第一章

1、 相对于集中式系统，分布式系统的优点

Economics（经济性）	微处理器能提供比大型机更好的性价比
Speed（速度）	分布式系统能提供比大型机更强的计算能力
Inherent distribution（固有的分布性）	有一些应用包含物理上分布的机器
Reliability（可靠性）	当某台机器崩溃时，整个系统仍能正常工作
Incremental growth（可扩展性）	计算能力逐步增加

相对于独立的 PC，分布式系统的优点

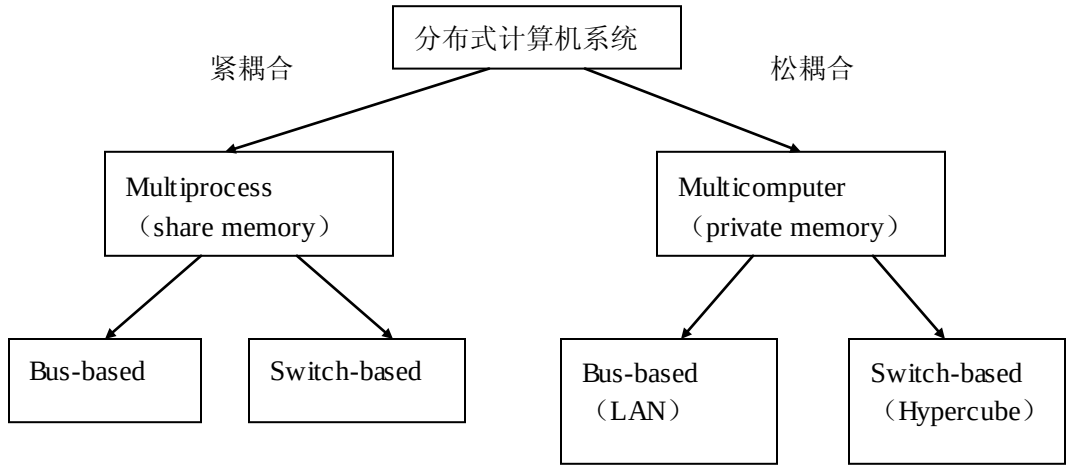
数据共享	允许用户共享一个数据库
外设共享	允许用户共享昂贵的外设，如彩色打印机
通信	使得个人与个人之间的通信更为方便，如 Email
灵活性	将工作负载更有效的分派到合适的机器上

分布式系统的缺点

软件	目前很少有分布式系统的软件
通信网络	网络可能饱和或其他问题
安全	数据共享造成机密数据容易被窃取

2、 分布式系统的定义：A collection of independent computers that appears to its user as a single, coherent system.

3、 分布式计算机系统的分类：



4、 分布式系统与分布式 OS，网络 OS 的比较（课本 P41 页）

第二章

1、 RPC 的概念：当机器 A 上的一个进程调用机器 B 上的一个过程，则调用进程挂起，被调用进程在 B 上执行。信息可通过参数从调用者传递到被调用者，或通过结果从被调用者返回给调用者。消息传递对程序员完全不可见。这种方法称 RPC。

2、 缺点：数据表示的不一致；参数传递问题；性能（慢）；安全

3、 Client 绑定到 Server：Server 先向本地 OS 请求一个端点（port），然后在 DCE daemon 中注册该端点，记录在端点表中。Server 再向一个目录 Server 注册自己的网络地址和一个名字。Client 以 Server 提供的服务名字为 key 在目录 Server 中查找对应的网络地址，然后向 Server 上的 DCE daemon 请求服务端点。这些信息都具备后，执行 RPC 操作。

4、 Remote Object Invocation：将 RPC 扩充到远程对象。

分布式对象：对象包括状态、方法和接口。将接口和对象分离使得我们能够将接口和对象本身放置在不同的机器上。

ROI 和 RPC 的比较

	Client	Server
ROI	Proxy（Interface）	Skeleton（Object）
RPC	Client Stub	Server Stub

传统的 RPC 和支持分布式对象的 RPC 的一个区别在于后者提供系统范围（systemwide）的对象引用。

6、• 将客户端和对象进行绑定：隐式绑定；显式绑定（客户端首先调用一个特殊函数 bind 和对象进行绑定，返回一个指向本地可获得的代理的指针）。

显式绑定的步骤：声明一个系统范围的对象引用；声明一个指向本地对象的指针；初始化分布式对象的引用；显式绑定，得到本地 proxy 的指针；调用本地 proxy 上的某个方法。

