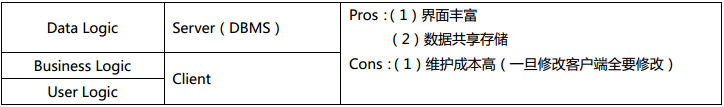
Client/Server计算模型:有多个客户端，至少有一个服务器

2层 c/s架构，即胖客户端

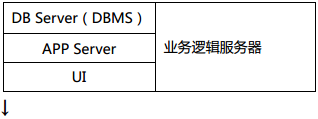


对比优点：比 Standalone 系统能提供更高的性能，因为客户端和服务器端将应用的处理要求分开，同时又共同实现其处理要求，对客户端程序的请求实现“分布式应用处理”。

问题（缺点）：系统的可靠性有所降低。一个客户机/服务器系统是由各自独立开发、制造和管理的各种硬件和软件的混合体，其内在的可靠性不如单一的、中央管理的大型机或小型机，出现问题时，很难立即获得技术支持和帮助。

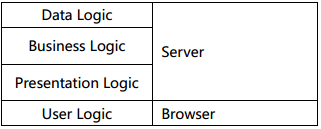
维护费用较高。尽管这种应用模式在某种程度上提高了生产效率，由于客户端需要安装庞大而复杂的应用程序，当网络用户的规模达到一定的数量之后，系统的维护量急剧增加，因而维护应用系统变得十分困难。

3层 C/S架构 N层B/S架构



三层模式中，Web服务器既作为一个浏览服务器、又作为一个应用服务器，在这个中间服务器中，可以将整个应用逻辑驻留在其上，而只有表示层存在存在客户机上。这种结构被称为“瘦客户机”。

优势：利用单一的访问点，可以在任何地点访问站点的数据库；对于各种信息源，无论是文本还是图形都采用相同的界面；所有的信息，无论其基于的平台，都可以采用相同的界面的访问；可跨平台操作；减少整个系统的成本；维护升级较方便；具有良好的开放性

三层C/S结构是将应用功能分为 表示层、功能层和数据层三部分。与二层C/S架构相比，其程序的可维护性要好得多，缺点：但其业务处理所需要的数据要从服务器传给客户机，故需要考虑其对系统的性造成的影响。

维护和升级方式简单。目前，软件系统改进和升级越来越频繁，B/S架构的产品明显体现着更为方便的特性。无论用户规模多大，有多少分支机构都不会增加维护的工作量，所有操只针对服务器进行。

IT鸿沟：指当业务不断发展的时候，现有的IT支持平台的发展不能满足业务的需要，而当IT支持平台发展到可支持业务需要的时候，业务又有了新的需求和发展。

存在的原因：企业中信息孤岛的问题。信息孤岛：相互之间在功能上不关联互助、信息不共享互换以及信息与业务流程和应用相互脱节的计算机应用系统。

解决办法：EAI（企业应用集成）。一套完整的EAI技术层次体系应该包括 应用接口层、应用整合层、流程整合和用户交互 四个层面

层次 由下到上：

应用接口层：解决的是应用集成服务器与被集成系统之间的连接和数据接口的问题。

应用整合层：解决的是被集成系统的数据转换问题，通过建立统一的数据模型来实现不同系统间的信息转换。

流程整合层：它将不同的应用系统连接在一起，进行协同工作，并提供商业流程管理的相关功能，包括流程设计、监控和规划，实现业务流程的管理。

用户交互层：用户在界面上提供一个统一的信息服务功能入口，通过将内部和外部各种相对分散独立的信息组成一个统一的整体，保证了用户既能够从统一的渠道访问其所需的信息，也可以依据每一个用户的要求来设置和提供个性化的服务。

中间件：是一种软件，处于系统软件（操作系统和网络软件）与应用软件之间，它能使处于应用层中的各应用成分之间实现跨网络协同工作（即互操作）。

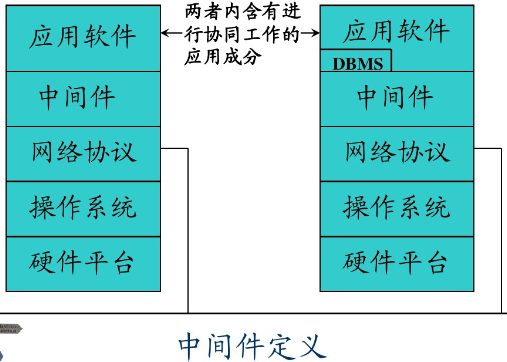
中间件的作用：

1. 提供了通道功能，可保证应用系统数据的可靠传输；

2．起着桥梁作用，达成异构应用资源之间的互联互通；

3. 是分布式应用集成开发框架，促成企业分布式应用的有机集成；

4. 作为平台，可有效实现分布应用的分阶段部署；

5. 起着过滤网作用，可有力保护关键信息的安全。

提供的最基本的系统服务：

通信服务：用户屏蔽低层网络通信接口，为高层应用提供可靠的消息通信和远程访问；

名字服务：提供全局的名字服务空间，屏蔽低层实体分布的具体物理位置；

存储服务：提供全局的一致数据访问，集成低层分布的局部文件系统和数据库；

分布式事务服务：提供分布式事务处理，包括事务管理和调度、负载均衡和失败恢复等；

安全服务：提供存取访问控制，在中间件层次保证分布式系统的安全性。

第二讲：

网格计算：网格计算在每个节点上安装中间件进行彼此间的连接，使用网格计算的用户需要参与提供计算机资源。网格计算利用互联网上计算机的CPU闲置资源处理能力来解决大型计算问题。

