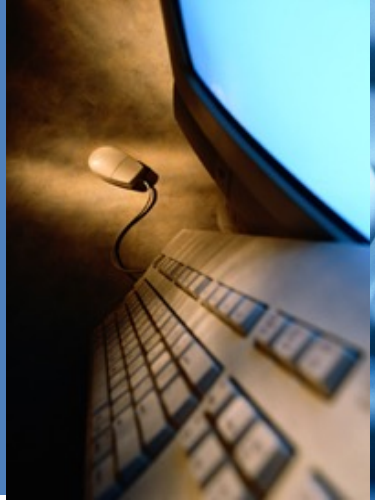
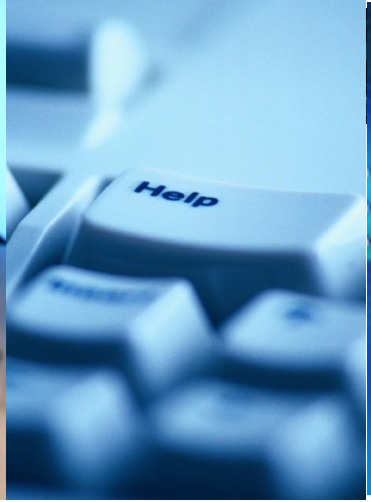



分布式系统
Distributed Systems

陈鹏飞
计算机学院
chenpf7@mail.sysu.edu.cn
办公室：学院楼310
主页：http://sdcs.sysu.edu.cn/node/3747

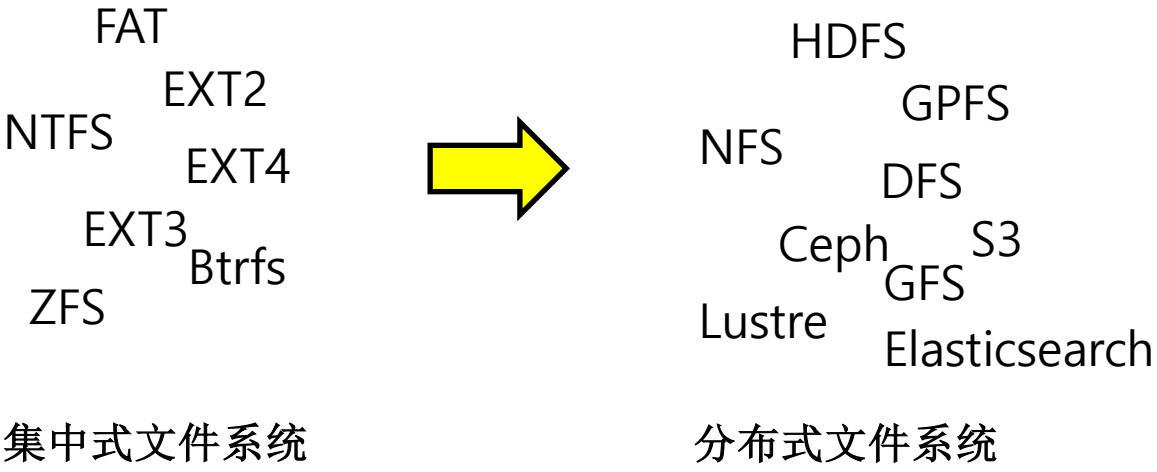
第十讲 — 分布式文件系统



问题

如果有一笔钱，可以用来**购买商业软件**，你会把钱用来**购买计算型**的软件、**存储型**的软件还是**网络型的软件？**

背景



共享数据是分布式系统的基础



分布式文件系统

➢ 重要属性

- 通过网络互连，文件共享；
- 分布式：文件系统的客户端、服务器、存储分散在不同物理机器上；
- 透明性：对于用户来讲，使用起来是就像是集中式文件系统；
- 性能：文件的访问时间应该满足QoS的要求；
- 并发的文件更新，支持多用户；



