1.分布式系统的定义

分布式系统是指由多个相互连接的处理资源组成的计算机系统，计算机之间通过消息传递进行通信和动作协调，从用户角度来说，它如同一个集中的单机系统。

对定义作如下说明：

（1）分布式系统是由多个处理器或者计算机系统组成，这些计算资源可以是物理上相邻的，也可以是地理上分散的。

（2）.分布式系统中的计算机是一个整体，对用户是透明的，用户在使用资源时无须知道这些资源在哪里。

（3）程序可以分散到各个计算资源上运行，在程序需要协作是，它们通过交换消息来协调彼此的动作。

（4）不存在集中的控制环节，各个计算系统之间是平等的。

2.分布式系统的特征

（1）透明性

访问透明性：用相同的操作访问本地和远程的资源，用户无须区分本地资源和远程资源；

位置透明性：不需要知道资源的位置就能够访问它们。当资源在系统中移动时，在资源名字保持变的情况下，原有的程序都可正常运行；

并发透明：多个进程能并发的对共享资源进行操作而互不干扰和破坏；

复制透明性：允许资源的多个副本同时在系统中存在以增加可靠性和性能，而用户和应用程序员无须了解副本的存在；

故障透明性：屏蔽错误，无论是硬件故障还是软件组件故障，整个系统都不会失效，允许用户和应用程序完成它们的任务；

伸缩透明性：在不影响用户或程序的操作的前提下，允许系统和应用扩展。

移动透明性：在不影响用户或程序的操作的前提下，允许资源和客户在系统内移动；

性能透明性：当负载变化时，允许系统重构以提高性能。

（2）资源共享

（3）并发性

分布式系统中的每个节点既独立工作，又与所有其他节点并行工作。

（4）异构性

系统中包含的节点可以由不同的计算与通信硬件组成。

（5）容错性

（6）安全性

分布式系统维护的众多信息源对用户具有很高的内在价值，因此它们的安全性相当重要。

（7）开放性

大部分分布式系统是可扩展的，因为在系统运行期间，可以添加或改变节点、组件和应用程序。开放性要求每个组件遵守一组最起码的策略、惯例和协议，以确保更新或增加的组件之间具有互操作性。

（8）分散控制

单独的计算机无须对整个系统的配置、管理或策略控制担负责任。分布式系统则是通过自主主体协议连接的域，而这些自主主体为提供聚合功能要达成足够的共同策略。

3.死锁问题

（1）死锁的定义（详细 ppt 1.4.1）

死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

死锁发生的条件：

严格地说，当且仅当以下四个条件同时成立时，才会发生死锁：

①互斥：同一资源在同一时刻不能被超过一个进程共享。

②占用并等待：必然有一个进程占用了至少一个资源，同时在等待被其他进程占用的资源。

③不可剥夺：资源不能被剥夺，系统将一个资源的访问权分配给某一进程后，系统不能强迫该进程放弃对该资源的控制权。

④循环等待：在等待图中的有一个循环。

（2）死锁的预防

死锁预防算法是通过限制进程的资源请求来达到预防死锁的目的，都是通过打破四个死锁条件中的一个条件来实现的。

①进程在开始执行之前同时获得所有所需资源。这种方法打破了占有并等待的条件。

②所有的资源都要被赋予一个唯一的数字编号。一个进程可以请求一个有唯一编号 i 的资源，条件是该进程没有占用编号小于或等于 i 的资源。这样，就打破了循环等待的条件。

③每个进程被赋予一个唯一的优先级标识。优先级标识决定了进程 Pi 是否应该等待进程 Pj，从而打破了不可剥夺的条件。

基于时间戳的预防死锁方法: 包括两种死锁预防方案。这两种方案相互补充，这种方法常用于分布式数据库系统中。（图解详见 PPT 1.4.2）

①等待—死亡方案(wait-die scheme)。该方案是基于非剥夺方法。当进程 Pi 请求的资源正被进程 Pj 占

有时，只有当 Pi 的时间戳比进程 Pj 的时间戳小时，即 Pi 比 Pj 老时，Pi 才能等待。否则 Pi 被卷回(roll-back)，即死亡。

②伤害—等待方案(wound-wait scheme)。它是一种基于剥夺的方法。当进程 Pi 请求的资源正被进程 Pj 占有时，只有当进程 Pi 的时间戳比进程 Pj 的时间戳大时，即 Pi 比 Pj 年轻时，Pi 才能等待。否则 Pj 被卷回(roll-back)，即死亡。

（3）死锁的检测（三种死锁检测方法详细 见 PPT 1.4.3）

①集中式死锁检测

②分布式死锁检测

③层次式死锁检测

（4）互斥算法（PPT 中提到 9 个互斥算法，下面列出选择的两个）

首先关于互斥问题的描述，以及集中式互斥算法和分布式互斥算法的区别，详见 PPT 1.4.6 节。

①Lamport 时间戳互斥算法

Lamport 时间戳互斥算法由以下 5 条规则组成：

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

ⅱ一个进程 Pj 如果收到具有时间戳 Tm:Pi 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 Pi 发送一个带有时间戳的承认报文。如果 Pj 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 Pj 从临界区中退出之后或 Pj 发送完自己的申请报文之后再发送，否则立即发送；