网格计算的定义：网格计算是动态多级虚拟组织中的一个协调的共享资源和解决问题的过程。

网格将网络上分布的资源聚合为一体，建立计算和数据处理的通用系统支持平台，实现资源共享和联合工作。 资源：计算、数据、存储、设备、软件和人员等 本质：资源的共享和协同，统一和横向设计 技术：虚拟组织之间的资源共享和协同工作技术

方法：建立广泛的资源共享和协同的标准协议 最终目标：数字信息基础设施。

需满足三个条件：在非集中控制环境下协同使用资源；使用标准的、开放的、通用的协议；提供非凡的服务。

优点：提供高效的资源共享服务，平衡多台计算机上的负载，可把程序放在最合适的位置上。

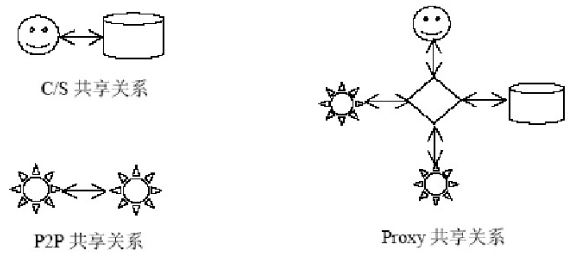
目的：高性能计算 网络：覆盖网络/应用层网络

网格体系结构：5层沙漏模型、开放网格服务体系结构 OGSA( Open Grid Services Architecture)

5层沙漏模型：

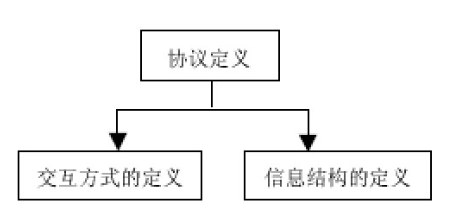
基本思想和概念：共享、互操作性、协议、服务。

共享：深层次、广泛、动态、具有多种形式的有条件受控的共享。



互操作性：共享关系可以跨越不同的平台、语言和编程环境，跨越不同的组织便捷、使用策略和资源类型。

协议：为了实现特定的操作而定义的分布式系统元素之间交互的方式以及交互过程中交换的信息结构。

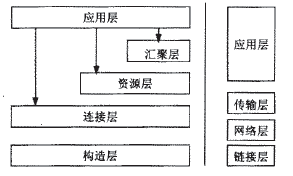


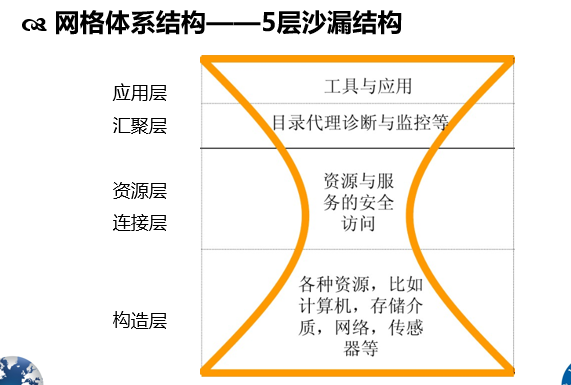
服务：由所使用的协议和所实现的行为来定义。服务抽象掉了与资源相关的信息。

中心思想：采用以协议为中新的分层结构

5层沙漏结构根据各组成部分离共享资源的距离，将对共享资源进行操作、管理和使用的功能分散在五个不同的层次中，越向下层就越接近共享的物理资源，与特定资源相关的成分就越多；越向上层就越感觉不到共享资源的细节特征，也就是说上层是更加抽象的共享资源的表示。

5层沙漏结构：构造层、连接层、资源层、汇聚层、应用层





构造层：其功能是向上提供网格中可共享的资源（物理或逻辑实体）。常用的资源有：处理能力、存储系统、目录、网络资源、分布式文件系统、分布式计算机池、计算机集群。

连接层：网格中网络事务处理、通信与授权控制的核心协议。

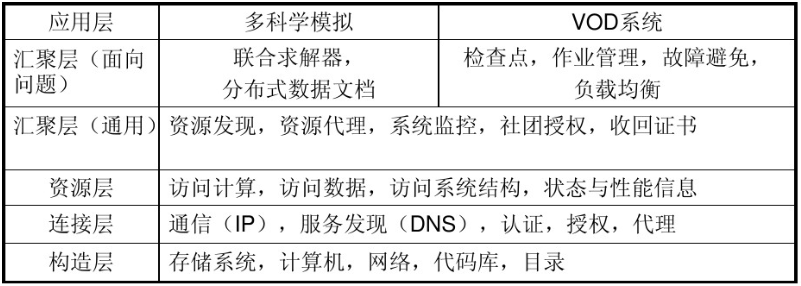
构造层提交的各种资源间的数据交换都在这一层的控制下实现。各资源间的授权验证、安全控制也在这里实现。资源间的数据交换通过传输、路由及名字解析等机制实现。