体系结构

➢ 客户-服务器体系结构

- 大多数分布式文件系统遵循客户-服务器体系结构；
- 典型代表是NFS（Network File System）；

➢ NFS

- 每个文件服务器都提供其本地文件系统的一个标准化视图；
- 每个NFS服务器都支持相同的模型；
- 底层模型是远程文件访问模型；

体系结构

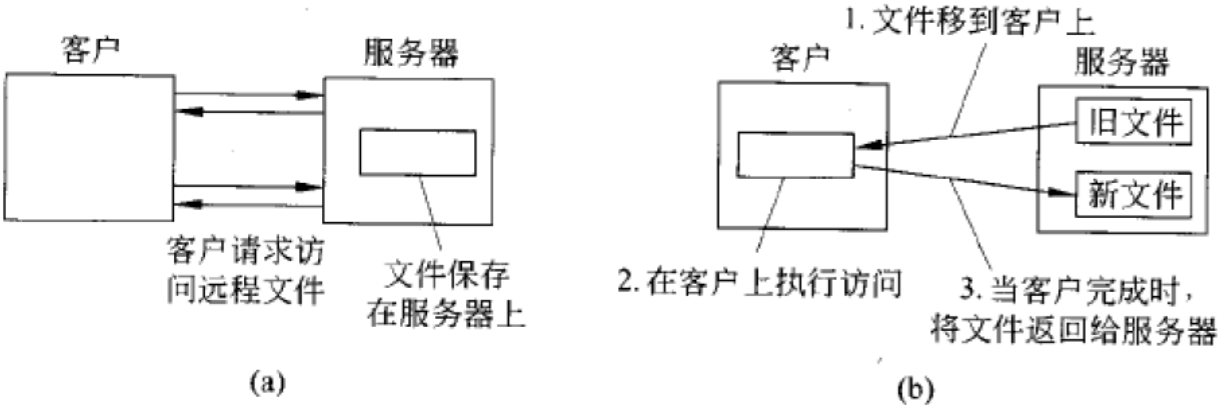


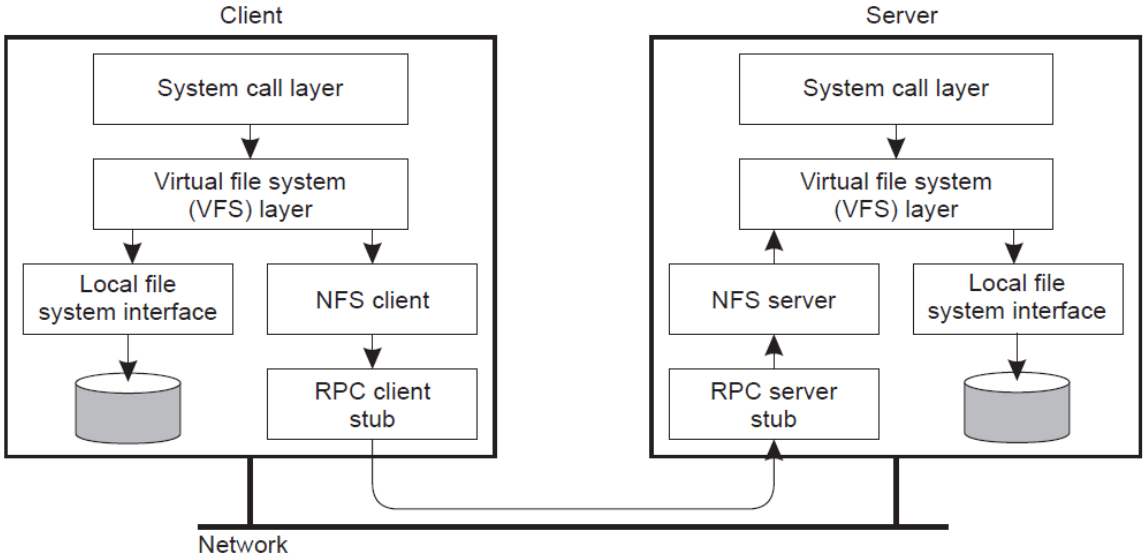
图 11.1

（a）远程访问模型；（b）上载/下载模型

两种不同远程文件访问方式比较

NFS架构

- NFS采用VFS (Virtual File System)实现，VFS是不同文件系统接口事实上的标准， 现代OS都提供VFS。VFS提供了系统结构，屏蔽了访问本地和远程文件的差异性。



NFS文件系统模型

- NFS提供了与UNIX系统完全一样的文件系统模型；
  - 分层组织成命名图；
  - NFS像任何UNIX文件系统一样支持硬链接和符号链接；

操作	版本 3	版本 4	描述
create	有	无	创建一个常规文件
create	无	有	创建一个非常规文件
link	有	有	创建一个文件的硬链接
symlink	有	无	创建一个文件的符号链接
mkdir	有	无	在给定目录下创建一个子目录
mknod	有	无	创建一个特殊文件
rename	有	有	更改文件名
remove	有	有	从文件系统中删除一个文件
rmdir	有	无	从一个目录中删除一个空的子目录
open	无	有	打开一个文件
close	无	有	关闭一个文件