• 静态 RMI 和动态 RMI（Remote Method Invocation）

RMI 和 RPC 的区别在于 RMI 提供系统范围（systemwide）的对象引用。

静态 RMI：使用预定义的接口定义，要求对象的接口在开发客户端应用程序时就知道，一旦接口改变，客户端应用程序必须在使用新接口前重新编译。

传值和传引用：当以一个对象作为参数调用一个方法时，远程对象传递该对象的引用；本地对象传递该对象。

7、 面向流的通信

分布式系统应该为与时间有关的（即连续媒体）信息（如声音流和视频流）提供怎样的机制？

• 数据流：数据单元的序列。离散数据流（如 UNIX 的管道，TCP/IP 的连接）和连续数据流（如播放音频文件时文件和设备之间的音频流）。

异步传输模式：流中的数据项一个接一个传输，对传输时间没有限制，一般用于离散数据流。

同步传输模式：对数据流中每个单元有一个端到端的最大延迟。

简单流（只有一个数据序列）和复杂流（有多个相关的简单流）。

• QoS（Quality of Service）：与时间相关的要求。可以通过提供对流精确的描述来指定 QoS。

• 流的同步：保持流之间的时间关系。一种是离散数据流与连续数据流之间的同步，一种是连续流之间的同步。在发送时进行同步比在接收端同步简单。

同步机制有：一个进程执行对多个简单流的读写操作，保证这些操作满足时间和同步限制；或为应用程序提供一个接口，允许它更简单的控制流和设备。

永久连接：提交用于传送的消息被通信系统存储，直到它被发送；

暂时连接：不提供存储机制，接受方必须准备接收将要发来的消息。

面向消息的 Middleware 提供永久异步通信。

第三章、进程

1、 进程的缺点：每创建一个进程，OS 必须创建一个完全独立的地址空间；CPU 在进程之间的切换开销大。

2、 多线程进程的优点：若进程中某个线程需等待某个条件，则可让其他线程执行，而单线程进程的话则整个进程都要挂起；在多处理器系统中执行程序时，多线程进程可以挖掘出并行性。

3、 线程通常以线程包的形式给出，实现线程包的两种途径是：创建一个完全在用户模式下执行的线程库；让 kernel 知道线程并安排它们的时间。

第一种的优点是：创建和消除一个线程代价小；切换线程上下文只需少量指令即可完成。缺点是一个被阻塞的系统调用将立即阻塞该线程所属的进程以及进程中的其他所有线程。第二种相反。更好的解决方法是两种形式的混合，LWP（轻量线程）。一个 LWP 运行在单个（重量）线程的上下文中，一个进程中可有多个 LWP。系统不仅提供 LWP，还可提供用户级线程。

对象适配器（Adapter）：又称 object wrapper，将对象根据 policy 来分组的机制。一个 server 可有多个 object adapter，一个 object adapter 可有多个对象。当调用请求递交给 server，它将请求分发给适当的 object adapter，object adapter 从请求中提取一个对象引用，将请求分配到该引用的对象的服务器端的 stub，stub 处理请求并调用适当的方法。

6、 代码迁移：分布式系统中的通信不仅可传送数据，还可传送程序（甚至有时正被执行）。

• 基本思想：总的系统性能在进程从重负荷机器移到轻负荷机器时可得到提高；有助于开发并行性而没有并行编程的复杂性；可提高灵活性。

• 创建分布式应用程序的传统方法是将应用划分为不同部分，事先决定每个部分应该在哪里执行，而若代码可迁移，则可动态配置分布式系统。动态配置分布式系统的优点是客户不需要预先安装所有软件就可以与服务器会话，缺点是安全性问题。

• 代码迁移模型：将进程划分成三部分（code segment, resource segment, execution segment）。

• 迁移和本地资源：资源部分不能简单的与其他部分一起转移而不发生改变。进程与资源的绑定有三种方式（binding by type, unattached resource, fastened resource）

