1. **Iii 描述企业的IT鸿沟、信息孤岛，给出包含EAI的企业级应用基本架构。**

**IT鸿沟**指的是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支持平台发展到可以支持业务的需要的时候，业务又有新的需求和发展，这样业务与IT支持之间持续存在一个鸿沟。

存在IT鸿沟的一个重要的原因就是企业中的**信息孤岛**问题。**信息孤岛**指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。

IT鸿沟的解决办法是EAI（企业应用集成）。一套完整的EAI层次体系包括应用接口层、应用整合层、流程整合层和用户交互层。EAI使得企业众多信息系统都与一个由中间件组成的底层基础平台相连接，各种“应用孤岛”、“信息孤岛”通过各自的“适配器”连接到一个总线上，然后再通过一个消息队列实现各个应用之间的交流。

EAI技术层次体系最下面的一层是**应用接口层**，它要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。

再往上去就是**应用整合层**，它要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。

应用整合层之上是**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。

到了最上端的**用户交互层**，则是为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。

**中间件定义**

中间件（Middleware）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的协同工作（也就是互操作），这是运行各应用软件之下所涉及的“系统结构、操作系统、通信协议、数据库和其他应用服务”各不相同。

1. **C/S和B/S的优缺点**

二层C/S结构的优点：

1. C/S体系结构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受；
2. 系统的客户应用程序和服务器构件分别运行在不同的计算机上，系统中每台服务器都可以适合各构件的要求，这对于硬件和软件的变化显示出极大的适应性和灵活性，而且易于对系统进行扩充和缩小。
3. 在C/S体系结构中，系统中的功能构件充分隔离，客户应用程序的开发集中于数据的显示和分析，而数据库服务器的开发则集中于数据的管理，不必在每一个新的应用程序中都要对一个DBMS进行编码。将大的应用处理任务分布到许多通过网络连接的低成本计算机上，以节约大量费用。

二层C/S结构的缺点：

1. 开发成本较高；
2. 客户端程序设计复杂；
3. 信息内容和形式单一；
4. 用户界面风格不一，使用繁杂，不利于推广使用；
5. 软件移植困难；
6. 软件维护和升级困难；
7. 新技术不能轻易应用。

三层C/S结构的优点：

1. 允许合理地划分三层结构的功能，使之在逻辑上保持相对独立性，能提高系统和软件的可维护性和可扩展性。
2. 允许更灵活有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层；并且这些平台和各个组成部分可以具有良好的可升级性和开放性。
3. 应用的各层可以并行开发，可以选择各自最适合的开发语言。
4. 利用功能层有效地隔离开表示层与数据层，未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，为严格的安全管理奠定了坚实的基础。

三层C/S结构的缺点：

1. 三层C/S结构各层间的通信效率若不高，即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。
2. 设计时必须慎重考虑三层间的通信方法、通信频度及数据量。这和提高各层的独立性一样是三层C/S结构的关键问题。

B/S体系结构的优点：

1. 基于B/S体系结构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。
2. B/S体系结构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

B/S体系结构的缺点：

1. B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能
2. B/S体系结构的系统扩展能力差，安全性难以控制
3. 采用B/S体系结构的应用系统，在数据查询等响应速度上，要远远地低于C/S体系结构
4. B/S体系结构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理应用

**C/S概念：**C/S 架构是一种典型的两层架构，其全程是Client/Server，即客户端服务器端架构，其客户端包含一个或多个在用户的电脑上运行的程序，而服务器端有两种，一种是数据库服务器端，客户端通过数据库连接访问服务器端的数据；另一种是Socket服务器端，服务器端的程序通过Socket与客户端的程序通信。  
C/S 架构也可以看做是胖客户端架构。因为客户端需要实现绝大多数的业务逻辑和界面展示。这种架构中，作为客户端的部分需要承受很大的压力，因为显示逻辑和事务处理都包含在其中，通过与数据库的交互（通常是SQL或存储过程的实现）来达到持久化数据，以此满足实际项目的需要。