基于集群的分布式文件系统

- 当处理非常大的数据集合的时候，传统的客户端-服务器方法就不再适合了；
- 解决方案一: **加速文件访问速度，应用文件分片划分技术**使文件可以并行访问；

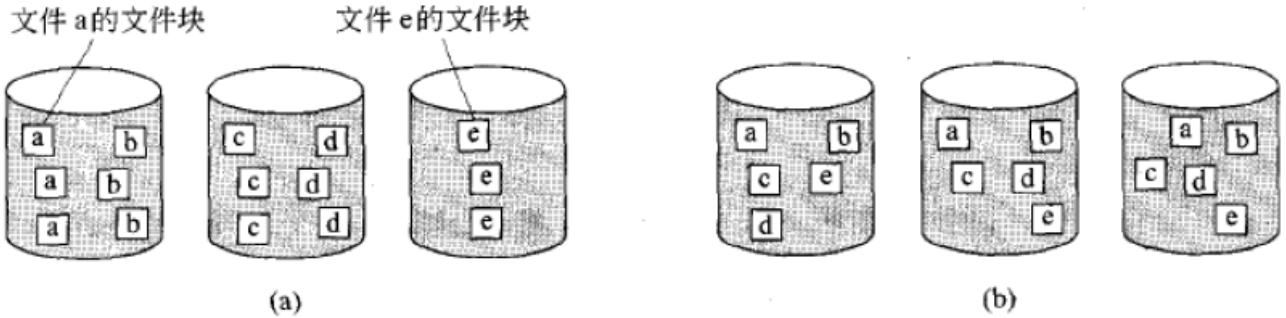
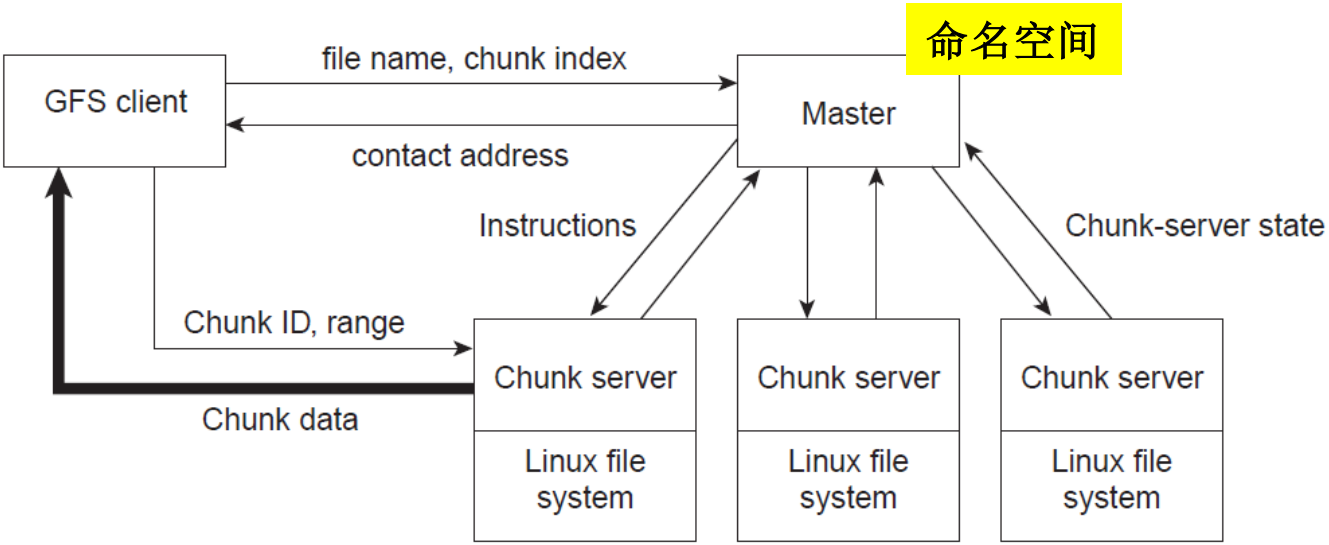


图 11.4 用于分布文件的两种方法之间的区别：
(a) 把整个文件分布在多个服务器上；(b) 对文件进行带区划分以进行并行访问

基于集群的分布式文件系统

- 解决方案二: 将文件分成较大的数据块如64MB，然后将数据块分布、复制到多个物理机器上（Amazon、Google）；



Google文件服务器的组织结构



GFS的特点

- 主服务器只是在内存中维护了一张（文件名，块服务器）的映射表 => 最小化IO， 以日志的形式保存对数据的更新操作，当日志过大时将创建检查点，存储主存数据；
- 大量的实际工作是块服务器完成的。文件采用主备模式复制，主服务器避开循环；
- 上述两个特点决定了GFS主服务器不会构成瓶颈，为系统带来重要的可伸缩性，单个主服务器可以控制数百个块服务器；