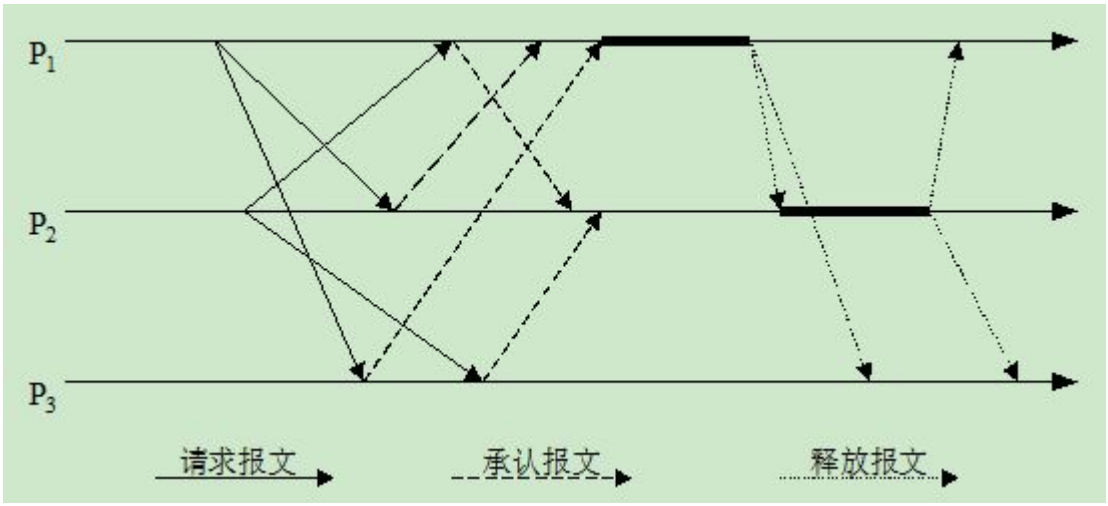
ⅲ一个进程 Pi 如果想释放资源，它先从自己的申请队列中删除对应的 Tm:Pi 申请报文，并向所有其他进程发送具有时间戳的 Pi 释放资源的报文；

ⅳ一个进程 Pj 如果收到 Pi 释放资源的报文，它从自己的申请队列中删除 Tm:Pi 申请报文；

ⅴ当满足下述两个条件时，申请资源的进程 Pi 获得资源：

a. Pi 的申请队列中有 Tm:Pi 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. Pi 收到所有其它进程的承认报文，其上面的时间戳值大于 Tm。



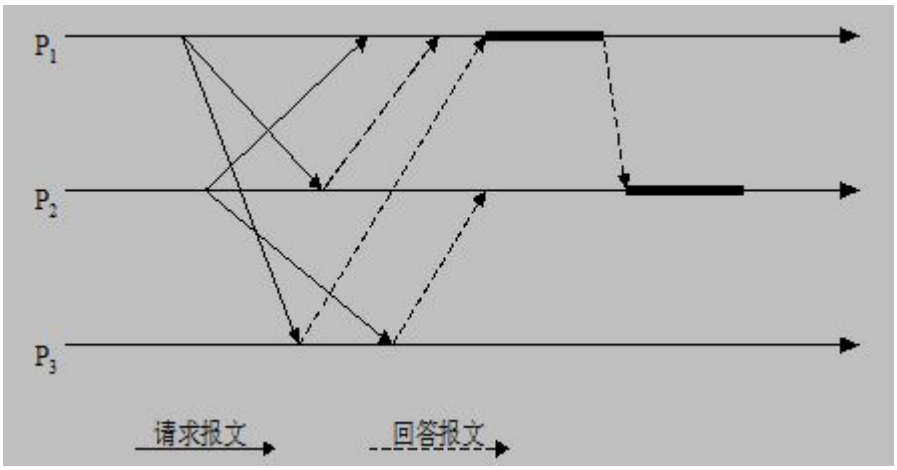
②Ricart-Agrawala 互斥算法

ⅰ一个进程申请资源时向所有其他进程发出申请报文；

ⅱ其它进程收到申请报文后若不在临界区并且自己未申请进入临界区，或者自己虽然发出了申请报文，但自己的报文排在收到的申请报文之后，则回答表示同意；

ⅲ申请资源的进程仅在收到所有进程的回答报文后才进入临界区使用资源；

ⅳ一个进程使用完资源后，它向所有未给回答的其它申请发送回答报文。



4. 容错技术

检查点算法：

（1）一致性检查点

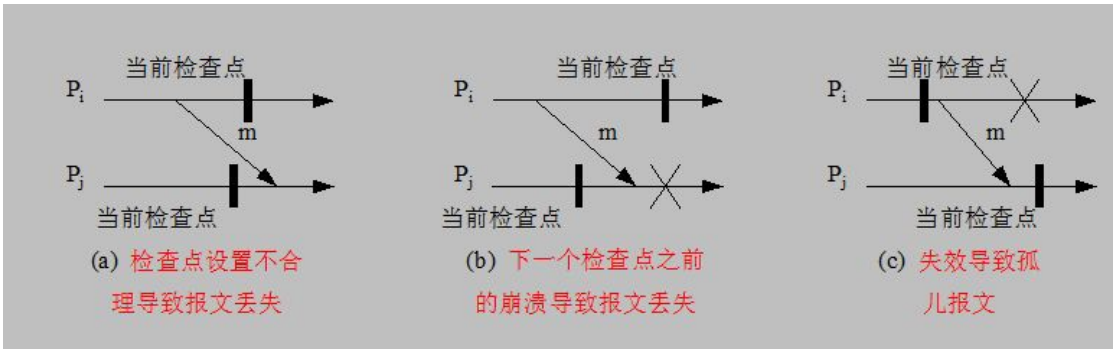
①全局状态：一种全局状态的定义是一系列局部状态的集合，这里的局部状态就是一个进程的检查点，每个局部进程有一个局部状态。