资源层：该层对单个资源实施控制，与可用资源进行安全握手、资源做初始化、检测资源运行状况，统计与付费等有关资源使用数据。

汇聚层：其作用是将资源层提交的受控资源汇聚在一起，供虚拟组织的应用程序共享、调用。

为了对来自应用层的共享进行管理和控制，汇聚层提供目录服务、资源分配、日程安排、资源代理、资源监测诊断、网格启动、负荷控制、账户管理等多种功能。

应用层：这层是网格上用户的应用程序。应用程序通过各层的API调用相应的服务，再通过服务调用网格上的资源来完成任务。应用程序开发涉及大量的库函数。为了便于网格应用程序的开发，需要创建支持网格计算的库函数。



开放网格体系结构OGSA：面向服务体系结构SOA，以服务为中心的“服务结构”。

云计算与网格计算比较 云计算和网格计算在目标、结构和技术方面有共同之处，但也有许多的区别。

相同之处目标：减少计算成本、增加可靠性、增加灵活性面临很多共同问题：管理大型设备、定义用户发现、请求和使用资源的方法

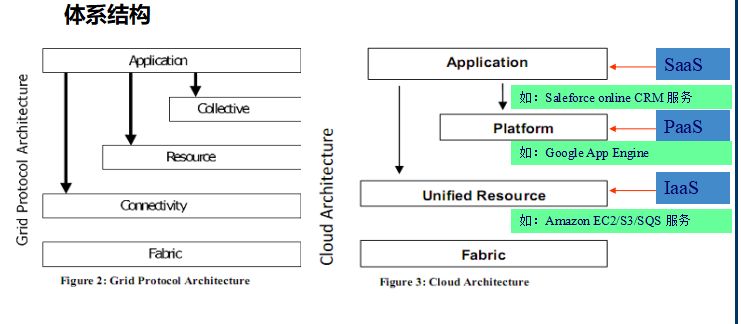
不同之处：

业务模型：

云计算：面向企业或单个用户，按需使用资源，按使用收费

网格计算：面向工程，集中于解决大型复杂的计算问题，一项任务可由来自于不同成员贡献的资源共同完成，资源一般是高性能的计算、存储设备。

体系结构：

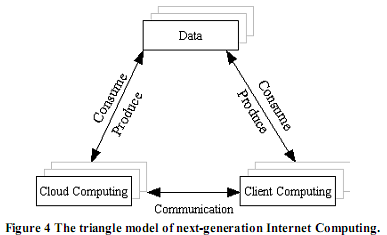


资源管理—计算模型：

网格：批量计算模型，不支持交换式应用云计算：资源同时被所有用户共享，支持延迟敏感的应用

资源管理——数据模型：

网格：在网格环境下有专门的数据网格处理数据集中式的应用，出现了虚拟数据的概念云计算：网络计算是数据、云计算和客户端计算的结合



资源管理——数据位置：

云计算：能够将数据分块，存储于不同的位置，可以将任务调度到靠近数据存储位置的地方执行，提高效率网格计算：数据存储依赖于共享的文件系统，难于利用数据位置信息，需要改进任务调度器为数据意识的。

关键需要的是计算和数据资源管理的综合，在访问模式中利用数据位置信息使数据移动的数据量最小，改善终端应用的性能和可扩展性。

资源管理——虚拟化网格：每一个组织自己控制和管理自身的资源，对虚拟化的依赖程度不如云计算高云计算：虽然不是必要的，但在实践中几乎成为不可缺少的元素

编程模型网格：MPI, Globus Tookit, WSRF 等云计算：MapReduce, Hadoop等

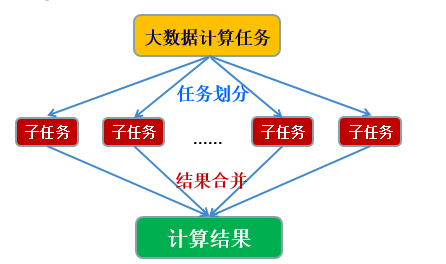
应用模型网格：支持多种不同的应用（高性能计算、高吞吐量计算等）。云计算：能处理和网络计算类似的应用，由于网络环境的不确定性，在高性能计算方面，云计算要稍逊一筹。支持交互式应用。浏览器和Web 2.0技术在用户和网格与云计算交互的过程中扮演重要角色。

安全模型网格：更为严格，网格安全的核心是单点登录、代理和数据隐私等。云计算：更为简单，具有更弱的安全性。采用网络管理为用户创建账户，并允许用户通过邮件在不安全和未加密的通信中重设密码和接收新的密码。新用户通过邮件能方便容易地使用云服务。

云计算安全是云计算研究中的重要课题，普通云用户面临的安全风险主要体现在以下方面：授权的用户接入：企业敏感数据要求只有特权用户才能访问以保证机密性。数据隔离：保证不同用户之间的数据达到完全隔离。数据恢复：有良好的容错机制以保证系统出错时数据不丢失。

MapReduce的目标：用于大规模数据集的并行运算。

（任务分配：） MapReduce模式的思想是将要执行的问题拆成Map(映射)和Reduce（化简）方式，先通过Map程序将数据切割成不相关的区块，分配（调度）给大量计算机处理达到分布运算的效果，再通过Reduce程序将结果汇整，输出开发者需要的结果。