对称式体系结构

- 基于对等（P2P）技术的完全对称的体系结构；

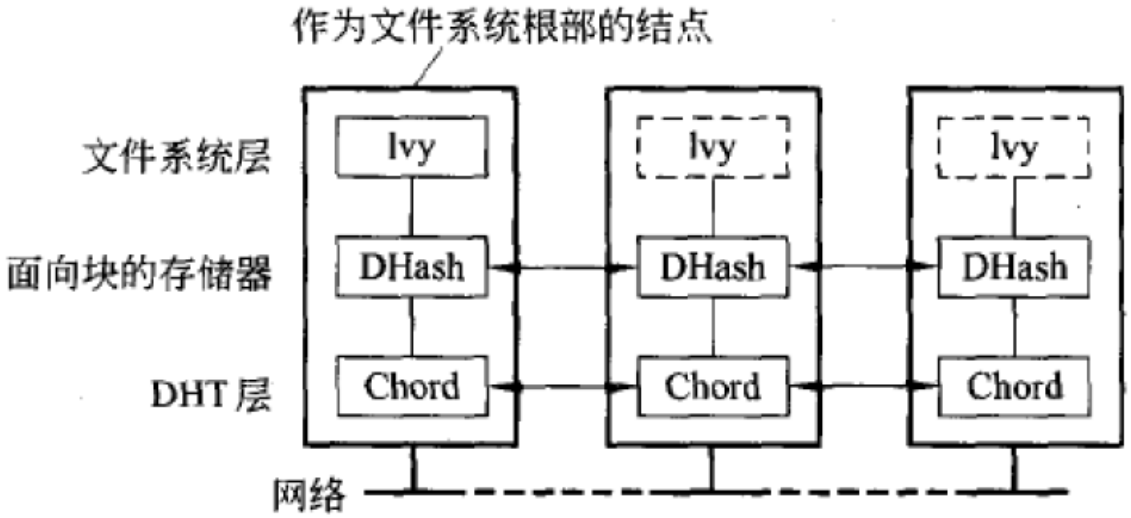


图 11.6 Ivy 分布式文件系统的组织结构

为什么不自己实现一个分布式文件系统？



文件系统中的进程

- 无状态的NFS：

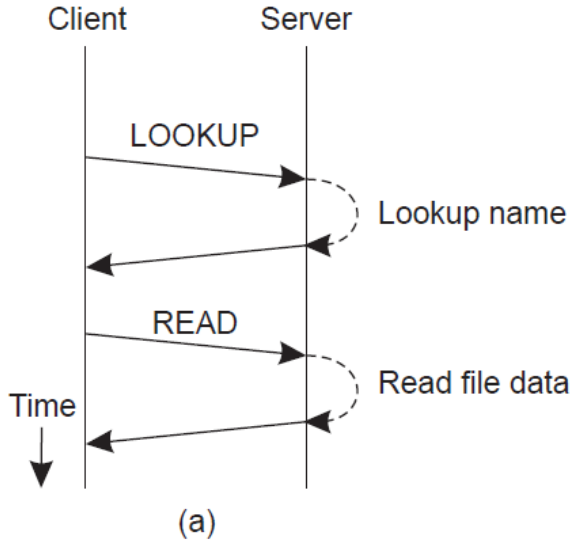
  实现简单， 服务器崩溃后，不需要恢复阶段；无法锁定访问文件；

- 有状态的NFS：

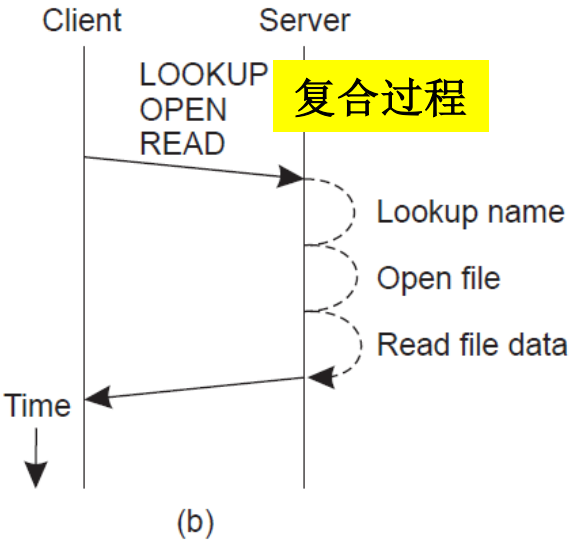
  需要服务器维护关于客户端的大量信息；

分布式文件系统中的RPC

- 选择RPC是为了使系统独立于底层操作系统、网络和传输协议；但是当需要跨网络访问文件时（存在性能问题），则需



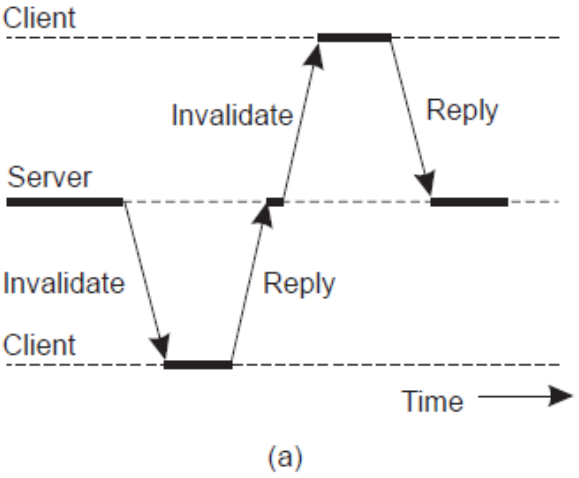
(a)



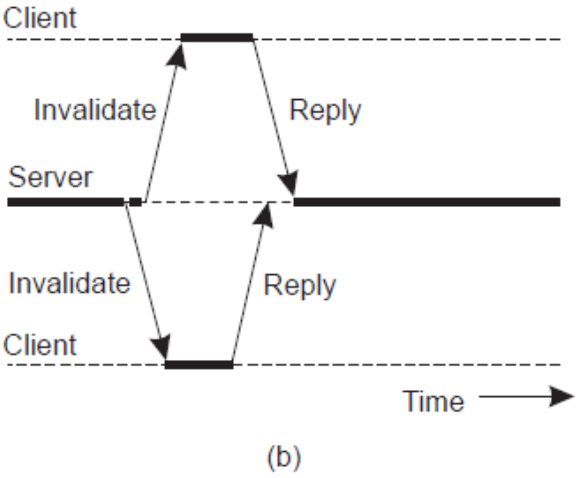
(b)

分布式文件系统中的RPC

- 当处理文件副本特别是客户端的副本更新时，服务器顺序地发送副本失效指令显然效率不高；



(a)



(b)