②局部检查点可能组成如下两种不一致的全局状态：

(a).丢失报文。进程 Pi 的检查点状态显示它给进程 Pj 发送了报文 m，但是进程 Pj 并没有关于该报文的纪录。

(b).孤儿报文。进程 Pj 的检查点状态显示它收到了一个来自进程 Pi 的报文 m，但是进程 Pi 的状态显示它没有向进程 Pj 发送过报文 m。

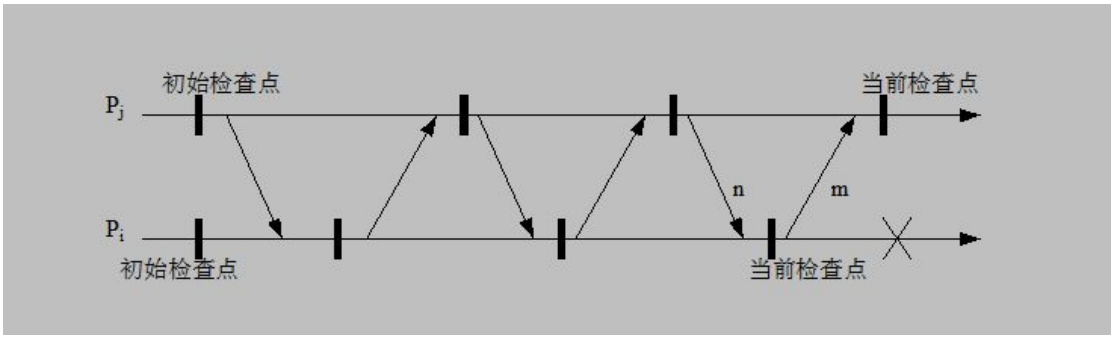
③不一致全局状态；



重点关注：丢失报文和孤儿报文的形成及处理

④多米诺效应。

为了解决孤儿报文的问题，进程 Pj 回卷到上一个检查点时，要清除对孤儿报文的纪录。然而，这样一来有可能出现这样一种情况：在最近的检查点和上一个检查点之间，Pj 向 Pi 发送了一个报文 n，假定 Pi 在当前检查点之前收到了报文 n，现在这个报文 n 成了孤儿报文。这样，Pi 需要进一步回卷。这种由于一个进程的回卷导致另外一个或多个进程的回卷的效应叫做多米诺效应。

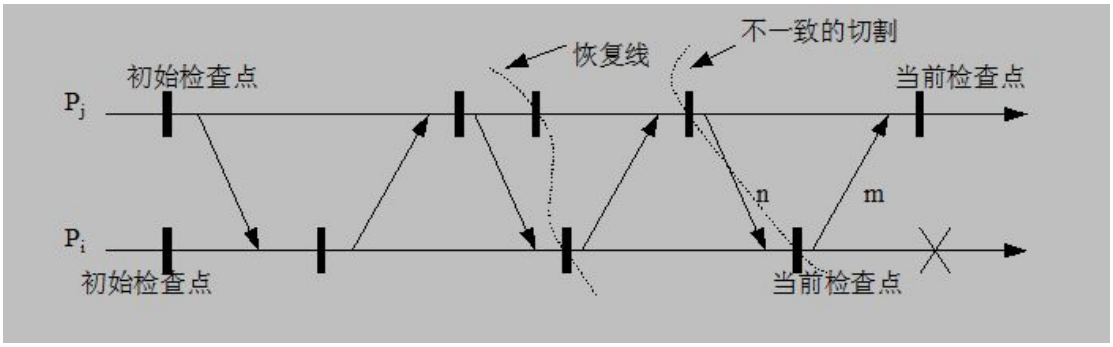


⑤一个强一致(strongly consistent)的检查点集合是由一系列的没有孤儿报文和没有丢失报文局部检查点组成。

⑥一个一致的检查点集合是由一系列没有孤儿报文的局部检查点组成。

显然一个强一致的检查点集合包括一系列局部检查点，在这些检查点之间，进程之间没有报文传送。如果每个进程都在发送一个报文之后生成一个检查点，那么最近的检查点集合将永远是一致的。

⑦恢复线和切割线：当一个进程或系统失效的时候要求利用这些局部状态重新构造一个全局一致的状态。一个分布式快照对应一个全局一致的状态，这个分布式快照可以作为一个恢复线(recovery line)用于恢复。一个恢复线对应于最近的一个一致性切割线。



5.远程调用 RPC 和 RMI（Java RMI 实例见 PPT 3.3.4）

远程过程调用(RPC)

面向对象的分布式系统通常使用远程客户端来唤醒一个对象，这种机制主要采用远程过程调用(RPC)。然而，使用 RPC 之前必须要预先完成以下一些关键步骤：

①绑定客户端与对象

②静态与动态远程过程调用

③参数传递

（1）绑定客户端与对象

传统的 RPC 系统与**支持分布式对象的系统**的区别在于后者通常提供全局对象引用，并且与传统 RPC 相比，分布透明性更高。

通过隐式结合，每个对象对应唯一的引用，客户端可以通过简单的机制实现对方法的直接调用。在引用指向实际对象时客户可以显性地绑定对象。相反，通过显式绑定，客户在调用方法之前首先必须调用一个特定的函数来绑定对象,而且通常要返回指针到代理才能做到局部有效。

（2）静态和动态远程过程调用

当客户端绑定对象后，它可以通过代理调用对象的函数。这种**远程方法调用(RMI)**在列集（类型可串行化）和参数传递方面非常类似于 RPC。**RMI 和 RPC 本质的区别**在于 RMI 通常支持全局对象引用，并且 RMI 不需要专用的客户端和服务器端存根。取而代之的是更简单的面向服务器的存根。

静态调用：调用预先定义好的接口的调用方法通常被称为静态调用。

动态调用：方法调用也可以用更动态的方式。支持实时调用的方法通常被称为动态调用。

静态调用与动态调用的区别：其与静态调用的本质区别在于实时调用的哪个方法将调用远程对象。

动态调用的应用：针对对象的浏览器，可以支持任何可能的接口，这要求接口能被实时检查并且方法的调用能够被动态地构建；动态调用的另一个应用是批处理服务，即调用请求能够在它应该被执行时实现。