7、 软件代理（Agent）：能够对环境作出反应，初始化其中的变化的独立进程，它可以与其他用户或代理合作。

• Collaborative agent：组成一个多代理系统一部分的代理，在该系统中的代理试图通过合作来达到某个共同的目标。

• Mobile agent：能够在不同机器中迁移的 agent。

• Interface agent：帮助终端用户使用一个或多个应用程序，它有学习功能。

• Information agent：从多个不同的源中管理信息。

代理平台的基本模型包括：一个代理管理组件，一个独立的本地目录服务和代理通信通道（ACC）。

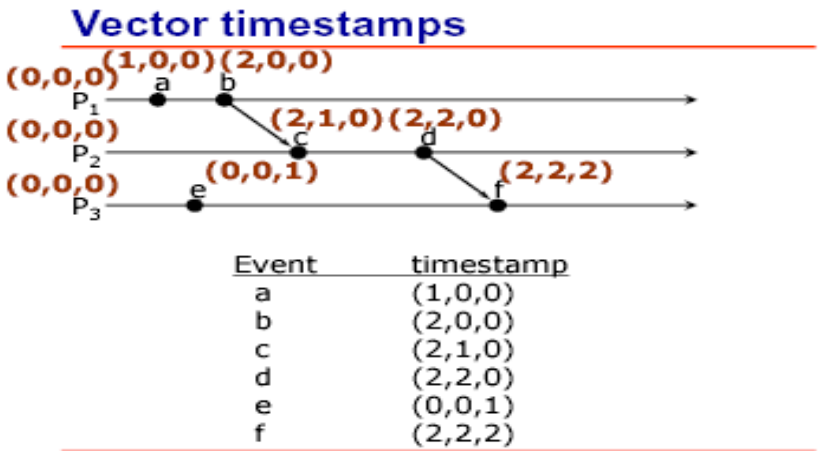
代理通信语言（ACL）：代理间的通信通过一个应用层的通信协议进行，该协议称为 ACL。ACL 的核心在于发送和接收代理都对消息的目的有相同的解释。

第五章

1、逻辑时钟：进程是否在精确时间上达成一致并不重要，关键是在事件发生的先后顺序上要达成一致。

2、Lamport 算法用于同步逻辑时钟。定义了事件上的二元偏序关系 happen-before。若消息抵达时间先于发送时间，则接收方将时钟在发送时间上加 1。

3、 向量时间戳：若事件 A happen-before B，则有 LC(A)<LC(B)，反之不成立。若用向量时间戳来定义，有若 VT(A)<VT(B)，则 A causally precede B。



Event	timestamp
a	(1,0,0)
b	(2,0,0)
c	(2,1,0)
d	(2,2,0)
e	(0,0,1)
f	(2,2,2)

4、分布式系统的全局状态＝各个进程的局部状态＋正被传送的消息。记录全局状态的方法

可以是分布式快照，快照反映了一个一致的全局状态。得到分布式快照的算法

一个进程执行一些特殊操作，该进程称为 coordinnator。选择 coordinator 有两种选举算法，都是选择具有最高进程号的进程。Bully 算法；Ring 算法（不使用 token，要与分布式互斥中的带有 token 的算法区别开）

5、 互斥（基础：Lamport 算法）。三种互斥算法：

	实现	优点	缺点
集中式算法	由 coordinator 管理临界区	无饥饿现象；易实现（三条指令）	Coordinator 单点失效和性能瓶颈
分布式算法	所有进程参与决策，要求所有事件有全局统一的顺序	一个进程 crash 将 block 所有等待进入临界区的进程；需保存所有组员信息；性能较差（所有进程参与决策）	
Token ring 算法	Token 沿环传递，得到 token 的进程才能进入临界区	正确性	Token 可能丢失且难于发现；某个进程可能 crash（但较易检测）

第六章

1、 复制的优点：reliability；performance。缺点：复制透明性；一致性问题。

2、 多个 clients 并发访问远程对象的两种控制方式：一是对象自己处理（使用一个互斥机制）；二是对象驻留的 server 用一个 adapter 来负责。

解决复制对象一致性问题的两种方法：一是对象知道自己能被复制，对象之间可以通信，优点是支持特定对象的赋值策略；二是由分布式系统自己管理（middleware 有一个复制协议），优点是应用程序开发者任务简单。复制和 cache 是两种扩展技术。

3、 Data-centric 一致性模型：Strict；linearizability；sequential；causal；FIFO；weak；release；entry。client-centric 一致性模型：eventual consistency 保证最后各个副本一致

4、 分布式协议：三种数据拷贝的类型：永久复制（多个拷贝分布再 LAN 的多个 server 上；镜像分布再 Internet 中）;server 发起的复制(要精确判断何时何地创建或撤销副本);client 发起的复制（主要是 cache）。

5、 更新传播的三种传播方式：仅传播一个更新通知；传播更新后数据；传播更新操作种类。推/拉协议的比较。

6、 一致性协议

- Primary-based 协议（每个数据项 x 有一个相关的 Primary 负责对 x 的读写操作）
  - Remote-write 协议（所有读写操作都在一个远程 server 上进行）
  - Primary-backup Remote-write 协议（可以本地读，但写必须由 primary 管理）
  - Local-write 协议（将数据项迁移到本地 server 后本地操作）
  - Remote-write Local-write 协议（本地写操作后更新 backup server）
- Replicated-write 协议（写操作可对多个副本同时进行）
  - Active replication（多个副本对象对其他对象的调用以及多个副本对象返回结果时要选择一个 coordinator）
  - Quorum-based 协议（客户再对副本进行读写操作前必须得到 server 的批准，两个不等式）
- cache-coherence 协议（静态检测，动态检测）