并行RPC

NFS中的命名

- 基本原则：
  - 允许客户完全透明地访问服务器维护的远程系统；
  - 实现方式：在本地文件系统中挂载远程文件系统；

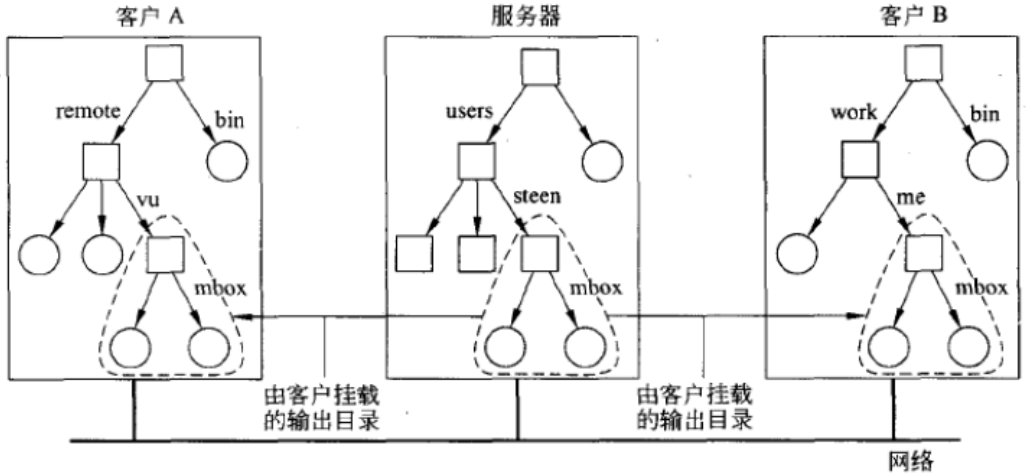


图 11.11 在 NFS 中挂载远程文件系统(的一部分)

NFS嵌套挂载

不允许

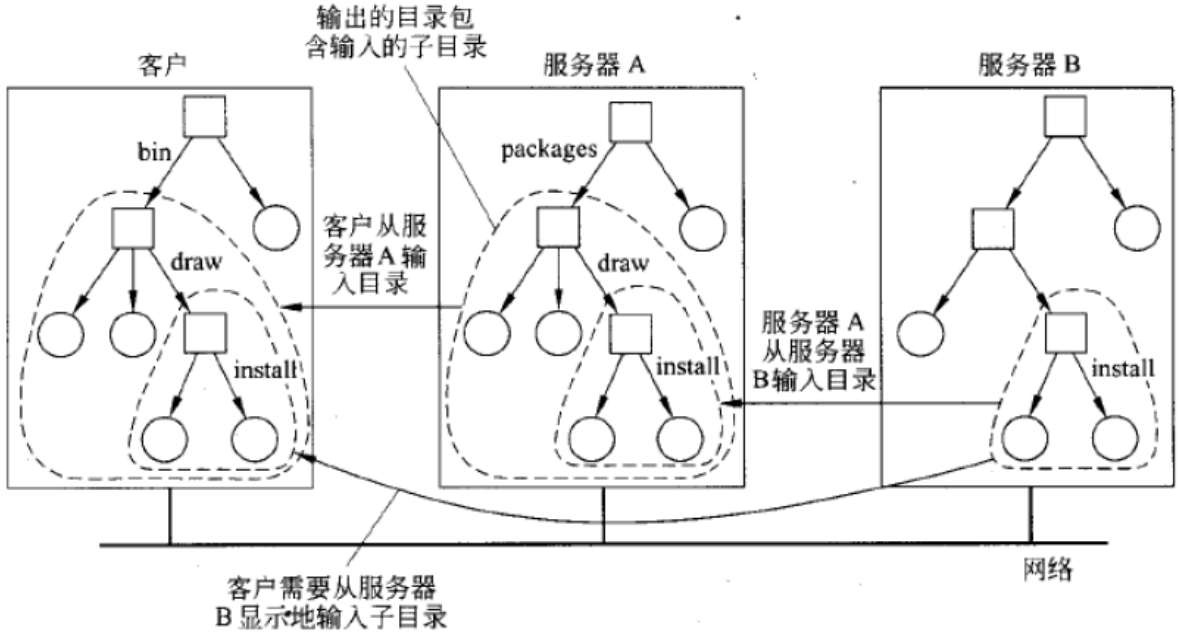


图 11.12　从 NFS 中的多个服务器挂载嵌套的目录

NFS自动挂载

- 何时挂载远程文件系统？

- Alice通过自己机器的/home/alice访问远端的文件系统，文件目录在登录时挂载；
- 她也可以通过/home/bob访问Bob的目录， 从而访问Bob的公共文件；
- Bob的主目录是否该在Alice登录时自动挂载？如果是，有什么缺陷？