MapReduce可解决哪些算法问题

MapReduce待处理的数据集可以分解成许多小的数据集,而且每一个小数据都可以完全并行地进行处理，因此不能解决不可分拆的计算任务，或者相互间有依赖关系任务，如Fibonacci函数: Fk+2 = Fk + Fk+1 。

MapReduce可解决的基本算法：各种全局数据相关性小、能适当划分数据的计算任务。如：分布式排序、关系代数操作（选择，投影，求交集、并集，连接，成组，聚合等）、矩阵向量相乘、矩阵相乘、词频统计(word count)、文档倒排索引等。

1、在编程的时候，开发者需要编写两个函数：

Map:(in\_key, in\_value)

Reduce:(key,[value 1,value 2…])

2、Map操作产生结果是<key,value>对

3、在Map，Reduce之间系统把同一Key归类到Reduce

3、Reduce操作对相同的Key进行归类处理

数据一致性理论：

CAP理论：

1. 强一致性(Consistency)：系统在执行某项操作后仍然处于一致状态。分布式系统中，更新操作执行成功后所有用户都应该读取到最新的值，这样的系统被认为具有强一致性。
2. 可用性（Availability）：每一个操作总是能够在一定时间内返回结果。
3. 分区容错性（Partition Tolerance）。分区容错性可以理解为系统在存在网络分区的情况下仍然可以接受请求（满足一致性和可用性）

CAP理论在分布式环境中设计和部署系统时需要考虑的三个重要的系统需求。根据CAP理论，数据共享系统只能满足这三个特性中的两个。因此，系统必须在这三个特性之间作出权衡。

在网络分区场景下，如发生网络故障，分区G1发送的数据消息不能传达到另一分区G2，数据将处于不一致状态，不能满足一致性要求，不能满足一致性要求。采用如阻塞、加锁、集中控制等技术来保证数据的一致性，但同步操作必定要消耗一定时间，必然会影响到系统的可用性和分区容错性，尤其在网络规模较大的时候。

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **选择** | **特点** | **例子** |
| 1 | C、A | 两阶段提交、缓存验证协议 | 传统数据库 |
| 2 | C、P | 悲观加锁 | 分布式加锁 |
| 3 | A、P | 冲突处理、乐观 | DNS |

放弃P：如果想避免分区容错性问题的发生，一种做法就是将数据放到一台机器上。虽然无法100%地保证系统不会出错，但不会碰到由分区带来的负面效果。当然，这个选择会严重影响系统的可扩展性。

放弃A：相对于放弃“分区容错性”，其方面就是放弃可用性。一旦遇到分区容错故障，那么受到影响的服务需要等待数据一致，因此等待期间系统无法对外提供服务。

放弃C：这里所说的放弃一致性，并是不完全放弃数据的一致性，而是放弃数据的强一致性，而保留数据的最终一致性。以网络购物为例，对只剩最后一件库存商品，如果同时接收了两份订单，较晚的订单将被告知商品售罄。

最终一致性模型：

分布式系统一般通过复制数据来提高系统的可靠性和容错性，并且将数据的不同副本存放在不同的机器上，由于维护数据副本一致性代价很高，许多系统采用弱一致性来提高性能。不同的一致性模型：强一致性、弱一致性、最终一致性。

强一致性：要求无论更新在哪个数据副本上执行，之后所有的读取操作在同一数据上执行，容易保证强一致性。

弱一致性：在这种一致性下，用户读取到某一操作对系统特定数据的更新需要一段时间。这段时间成为“不一致性窗口”。

最终一致性：是是弱一致性的一种特例，在这种一致性下系统保证用户最终能够读取到某操作对系统特定数据的更新。

此种情况下，“不一致性窗口”的大小依赖于交互延迟、系统负载，以及复制技术中replica的个数（可以理解为master/slave模式中，slave的个数）。 DNS系统是在最终一致性方面最出名的系统，当更新一个域名的IP以后，根据配置策略以及缓存控制策略的不同，最终所有客户都会看见最新值。

最终一致性可划分为多个模型：

因果一致性（Causal Consistency）：假如有相互独立的A、B、C三个进程对数据新操作。进程A对某数据进行更新后并将该操作通知给B，那么B接下来的读操作能够读到A更新的数据值。但是由于A没有将该操作通知给C，那么系统将不保证C一定能够读取到A更新的数据。

读自写一致性（Read Your Own Writes Consistency）：用户更新某个数据后，读取该数据时能够获取其更新后的值，而其他用户读取该数据时则不能保证读取到最新值。

会话一致性（Session Consistency）： 指读自写一致性被限制在一个会话范围内，也就是说提交更新操作的用户在同一个会话里读取该数据时能够保证数据是最新的。 单调读一致性（Monotonic Read Consistency）：指用户读取某个数据值后，后续操作不会读到该数据更早版本的值。

时间轴一致性（Timeliness Consistency）：要求数据所有副本以相同顺序执行所有更新操作，也称为单调写一致性（Monotonic Write Consistency）。