（3）参数传递

由于绝大多数 RMI 系统支持全局对象引用，在方法调用中传递参数通常也没有 RPC 方法中那么严格。但是需要注意**仅有分布式对象的情况，虽然始终可以像调用方法中的参数一样来使用对象引用，但是其效率非常低下**。

在机器 A 上运行客户端程序，机器 C 上运行服务器程序。客户端有一个指向本地对象的 O1 的引用，在呼叫机器 C 上的服务器程序时被用作参量。此外，机器 A 上有一个指向位于机器 B 的远程对象 O2 的引用，同样被用作参量。当呼叫服务器时，O1 的拷贝被传递到机器 C 上的服务器，同时仅将一个引用的拷贝传递给 O2。

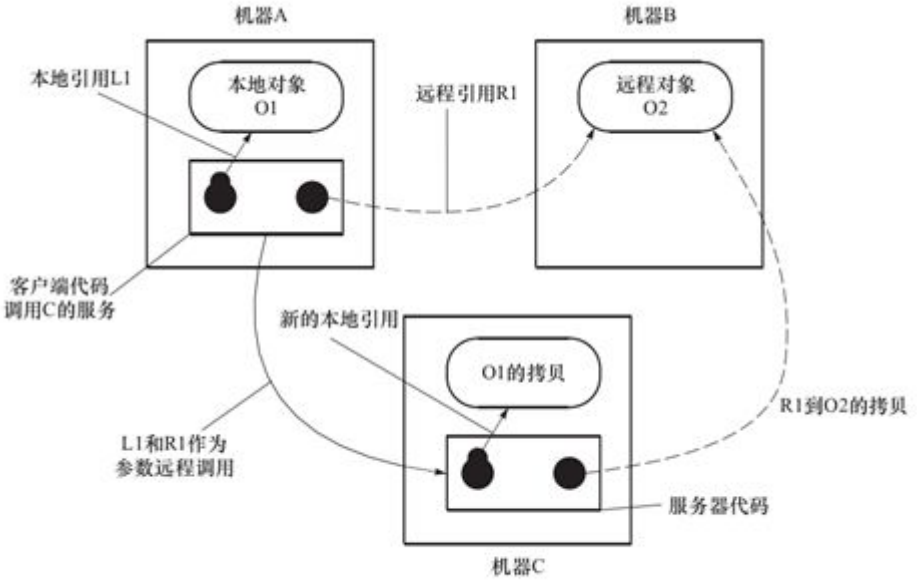


图 3-3　对象传递示例图(引用与参量)

远程对象是一种状态总是储存在一台单独的机器上的分布式对象，但是其接口能被远程进程实现。

远程对象和本地对象间的区别是很少但是微妙而重要的。由于克隆本地和克隆远程对象是不同的。因此，**克隆远程对象操作仅能被服务器进行**，即**在服务器的地址空间完全复制实际对象，实际对象的代理并不被复制**。

6.基于 web 的分布式系统（C/S 结构、B/S 结构，及其区别，三层 C/S 结构）

（1）C/S 结构：即客户机和服务器软件系统体系结构。通过它可以充分利用两端硬件环境的优势，将任务合理分配到 Client 端和 Server 端来实现，降低了系统的通讯开销。

C/S 模式的优点：

①客户端实现与服务器的直接相连，没有中间环节，因此响应速度快；

②操作界面漂亮、形式多样，可以充分满足客户自身的个性化要求；

③C/S 结构的管理信息系统具有较强的事务处理能力，能实现复杂的业务流程。

C/S 模式的缺点：

①需要专门安装客户端程序，分布功能弱，针对点多面广且不具备网络条件的用户群体，不能够实现快速部署安装和配置;

②兼容性差，对于不同的开发工具，具有较大的局限性;若采用不同工具，需要重新改写程序;

③开发成本较高，需要具有一定专业水准的技术人员才能完成。

（2）B/S 结构：即浏览器和服务器结构。它是随着互联网技术的兴起，对 C/S 结构的一种变化或者改进的结构。

在这种结构下，用户工作界面是通过 WWW 浏览器来实现，极少部分事务逻辑在前端(Browser)实现，但是主要事务逻辑在服务器端(Server)实现。

B/S 模式的优点：

①维护和升级方式简单。

②在系统的性能方面，B/S 结构的优势是其异地浏览和信息采集的灵活性。任何时间、任何地点、任何系统，只要可以使用浏览器上网，就可以使用 B/S 系统的终端。

③适用互联网、维护工作量等方面，B/S 比 C/S 要强得多。

B/S 模式的缺点：

①采用 B/S 结构，客户端只能完成浏览、查询、数据输人等简单功能，绝大部分工作要由服务器承担，这使得服务器的负担很重。

②在运行速度、数据安全、人机交互等方面，B/S 远不如 C/S。

区别：

首先必须强调的是 C/S 和 B/S 并没有本质的区别：B/S 是基于特定通信协议(HTTP)的 C/S 架构，也就是说 B/S 包含在 C/S 中，是特殊的 C/S 架构。之所以在 C/S 架构上提出 B/S 架构，是为了满足瘦客户端、一体化客户端的需要，最终目的节约客户端更新、维护等的成本，及广域资源的共享。

①B/S 属于 C/S，浏览器只是特殊的客户端;

②C/S 可以使用任何通信协议，而 B/S 这个特殊的 C/S 架构规定必须实现 HTTP 协议;

③浏览器是一个通用客户端，本质上开发浏览器，还是实现一个 C/S 系统。

（3）三层 C/S 结构

三层 Client/Server 结构 （以下简称三层模式 ）在两层模式的基础上，增加了新的一级。这种模式在逻辑上将应用功能分为三层：客户显示层、业务逻辑层、数据层。客户显示层是为客户提供应用服务的图形界面，有助于用户理解和高效的定位应用服务。业务逻辑层位于显示层和数据层之间，专门为实现企业的业务逻辑提供了一个明确的层次，在这个层次封装了与系统关联的应用模型，并把用户表示层和数据