NFS自动挂载

- 按需挂载

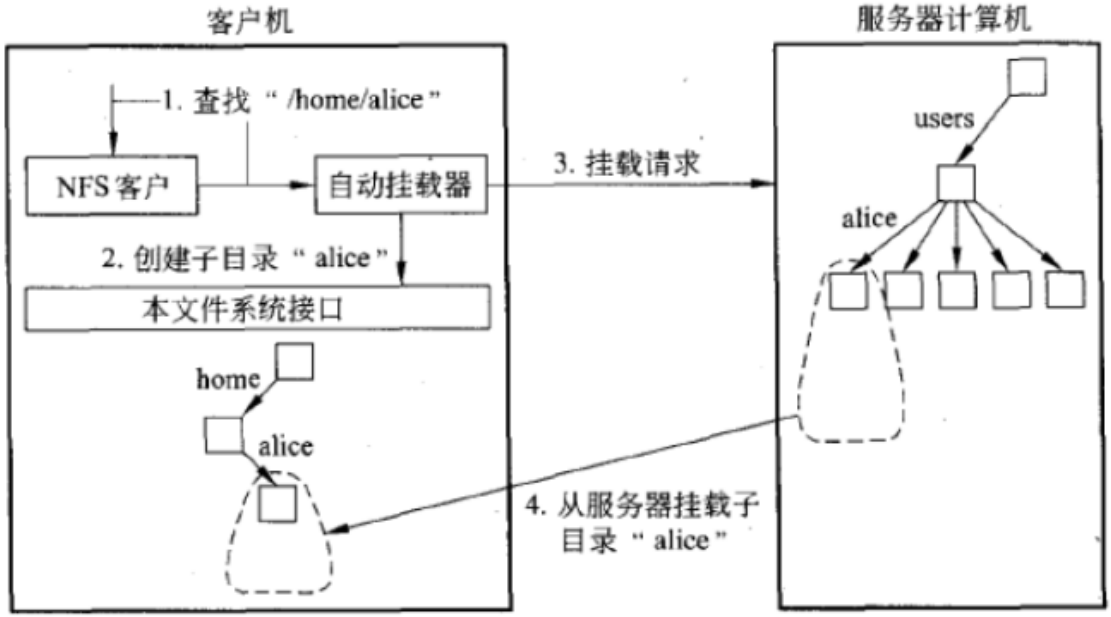


图 11.13 简单的 NFS 自动挂载器

构造全局名称空间

➢ **需求**

对于一个大型分布式系统，当需要提供对文件的共享访问时，至少要有一个全局名称空间。

□ 全局名称空间服务（ GNS）；

把现有文件系统集成进单个全局名称空间中，只使用用户级解决方案。

接合点	描述
GNS接合点	引用另一个GNS实例
逻辑文件系统名称	引用将在定位服务中查找的子树
逻辑文件名称	引用将在定位服务中查找的文件
物理文件系统名称	引用可直接远程访问的子树
物理文件名称	引用可直接远程访问的文件

图 11.15 GNS中的接合点

同步

- 文件共享语义

当需要处理分布式文件系统时，我们需要考虑并行读写的操作顺序和期望的语义（处理好一致性）；

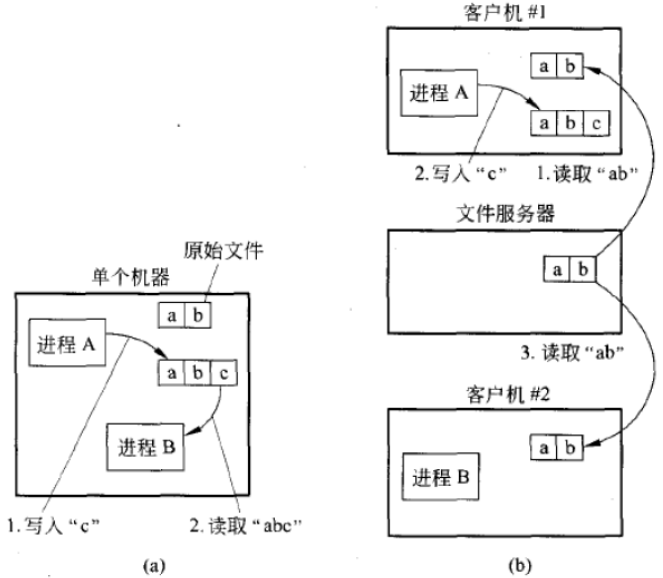


图 11.16

同步解决方案

➢ 同步问题解决方法

- 立即将缓存文件的所有改动传播回服务器，简单但是效率低；
- 对打开的文件所做的改动最初只对修改文件的进程可见，只有当文件关闭时，这些改动才对其他进程可见即会话语义；
- 所有文件都是不可改变的，文件上的操作只有 create和read操作；
- 使用原子事务处理共享文件；

同步解决方案

方法	注释
UNIX 语义	一个文件上的每个操作对所有进程都是即时可见的
会话语义	在文件关闭之前，所有改动对其他进程都是不可见的
不可改变的文件	不允许更新文件；简化了共享和复制
事务	所有改动都以原子方式发生

图 11.17　4 种处理分布式系统中的共享文件的方法

文件锁定

➢ 背景

- 客户-服务器体系结构中具有无状态的服务器，需要额外的方式同步对共享文件的访问。
- 实现方式：利用锁管理器。

➢ 问题

- 文件加锁机制；
- 锁的粒度；

文件锁定

- **NFS文件锁操作**

操作	描述
lock	为一定范围的字节创建锁
lockt	测试是否授予了相冲突的锁
locku	删除一定范围的字节上的锁
renew	继续租用指定的锁

图 11.18　NFS 版本 4 中关于文件锁定的操作

共享预约

- 一种锁定文件的隐含方法，共享预约独立于锁定。主要包含：**客户打开文件时指定的访问类型，以及服务器应该拒绝的其他客户的访问类型。**

当前文件拒绝状态

请求访问	NONE	READ	WRITE	BOTH
READ	成功	失败	成功	失败
WRITE	成功	成功	失败	失败
BOTH	成功	失败	失败	失败

(a)

请求的文件拒绝状态

当前访问状态	NONE	READ	WRITE	BOTH
READ	成功	失败	成功	失败
WRITE	成功	成功	失败	失败
BOTH	成功	失败	失败	失败

(b)