CAP理论指出一致性、可用性和分区容错性不能同时满足。对于数据不断增长的系统（如社会计算、网络服务的系统），他们对可用性及分区容错性的要求高于强一致性，并且很难满足事务所要求的ACID特性，因此BASE理论被提出。

ACID:

事务是用户定义的一个数据库操作序列，这些操作要么全不做，要么全做，是一个不可分割的单位，ACID是事务所具有的特性。

原子性（Atomicity）：事务中的操作要么都做，要么都不做。

一致性（Consistency）：系统必须始终处于强一致性状态下。

隔离性（lsolation）：一个事务的执行不能被其他事务所干扰。

持续性（Durability）：一个已提交的事务对数据库中数据的改变是永久性的。

ACID特性是传统关系型数据库中事务管理的重要任务，也是恢复和并发控制的基本单位。

BASE方法通过牺牲一致性和孤立性来提高可用性和系统性能

BASE分别代表：

基本可用（Basic Availability）：系统基本能够运行、一直提供服务。 软件状态（soft-state）：系统不要求强一致性。

最终一致性（Eventually consistency）：系统需要在某一刻后达到一致性要求。



Paxos算法：

问题：假设许多节点都可以提交决议（value），一致性算法需要保证在所有被提出的决议中只有一个能被接受。某个决议一旦被选择，其他节点能够知道所选择的决议。一致性需要满足一下条件：

决议只有被提出后才能被选择

算法的执行实例只能选择一个决议

决议只有被选中后才能让其他节点（learner）知道。

Paxos算法中 三个角色：提议者(proposer)：提出决议

批准者（acceptor）：负责批准决议

学习者（learner）：主要学习决议

在具体实现中，一个节点可以担任多个角色。

假设节点之间通过 发送信息进行通信，这里使用异步、非拜占庭模型。在该模型中：

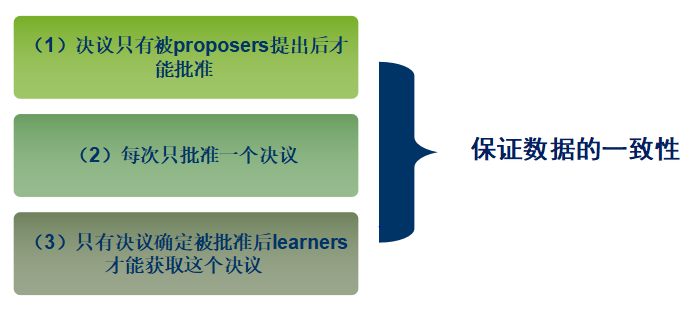
节点以任意的速度进行操作，可能因为故障而停止，也可以重新启动。并且节点所选择的决议不会因为重启等故障消失。

消息可以延迟发送、多次发送或丢失，但不会被篡改（即不存在拜占庭问题）

Paxos算法---选择决议：

采用唯一的批准者，当批准者发生故障后，规则如下：当决议被大多数批准者所批准，则表明决议被批准。（如果一个批准者最多只能批准一个决议，“大多数”设置为超过半数即可，保证系统只有一个决议被批准）

在一个系统中，一个批准者应该需要被设置为可以批准多个决议。以数据存储为例，分布式系统中不可能只存一份数据。为区分批准者所批准的决议，每个决议都分配有版本号和提议值。



P2P：对等网络（Peer-to-Peer Networks）是分布式系统和计算机网络相结合的产物，在应用领域和学术界获得了广泛的重视和成功，被称为“改变Internet的一代网络技术”。

Peer指网络结点，在行为上是自由的----任意加入、退出，不受其他结点的限制，匿名；

在功能上是平等的---不管实际能力的差异；在连接上是互联的---直接/间接，任两结点可建立逻辑链接，对应物理网上的一条IP路径。 充分利用网络带宽、节点资源、提高工作效率。