库代码分开　。这个层次提供客户应用程序和数据服务之间的联系，主要功能是执行应用策略和封装应用模式，并将封装的模式呈现给客户应用程序。数据层是三层模式中最底层，他用来定义、维护、访问和更新数据并管理和满足应用服务对数据的请求。

三层模式的主要优点为：

①良好的灵活性和可扩展性。对于环境和应用条件经常变动的情况，只要对应用层实施相应的改变，就能够达到目的。

②可共享性。单个应用服务器可以为处于不同平台的客户应用程序提供服务，在很大程度上节省了开发时间和资金投入；

③较好的安全性。在这种结构中，客户应用程序不能直接访问数据，应用服务器不仅可控制哪些数据被改变和被访问，而且还可控制数据的改变和访问方式　。

④增强了企业对象的重复可用性。“企业对象”是指封装了企业逻辑程序代码，能够执行特定功能的对象。随着组件技术的发展，这种可重用的组件模式越来越为软件开发所接受。

⑤三层模式成为真正意义上的“瘦客户端”，从而具备了很高的稳定性、延展性和执行校率。

⑥三层模式可以将服务集中在一起管理，统一服务于客户端，从而具备了良好的容错能力和负载平衡能力。

7.web 服务体系及使用（详细介绍见 PPT 4.5.2）

Web 服务资源的体系结构由四个部分组成:

①服务提供者与服务代理组织形成区域自治代理系统(area proxy autonomy system, APAS)，它仅仅管理本区域内的 Web 服务注册与维护。

②将同一类 Web 服务资源组织成分布式的生成资源树，称为 Web 服务资源组织树(Web services resource organizing tree, WSROT) ,它是一个逻辑上的同类资源形成的资源树，记录了各 APAS 中的资源情况与负载信息，主要目的是当用户申请资源时，它将当前最“合适”的服务提供者返回给用户。

③Web 服务资源注册中心(Web services resource register center, WSRRC) , WSRRC 中记录所有各类 Web 服务资源的资源号与相对应的 WSRRC，它是分布式、一致性的全球性系统。

④用户系统，它并不需要知道网络中有 WSROT,也不需知道 APAS，所以用户并不直接与资源的组织和管理系统交互，它先向 WSRRC 查询资源地址，由 WSRRC 向所对应的 WSROT 转发请求，由 WSROT 返回给用户服务提供者的 IP 地址，然后用户采用 SOAP 与服务提供者交互得到服务。