图 11.19 使用 NFS 共享预约实现 open 操作的结果

(a) 在当前拒绝状态下，客户请求共享访问时的情况；(b) 在当前文件访问状态下，客户请求拒绝状态时的情况

一致性和复制

缓存

➢ NFS缓存模型

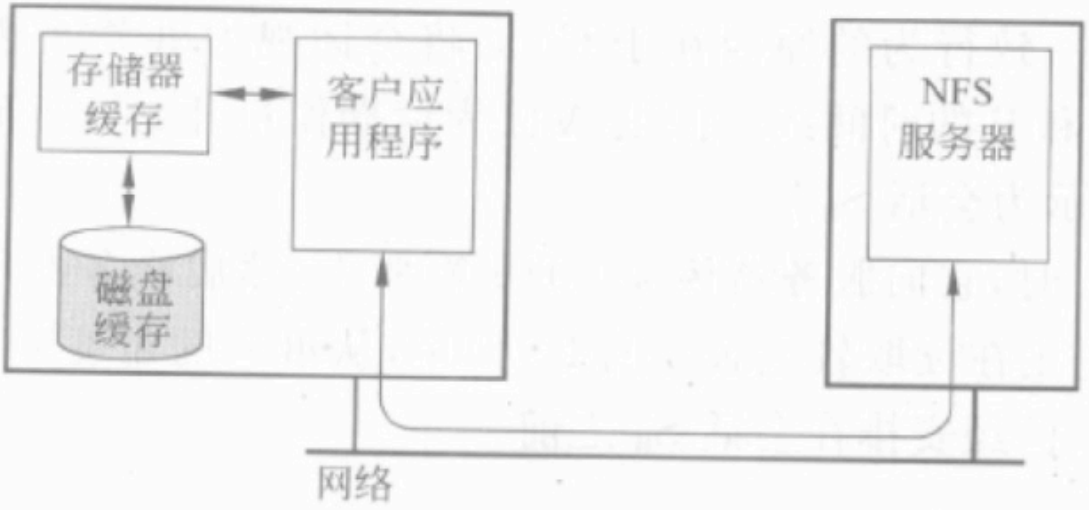


图 11.21 NFS 中的客户端缓存

NFS缓存

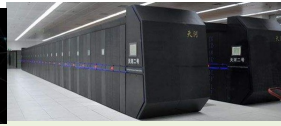
- **缓存方法一：**
  - 当客户打开文件时，将它从服务器获得的数据缓存起来，并把这些数据作为各种read操作的结果。
  - 另外，write操作也可以在缓存中执行。 客户关闭，如果文件被修改，必须回送到服务器。
- **缓存方法二：**
  - 服务器可以将某些权限委托给客户，即开放式委托。开放式委托发生在客户机被允许在本地处理来自同一机器的其他客户的Open和close操作。

对等文件系统中的复制

复制的主要作用是加快搜索和查找请求的速度，还可以平衡节点之间的负载。

- **非结构化对等系统：**
  - 基本原理：将查找数据归结为搜索网络中的数据；
  - 广播负载过高，如何寻找一种最优的复制策略。
- **结构化对等系统：**
  - 复制的作用是用于平衡节点之间的负载；





容错性

- 处理Byzantine故障

  基本思想是通过构造有限状态机的集合来部署主动复制，并且这个集合中具有无故障的进程以相同的顺序执行操作。

- Byzantine故障解决方案
  - 为了在异步环境中进行Byzantine容错，服务器组必须包含至少 3k+1 个进程；
  - 难点：确保无故障的进程以相同的顺序执行所有的操作。
  - 简单解决方案：指定一个协调器，它通过简单地给每个请求附加一个序号来序列化所有的操作。
  - 问题转嫁到协调器身上；

Byzantine容错

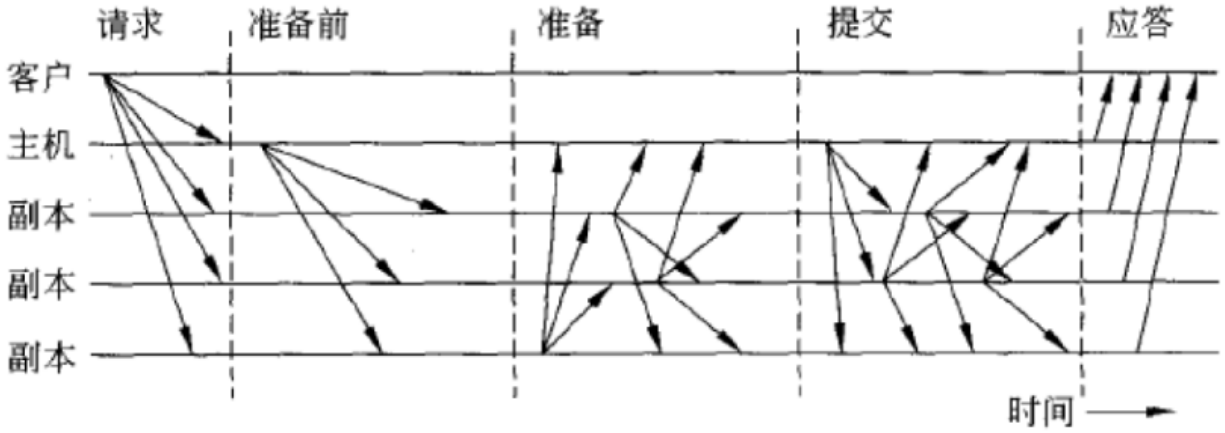


图 11.26 Byzantine 容错性中的不同阶段

- 进程会形成一系列视图， 每个视图都会对无故障的进程达成一致意见，并且在当前主机看起来失败时更新视图。
- 该协议的关键：可以正确地对请求排序；

对等系统（P2P）中的高度可用性

- 问题背景

P2P系统的节点的不可用性非常高，简单的复制文件已不能保证可用性；

- 冗余性方案
  - 复制：通过放置多个副本提高数据的冗余度从而提高可用性；
  - 擦除编码（erasure coding）：通过把一个文件分成 m 块，随后把它记录到 n>m 块中。任何m个编码块的集合都足以用于重构造原始文件；冗余性因子：