DHT算法：

定位（Locating）：节点ID和其存放的<K,V>对中的K存在着映射关系，因此可由K获得存放该<K，V>对的节点ID

路由（Routing）：在覆盖网上根据节点ID进行路由，将查询消息最终发送到目的节点。每个节点需要到其临近节点的路由信息。

网络拓扑： 拓扑结构由节点ID和其存放的<K,V>对中的K之间的映射关系决定。拓扑动态变化，需要处理节点加入、退出、失效的情况。

在重叠网上节点始终由节点ID标识，并且根据ID进行路由。

结构化P2P：直接根据查询内容的关键字定位其索引的存放节点。

Chord环（环形拓扑），核心思想：解决P2P应用中遇到的基本问题：如何在P2P应用中找到特定数据的节点

Chord使用一致性哈希作为哈希算法。在Chord协议中将其规定为SHA-1。

应用程序接口：

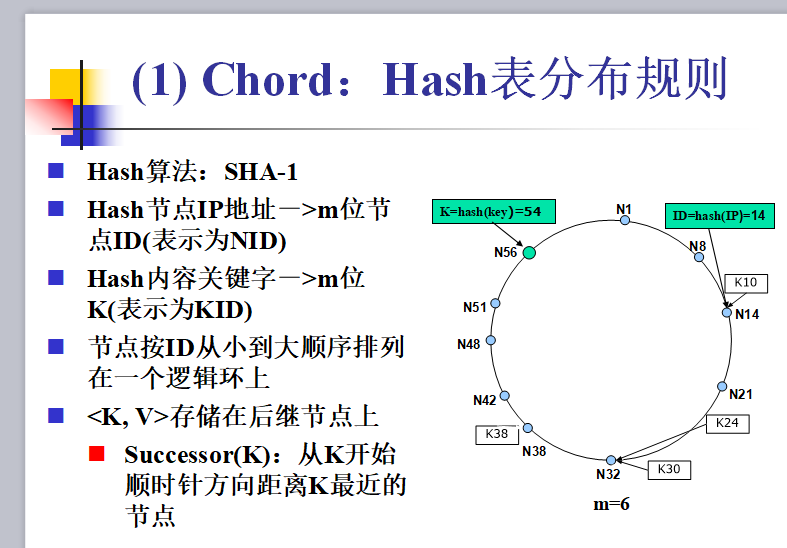
Insert(K , V)：将<K,V>对存放到节点ID为Successor（K）上。

Lookup(K , V)：根据K查询相应的V

Update(K , new\_V)：根据K更新相应的V

Join(NID)：节点加入

Leave()：节点主动退出



Chord小结：

1. 算法简单
2. 负载平衡：所有节点以同等的概率分担系统负荷，从而避免某些节点负载过大
3. 可扩展性：查询过程的通信开销和节点维护状态随着系统总节点数增加成对数关系（O（logN）数量级）
4. 可用性：要求节点根据网络变化动态更新查询表，能够及时恢复路由关系，使得查询可靠地进行。
5. 拓扑适配问题

Pastry概述：

考虑网络的本地性，解决物理网络和逻辑网络的拓扑适配问题：基于应用层定义的邻近性度量，例如IP路由跳数、地理距离、往返延时等。 节点ID分布采用环形结构。

Pastry查询过程：

当一个K为D的查询消息到达节点A

1.节点A首先看D是否在当前节点的叶子节点集中，如果是，则查询消息直接被装到目的节点，也就是叶子节点集中节点ID与D数值最接近的那个节点（有可能就是当前节点）

2.在路由表中根据最长前缀优先的原则选择一个节点作为路由目标，转发路由消息，如果该表项不为空，则将查询消息直接转发到该节点，否则进行下一步。

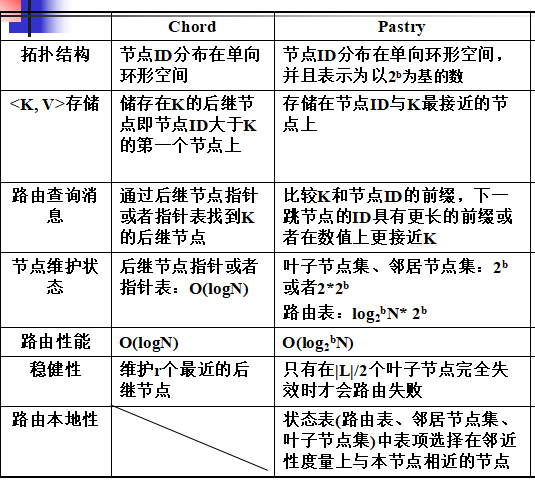
3．如果不存在这样的节点，当前节点将会从其维护的所有节点集合（包括路由表叶子集合及邻居集合中的节点）中选择一个距离消息键值最近的节点作为转发目标。

逻辑跳数：

Pastry节点加入：

初始化状态表

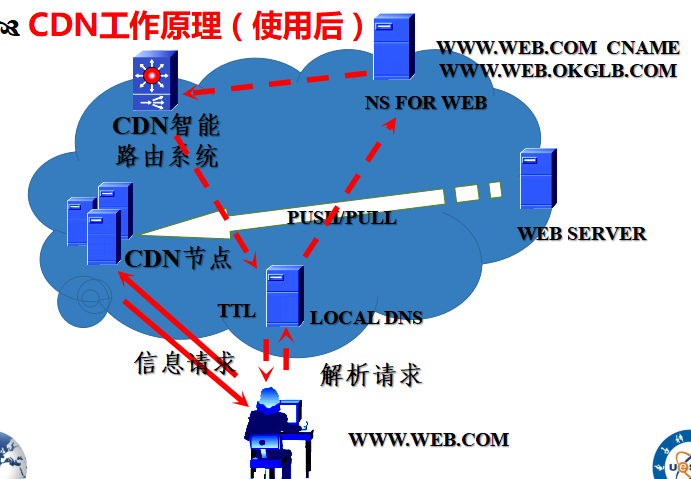
1. 新节点开始使知道一个根据邻近性度量的接近自己的节点A，节点A可以通过使用扩展环IP组播等机制自动定位，或由系统管理员通过其他手段获得。
2. 新节点通过运行SHA-1算法计算自己的IP地址的摘要得到节点ID为X



CDN（Content Delivery Network，内容分发网络）网络：

目的：通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络边缘，使用户可以就近取得所需的内容，解决Internet网络拥挤的状况，提高用户访问网站的响应速度。