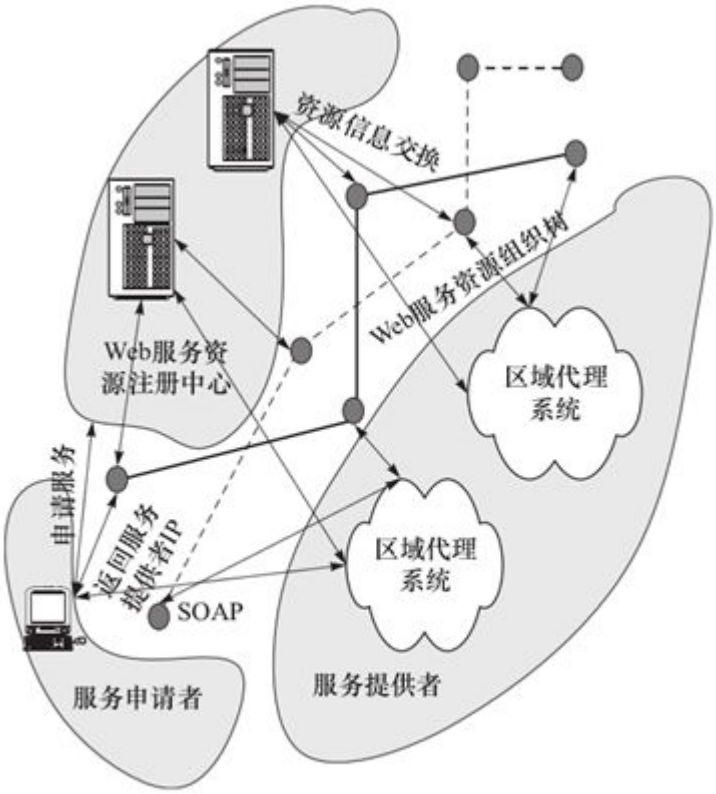


图 4-15　网格 Web 服务结构模型

8.网格体系结构

主流的网格体系结构有三个：五层沙漏结构、开放网格服务体系结构（OGSA）和 Web 服务资源框架（WSRF）。

（1）五层沙漏结构：

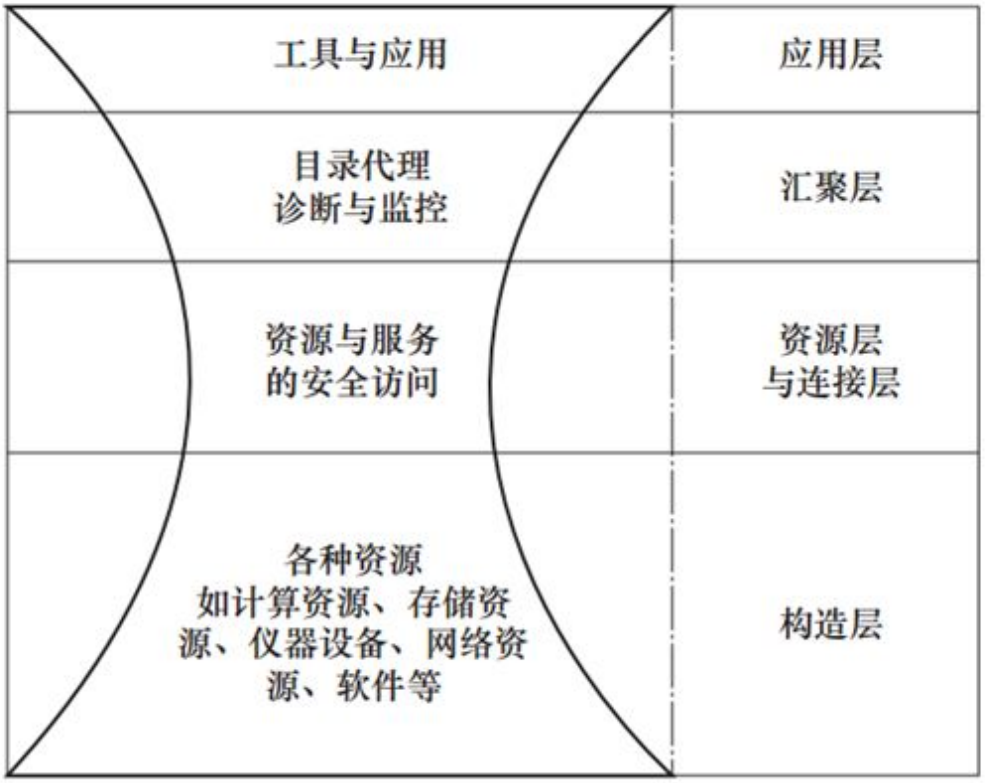
特点：侧重于定性的描述而不是具体的协议定义，容易从整体上进行理解

基本思想：以协议为中心，强调服务与 API 和 SDK 的重要性

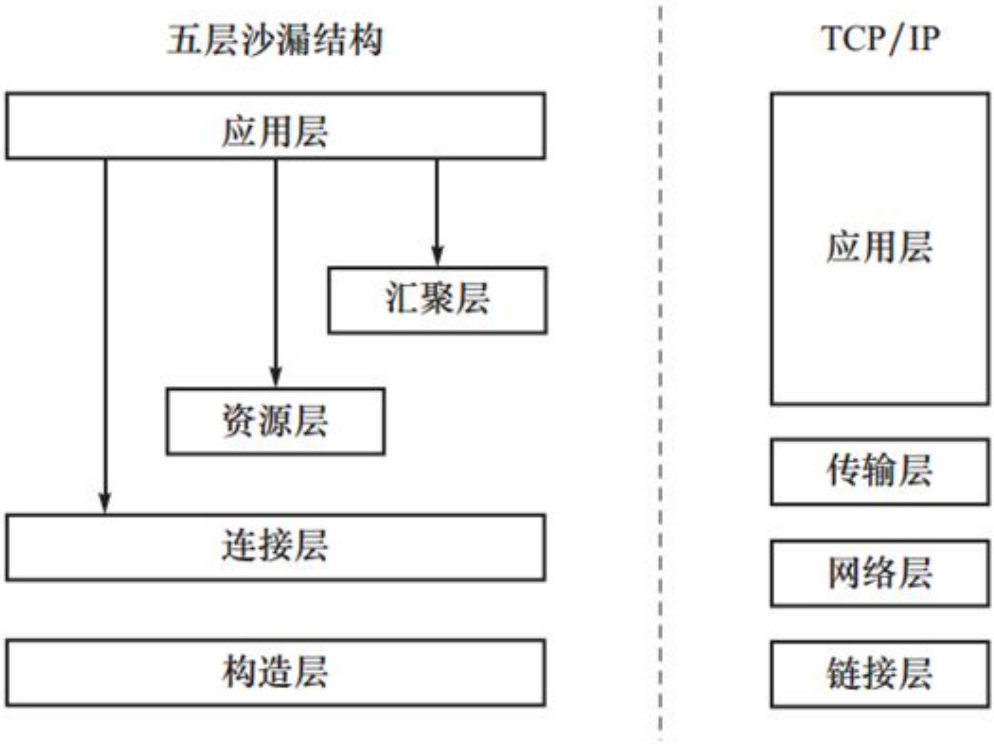
设计原则：保持参与的开销最小

优点：层次清晰，易于理解和实现

缺点：五层中各层的协议数量不同，对于核心的协议，既要能实现上层协议向核心协议的映射，又要能实现核心协议向下层协议的映射，这样核心协议就成为协议层次结构中的瓶颈，形成沙漏。


工具与应用
应用层
目录代理
诊断与监控
汇聚层
资源与服务
的安全访问
资源层
与连接层
各种资源
如计算资源、存储资
源、仪器设备、网络资
源、软件等
构造层

沙漏形状的五层结构



五层沙漏结构与TCP/IP网络协议的对比

9.开放网格服务体系结构（OGSA）的基本思想（其他内容见 PPT 5.2）

OGSA 的基本思想：以服务为中心

在 OGSA 框架中，将一切抽象为服务，包括各种计算资源、存储资源、网格、程序、数据库等。

10.网格资源管理中的负载均衡（同分布式系统资源的负载均衡）

负载均衡建立在现有网络结构之上，它提供了一种廉价、有效、透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

另附：PPT 5.3.2 节 负载均衡的简单划分介绍，自查

分布式系统的负载均衡策略通常分为**静态和动态负载均衡策略**。进一步来说，**静态均衡策略**是利用简单的系统信息，例如子作业数量、平均作业执行周期等相关信息，并通过数学函数调度算法来选择节点，从而分配、执行任务。这种策略的优点是不用总要收集节点的信息，而只需要一些简单地分析和处理上面相关简单信息。由于这种策略不能够动态地调整系统中节点信息的变化，因此一些节点利用率很低。通常的静态负载均衡策略有 SWRSP (speed-weighted randomsplitting policy)和 LDSP(load-dependents static policy)。这两

种策略可根据系统的不同、任务的大小而选择使用。**动态负载均衡策略**是根据系统当前状态或者最近状态去决定如何给分布式系统的每个节点分配任务。如果系统的一些节点任务超载，这个超载任务将被转移给