$$r_{ce} = \frac{n}{m}$$



擦除编码（erasure coding）

- 可用性分析

假定平均节点的可用性为 a, 必要的文件不可用性为 ε，需要保证至少有 m 块可用，即：

$$1-\varepsilon\sum_{i=m}^{n}\begin{bmatrix}n\\i\end{bmatrix}a^{i}(1-a)^{n-i}$$

与复制方法比较，发现文件的不可用性完全由它所有的 r_{rep}副本不可用的概率决定的。

$$1-\varepsilon=1-(1-a)^{r_{rep}}$$

复制与擦除编码之间的区别

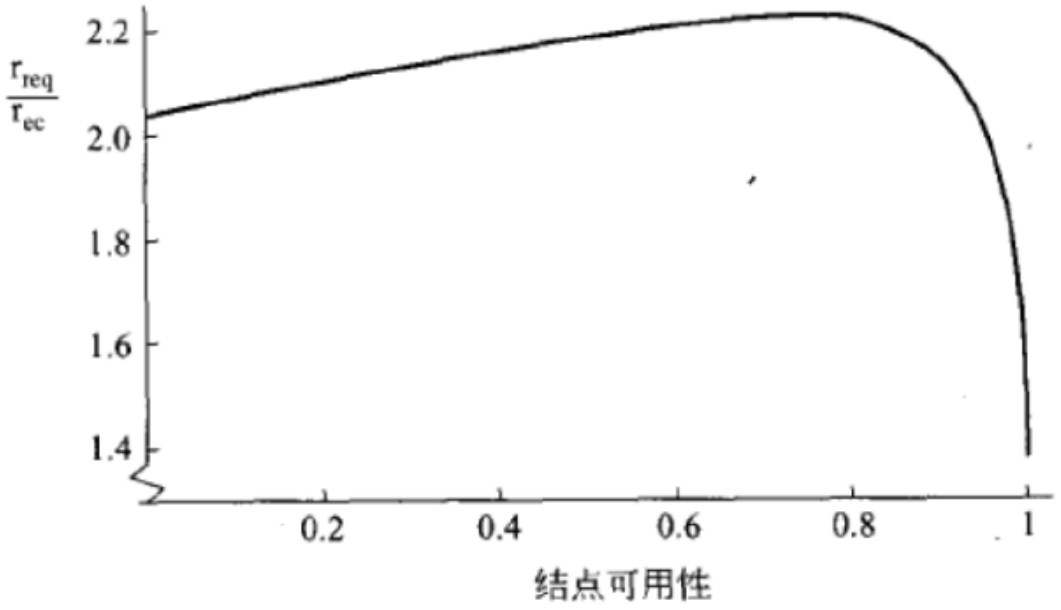


图 11.27　将比率 r_{rep}/r_{ec} 作为结点可用性 a 的函数

安全性

- NFS的安全性

NFS中的安全性主要集中于客户和服务器之间的通信；

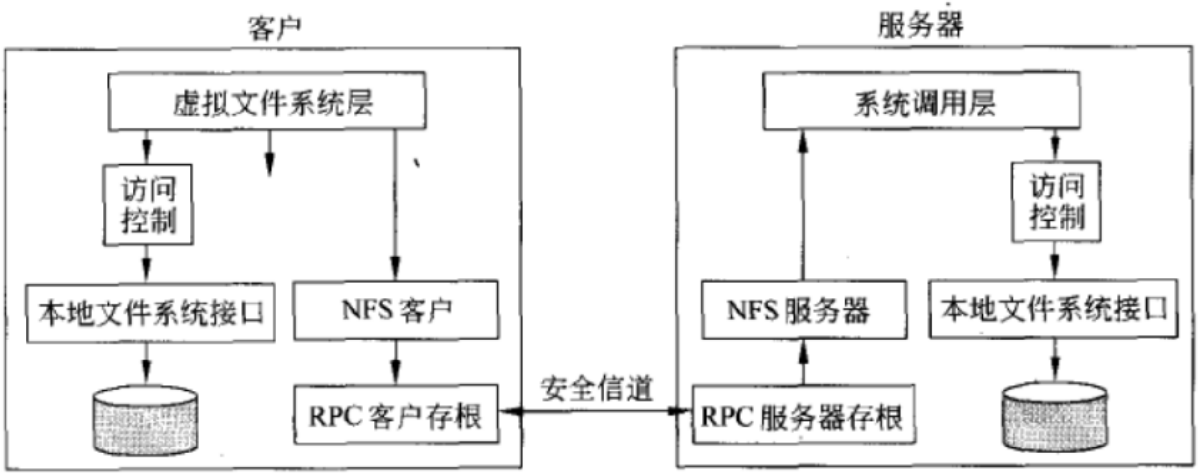


图 11.28 NFS 安全性体系结构



安全的RPC

- 安全的身份认证方法
  - 系统身份认证：
  - 使用密钥交换建立一个会话密钥；
  - 利用 Kerberos 身份认证协议；

安全的对等文件共享系统

- 基于DHT的系统中的安全查找

需要依靠安全查找操作，实质上可以归结为需要安全路由。需处理以下三个问题：

- 以安全的方式分配节点的标识符 （sybil attack）；
- 安全地维护路由表（eclipse attack）；
- 在节点之间安全地转发查找请求；

谢谢！