从技术上解决由于网络带宽小、用户访问量大、网点分布不均等原因，解决用户访问网站的响应速度慢 的根本原因。



工作原理：当用户访问某域名上的内容信息时，首先需通过域名解析系统将该域名转换为存储该内容信息的内容服务器的IP地址，再根据该IP地址访问内容服务器以获取内容信息。如果该域名使用了CDN技术，该域名上的内容信息将会被复制并发布到广泛分布的、位于不同地理位置的CDN网络的边缘节点服务器。用户访问使用CDN技术的域名时，其地理位置附近的CDN网络的边缘节点服务器向其提供相关内容信息。

传统未加缓存服务的访问过程VS 使用CDN缓存后网站访问过程

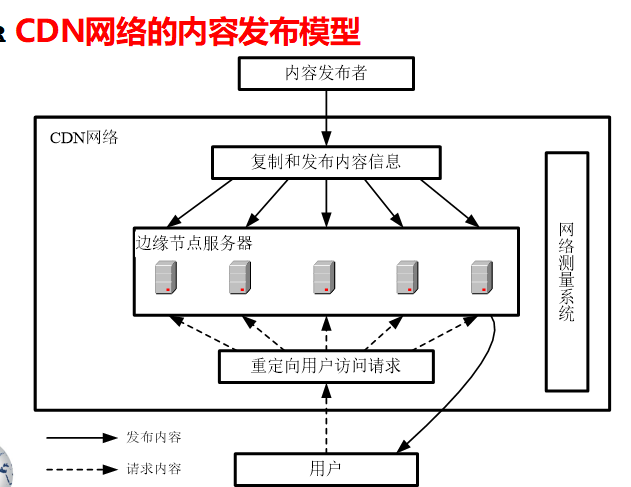
传统：用户提交域名 ---🡪 浏览器对域名进行解释 ---🡪 得到目的主机的IP地址 -🡪

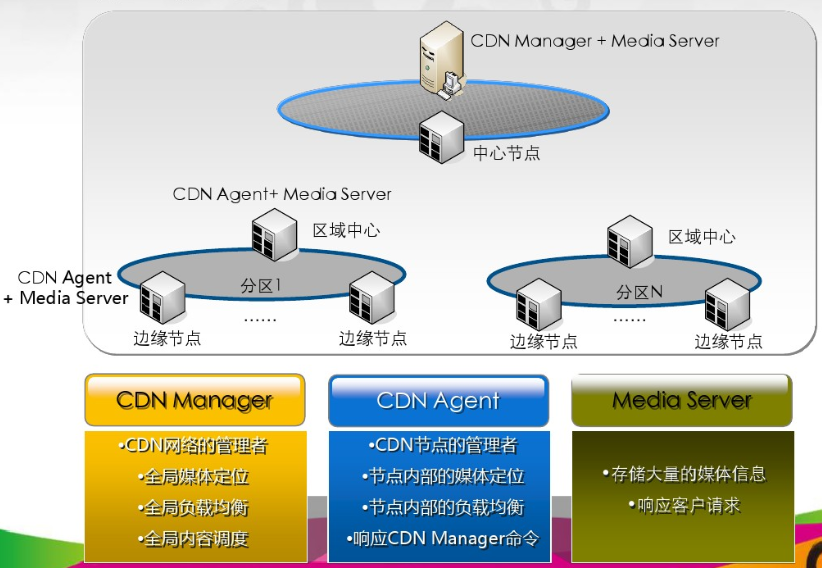
根据目的ip地址访问发出请求 ---🡪 得到请求数据并回显

CDN：用户提交域名---🡪浏览器对域名进行解释--🡪 CDN域名服务器返回指定域名的CNAME记录------🡪对CNAME记录进行再解释-----🡪得到CDN缓存服务器的地址----🡪 根据ip地址发出访问请求-----🡪缓存服务器通过内部DNS得到真实主机地址-----🡪向真实主机发出请求，并把请求返回客户端----🡪客户端浏览器得到请求数据并回显

传统访问模式 VS CDN访问模式

传统： 传统DNS解析过程在将 网站主机域名主机转换未IP地址时，并不预先判断该服务器是否正常工作；即使该服务器已经宕机不能提供服务了，用户的请求仍将被发往这台服务器，造成服务的中断。不同网络间的兼容以及不同网络运营商ISP之间的传输瓶颈等问题使得数据的流通受到限制。