其它节点来处理，从而达到动

态均衡的目的。但是，任务的转移将给系统带来额外的负担，因为系统不得不为了转移而存储一些用来收集和保持系统状态的资源信息。通常，动态负载均衡策略包括 SQP(shortest queue policy)、NQP (never queue policy)、MTP (maximum throughput policy) 和 ASP(adaptive separable policy)。由上面可知，**静态和动态负载均衡策略各有自己的优缺点**。**静态负载均衡策略**不需要不断地收集系统的状态而只要分析系统相关可用的信息。因此就不存在额外的负担任务。相对说来，如果相关信息不能持续收集和系统环境没能迅速调整的话，每个节点的负载就不能完全调整均衡而且有些节点的利用率也非常低。但是，**动态负载均衡策**略的监控系统能够实时根据每个节点的负载来动态调整任务的分配、执行。但是，持续监控会也给系统带来额外任务而增加了系统的复杂性。据相关研究、实验表明，由于静态负载均衡策略采用的算法简单并且不会给系统带来额外负担，所以在任务执行前静态负载均衡策略能够有更好的效果。如果系统的负载状态频繁改变的话，则动态负载均衡策略比静态负载均衡策略更适合。另外，网格环境中有空闲节点，而每个节点提供的资源是非常动态的并且静态负载均衡策略和动态均衡策略也是不可能完全融合在一起使用。因此，本文提出了混合负载均衡策略，它采用以静态负载均衡策略为主为系统服务，同时结合动态负载均衡策略避免了节点的变化带来系统性能降低的情况。

11.网格系统的安全问题

网格的安全问题主要体现在以下几个方面：

①用户的身份管理和用户验证

②合法用户间的安全通信

③资源访问授权与认证机制

访问网格提供的资源必须由每个资源的授权策略所控制。各种资源需要不同的授权和认证机制，这些机制的改变是由具体网格环境限定的。

④审计

网格的安全问题的来源：

①现有系统和技术的集成及扩展

②不同主机环境协同工作的能力

③相互影响的主机环境之间的信任关系

12.非结构化 P2P 系统的优缺点

缺点：1.洪泛式搜索系统开销大

2.搜索深度有限，不一定能查找到已有文件

优点：1.网络维护开销小

2.容错性和灵活性高

3.支持模糊查询与关键字查询

实例请看书本P193页倒数第10行至倒数第6行

13.基于信誉机制的激励机制

利用节点的信誉度显示了节点在系统中长期的行为表现，信誉度高的节点可以获得更好的服务。非结构化P2P系 统如Gnutell和Freenet由于没有统一的基础设施进行信任信息存储和访问，通过邻居节点对服务搜索请求和信任 信息的获取请求的应答和转发，节点可以获取所需服务和进行信任决策。因此，节点能否获取高质量的服务及正 确的信任信息和邻居节点的选择密切相关，拥有更多良好行为合作的邻居节点，节点更易获得高质量的服务与正 确的信任信息，还可以免受某些类型的攻击，如DOS攻击。反之，具有许多freerider和恶意的不合作邻居节点， 节点更容易获得低质量的服务和不正确的信任信息。

14.云计算、普适计算、网格计算及集群计算

网格计算：网格计算是分布式计算的一种，也是一种与集群计算非常相关的技术。如果我们说某项工作是分布式的，那么，参与这项工作的一定不只是一台计算机，而是一个计算机网络，显然这种“蚂蚁搬山”的方式将具有很强的数据处理能力。网格计算的实质就是组合与共享资源并确保系统安全。网格计算通过利用大量异构计算机的未用资源（CPU周 期和磁盘存储），将其作为嵌入在分布式电信基础设施中的一个虚拟的计算机集群，为解决大规模的计算问题提供一个模型。网格计算的焦点放在支持跨管理域计算 的能力，这使它与传统的计算机集群或传统的分布式计算相区别。网格计算的目标是解决对于任何单一的超级计算机来说仍然大得难以解决的问题，并同时保持解决 多个较小的问题的灵活性。这样，网格计算就提供了一个多用户环境。

集群计算：计算机集群使将一组松散集成的计算机软件和/或硬件连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台计算机。集群系统中的单个计算机通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。集群计算机通常用来改进单个计算机的计算速度和/或可靠性。一般情况下集群计算机比单个计算机，比如工作站或超级计算机性价比要高得多。根据组成集群系统的计算机之间体系结构是否相同，集群可分为同构与异构两种。集群计算机按功能和结构可以分为，高可用性集群（High-availability (HA) clusters）、负载均衡集群（Loadbalancing clusters）、高性能计算集群（High-performance (HPC)clusters）、网格计算（Grid computing）。

高可用性集群：一般是指当集群中有某个节点失效的情况下，其上的任务会自动转移到其他正常的节点上。还指可以将集群中的某节点进行离线维护再上线，该过程并不影响整个集群的运行。

负载均衡集群：负载均衡集群运行时，一般通过一个或者多个前端负载均衡器，将工作负载分发到后端的一组服务器上，从而达到整个系统的高性能和高可用性。这样的计算机集群有时也被称为服务器群（Server

Farm)。 一般高可用性集群和负载均衡集群会使用类似的技术，或同时具有高可用性与负载均衡的特点。Linux 虚拟服务器（LVS）项目在 Linux 操作系统上提供了最常用的负载均衡软件。

高性能计算集群：高性能计算集群采用将计算任务分配到集群的不同计算节点儿提高计算能力，因而主要应用在科学计算领域。比较流行的 HPC 采用 Linux 操作系统和其它一些免费软件来完成并行运算。这一集群配置通常被称为 Beowulf 集群。这类集群通常运行特定的程序以发挥 HPC cluster 的并行能力。这类程序一般应用特定的运行库，比如专为科学计算设计的 MPI 库。HPC 集群特别适合于在计算中各计算节点之间发生大量数据通讯的计算作业，比如一个节点的中间结果或影响到其它节点计算结果的情况。

