访问描述、服务位置描述的组成。

1. Web服务契约：契约是供求双方间进行交换的一种约定。在面向服务的分布式计算系统中，契约是系统之间交换数据时应遵守的约定。
2. 服务功能描述、服务访问描述、服务位置描述的组成

   ① 服务功能描述(What)

   端口类型（接口）定义　操作定义　消息定义　类型定义　策略定义

   ② 服务访问描述(How)

   端口类型（接口）绑定　操作绑定　消息绑定　策略定义

   ③ 服务位置描述(Where)

   服务定义　端口定义　地址定义　策略定义

3、SOAP、WSDL、UDDI的概念

1. SOAP(Simple Object Access Protocol)简单对象访问协议是基于XML的一种通信协议，与平台和引用编程语言无关。
2. WSDL(Web Service Description Language)Web服务描述语言是基于XML的。WSDL文档用来描述和定位一个Web服务。WSDL文档是在Web服务开发时，随着Web服务由开发平台和工具自动生成的。用于描述某个Web Service，规定了服务的位置，服务提供的操作或方法。
3. UDDI 通用描述、发现和集成是Web服务提供者和消费者之间的接洽点

4、Web服务应用开发

1. Web服务流程

   ① 注册

   ② 查询

   ③ 查询响应

   ④ 请求web服务

   ⑤ 返回web服务描述

   ⑥ 根据web服务描述调用选定的服务