中心节点：全局内容发布中心，全局内容存储点，服务区域中心未命中的用户，CDN网络架构的二级缓冲，提供全局冗余能力

区域中心：服务边缘节点未命中的用户、作为整个CDN网络架构的一级缓冲，承担部分冗余

边缘节点：就近部署，存储热点服务，服务大部分用户，配置基本要素包括负载均衡，+cache，节点能力和网络出口相关、组网要考虑高可靠性。

Akamai CDN技术架构：

CDN关键技术：

内容管理（内容、索引、访问状态信息收集）

内容存储（基于缓存技术）

内容分发（主动push、被动pull）

内容路由（负载均衡）

确保用户就近性和服务器负载判断，CDN确保内容以一种高效的方式为用户的请求提供服务。

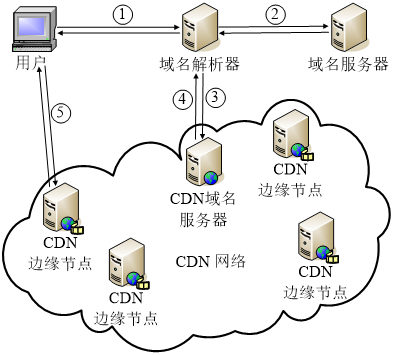
内容路由技术，即CDN实现负载均衡技术。

负载均衡是整个CDN的核心，负载均衡的准确性和效率直接决定了整个CDN的效率和性能。

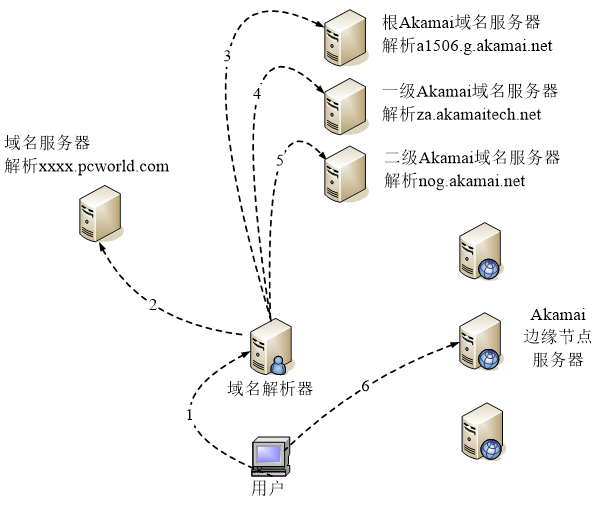
负载均衡技术将网络流量尽可能均匀地分配到几个能完成相同任务的服务器或网络节点上处理，避免部分网络节点过载而另一部分节点空闲的不利状况，既可以提高网络流量，又可以提高网络的整体性能。 CDN负载均衡系统实现CDN的内容路由功能，将用户导向CDN网络中最佳的节点。

内容路由的实现机制：

LimeLight CDN的主要实现方法为：IP Anycast Akamai CDN的主要实现方法为DNS重定向。



DNS重定向



首先，域名解析器访问域名xxxx.pcworld.com的域名服务器，请求解析该域名。该域名的域名服务器返回一个CNAME（Canonical NAME）类型的DNS记录。该CNAME类型的DNS记录指示了所请求解析的域名xxxx.pcworld.com的别名a1506.g.akamai.net。其作用是通过返回一个Akamai CDN网络的内部域名，将域名解析请求重定向到Akamai CDN网络的域名服务器，即根Akamai域名服务器。

然后，域名解析器访问根Akamai域名服务器，请求解析域名a1506.g.akamai.net。根Akamai域名服务器根据域名解析器的IP地址和网络状况等动态信息，返回CNAME类型的DNS记录，即域名a1506.g.akamai.net的CNAME类型的DNS记录为za.akamaitech.net，将域名解析请求重定向到一级Akamai域名服务器。 同样的方法，域名解析器访问一级Akamai域名服务器，请求解析域名za.akamaitech.net。一级Akamai域名服务器也根据域名解析器IP地址和网络状况等动态信息，返回CNAME类型的DNS记录，即域名za.akamaitech.net的CNAME类型的DNS记录为nog.akamai.net，再将域名解析请求重定向到二级Akamai域名服务器。 最后，域名解析器访问二级Akamai域名服务器，请求解析域名nog.akamai.net。二级Akamai域名服务器根据域名解析器IP地址和网络状况等动态信息，返回A类型的DNS记录，即Akamai CDN网络边缘节点服务器的IP地址。通常情况下，二级Akamai域名服务器将返回2个IP地址，用户可选择其中的任意一个边缘节点服务器发出访问请求。

物联网体系结构：

按三层阐述：感知层、网络层、应用层

感知层：包括二维码标签和识读器、RFID标签和读写器、摄像头、CPS（Cyber Physical Systems，网络化物理系统）、各种传感器、传感器网络（指由大量各类传感器节点组成的自治网络，具有自组织、自愈合的特点）、T2T&H2T&H2H终端和传感器网关等，感知层的主要作用是感知和识别物体，采集并捕获信息。

网络层：包括各种通信网络与互联网形成的融合网络，这被普遍认为是最成熟的部分，除此之外还包括物联网管理中心、信息中心等利用网络的能力对海量信息进行智能处理的部分。也就是说网络层不但要具备网络运营的能力，还要提升信息运营的能力。网络层是物联网成为普遍服务的基础设施。 应用层：是将物联网技术与行业专业技术相结合，实现广泛智能化应用的解决方案集。物联网通过应用层最终实现信息技术与行业的深度融合，对国民经济和社会发展具有广泛影响。应用层的关键问题在于信息的社会化共享以及信息安全的保障。

按五层阐述：感知层、接入层、网络层、中间件层、应用层

物联网的概念： 广义来讲，物联网是一个未来发展的愿景，等同于“未来的互联网”，或者是“泛在网络”，能够实现人在任何时间、地点，使用任何网络与任何人与物的信息交换，以及物与物之间的信息交换；狭义来讲，物联网是物品之间通过传感器连接起来的局域网，不论接入互联网与否，都属于物联网的范畴。

物联网的一种定义是：通过射频识别（RFID）、红外感应器、全球定位系统（GPS）、激光扫描器等信息传感设备，按约定的协议，把任何物品与互联网连接起来，进行信息交换和通讯，以实现智能化识别、定位、跟踪、监控和管理的一种网络。