集群计算与网格计算的区别：（1）简单地，网格与传统集群的主要差别是网格是连接一组相关并不信任的计算机，它的运作更像一个计算公共设施而不是一个独立的计算机。网格通常比集群支持更多不同类型的计算机集合。（2）网格本质上就是动态的，集群包含的处理器和资源的数量通常都是静态的。在网格上，资源则可以动态出现，资源可以根据需要添加到网格中或从网格中删除。（3） 网格天生就是在本地网、城域网或广域网上进行分布的。网格可以分布在任何地方。而集群物理上都包含在一个位置的相同地方，通常只是局域网互连。集群互连技 术可以产生非常低的网络延时，如果集群距离很远，这可能会导致产生很多问题。物理临近和网络延时限制了集群地域分布的能力，而网格由于动态特性，可以提供 很好的高可扩展性。（4）集群仅仅通过增加服务器满足增长的需求。然而，集群的服务器数量、以及由此导致的集群性能是有限的：互连网络容量。也就是说如果一味地想通过扩大规模来提高集群计算机的性能，它的性价比会相应下降，这意味着我们不可能无限制地扩大集群的规模。 而网格虚拟出空前的超级计算机，不受规模的限制，成为下一代 Internet 的发展方向。（5）集群和网格计算是相互补充的。很多网格都在自己管理的资源中采用了集群。实际上，网格用户可能并不清楚他的工作负载是在一个远程的集群上执行的。尽管网格与集群之间存在很多区别，但是这些区别使它们构成了一个非常重要的关系，因为集群在网格中总有一席之地—— 特定的问题通常都需要一些紧耦合的处理器来解决。然而，随着网络功能和带宽的发展，以前采用集群计算很难解决的问题现在可以使用网格计算技术解决了。理解网格固有的可扩展性和集群提供的紧耦合互连机制所带来的性能优势之间的平衡是非常重要的。

云计算：云计算是最新开始的新概念，它不只是计算等计算机概念，还有运营服务等概念了。它是分布式计算、并行计算和网格计算的发展，或者说是这些概念的商业实现。云计算不但包括分布式计算还包括分布式存储和分布式缓存。分布式存储又包括分布式文件存储和分布式数据存储。

普适计算又称普存计算、普及计算（英文中叫做 pervasive computing 或者 Ubiquitous computing）这一概念强调和环境融为一体的计算，而计算机本身则从人们的视线里消失。在普适计算的模式下，人们能够在任何时间、任何地点、以任何方式进行信息的获取与处理。普适计算的核心思想是小型、便宜、网络化的处理设备广泛分布在日常生活的各个场所，计算设备将不只依赖命令行、图形界面进行人机交互，而更

依赖“自然”的交互方式，计算设备的尺寸将缩小到毫米甚至纳米级。普适计算的含义十分广泛，所涉及的技术包括移动通信技术、小型计算设备制造技术、小型计算设备上的操作系统技术及软件技术等。间断连接与轻量计算（即计算资源相对有限）是普适计算最重要的两个特征。普适计算的软件技术就是要实现在这种环境下的事务和数据处理。在信息时代，普适计算可以降低设备使用的复杂程度，使人们的生活更轻松、更有效率。实际上，普适计算是网络计算的自然延伸，它使得不仅个人电脑，而且其它小巧的智能设备也可以连接到网络中，从而方便人们即时地获得信息并采取行动。

云计算与并行、分布式、网格和集群计算的区别：云计算是从集群技术发展而来，区别在于集群虽然把多台机器联了起来，但其某项具体任务执行的时候还是会被转发到某台服务器上，而云可以简单的认为是任务可以被分割成多个进程在多台服务器上并行计算，然后得到结果，好处在于大数据量的操作性能非常好。云可以使用廉价的PC服务器 ，可以管理大数据量与大集群，关键技术在于能够对云内的基础设施进行动态按需分配与管理。云计算与并行计算、分布式计算的区别，以计算机用户来说，并行计算是由单个用户完成的，分布式计算是由多个用户合作完成的，云计算是没有用户参与，而是交给网络另一端的服务器完成的。

并行计算：并行计算是相对于串行计算来说的。可分为时间上的并行和空间上的并行。 时间上的并行就是指流水线技术，而空间上的并行则是指用多个处理器并发的执行计算。例如基于CUDA编程。并行计算的目的就是提供单处理器无法提供的性能（处理器能力或存储器），使用多处理器求解单个问题。

总结：并行的主体 -- 处理器；进程/线程级并行。

分布式计算：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。最近的分布式计算项目已经被用于使用世界各地成千上万位志愿者的计算机的闲置计算能力，通过因特网，可以分析来自外太空的电讯号，寻找隐蔽的黑洞，并探索可能存在的外星智慧生命等。

总结：并行的主体 -- 计算机；各个计算机并行

并行计算与分布式计算的区别：（1）简单的理解，并行计算借助并行算法和并行编程语言能够实现进程级并行（如MPI）和线程级并行（如openMP）。而分布式计算只是将任务分成小块到各个计算机分别计算各自执行。（2）粒度方面，并行计算中，处理器间的交互一般很频繁，往往具有细粒度和低开销的特征，并且被认为是可靠的。而在分布式计算中，处理器间的交互不频繁，交互特征是粗粒度，并且被认为是不可靠的。并行计算注重短的执行时间，分布式计算则注重长的正常运行时间。（3）联系，并行计算和分布式计算两者是密切相关的。某些特征与程度（处理器间交互频率）有关，而我们还未对这种交叉点（crossover point）进行解释。另一些特征则与侧重点有关（速度与可靠性），而且我们知道这两个特性对并行和分布两类系统都很重要。（4）总之，这两种不同类型的计算在一个多维空间中代表不同但又相邻的点。