**优点：**1.C/S架构的界面和操作可以很丰富。  
2.安全性能可以很容易保证，实现多层认证也不难。  
3.由于只有一层交互，因此响应速度较快。

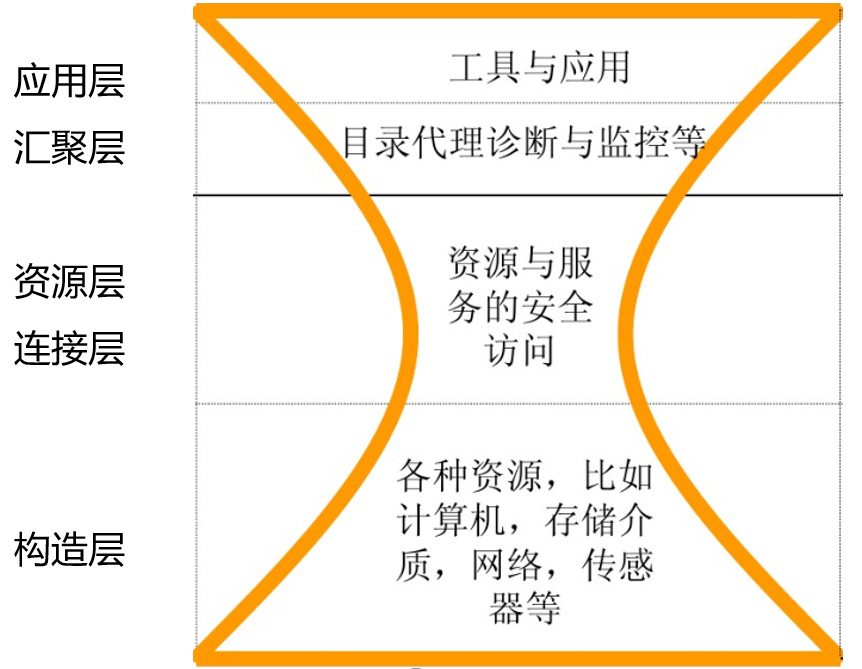
**缺点：**1.适用面窄，通常用于局域网中。  
2.用户群固定。由于程序需要安装才可使用，因此不适合面向一些不可知的用户。  
3.维护成本高，发生一次升级，则所有客户端的程序都需要改变。

**B/S概念：**B/S架构的全称为Browser/Server，即浏览器/服务器结构。Browser指的是Web浏览器，极少数事务逻辑在前端实现，但主要事务逻辑在服务器端实现，Browser客户端，WebApp服务器端和DB端构成所谓的三层架构。B/S架构的系统无须特别安装，只有Web浏览器即可。  
B/S架构中，显示逻辑交给了Web浏览器，事务处理逻辑在放在了WebApp上，这样就避免了庞大的胖客户端，减少了客户端的压力。因为客户端包含的逻辑很少，因此也被成为瘦客户端。

**优点：**1）客户端无需安装，有Web浏览器即可。  
2）BS架构可以直接放在广域网上，通过一定的权限控制实现多客户访问的目的，交互性较强。  
3）BS架构无需升级多个客户端，升级服务器即可。

**缺点：**1）在跨浏览器上，BS架构不尽如人意。  
2）表现要达到CS程序的程度需要花费不少精力。  
3）在速度和安全性上需要花费巨大的设计成本，这是BS架构的最大问题。  
4）客户端服务器端的交互是请求-响应模式，通常需要刷新页面，这并不是客户乐意看到的。（在Ajax风行后此问题得到了一定程度的缓解）

1. **画出沙漏模型层次图，并解释各层功能。**



**中心思想**：采用以协议为中心的分层结构。

构造层：提供网格服务的基础硬件并控制局部的资源，包括：计算、存储、网络等。

连接层：将硬件连接起来实现相互通信，方便查找和使用，并保证访问安全。

资源层：将硬件转换成资源，并为资源提供管理和维护功能。有限的资源构成瓶颈。五层模型中，资源层和连接层共同组成瓶颈部分，使得该结构呈沙漏形状。

汇聚层：协调多种资源的共享，将所有单个的资源形成一个网络。

应用层：使用资源的用户应用程序。

五层结构中各层协议的数量是不同的，对于最核心的协议，既要能实现上层各种协议向核心协议的映射，同时又要能实现核心协议向下层协议的映射，核心协议在所有支持网格计算的地点都应该得到支持，因此核心协议的数量不应该太多，这样核心协议就形成了协议层次结构中的瓶颈，形成沙漏形状。

1. **列举云计算、网格计算的主要差别。**
2. 网格计算的思路是聚合分布资源，支持虚拟组织，提供高层次的服务。云计算的资源相对集中，主要以数据中心的形式提供底层资源的使用。
3. 网格计算用聚合资源来支持挑战应用，要把分散的资源聚合起来，后来逐渐强调适应普遍的信息化应用。云计算从一开始就支持广泛企业计算、Web应用，普适性更强。
4. 网格计算用中间件屏蔽异构系统，力图使用户面向同样的环境，把困难留在中间件，让中间件完成任务。云计算实际上承认异构，用镜像执行或者提供服务的机制来解决异构性的问题。
5. 网格计算用执行作业形式使用，在一个阶段内完成作用产生数据。云计算支持持久服务，用户可以利用云计算作为其部分IT基础设施，实现业务的托管和外包。
6. 网格计算更多地面向科研应用，商业模型不清晰。而云计算从诞生开始就是针对企业商业应用，商业模型比较清晰。
7. 云计算是以相对集中的资源，运行分散的应用。网格计算则是聚合分散的资源，支持大型集中式应用。

但从根本上来说，从应对Internet的应用的特征特点来说，他们是一致的，为了完成在Internet情况下支持应用，解决异构性、资源共享等等问题。

1. **简述Map-Reduce和Hadoop的基本原理。**

Map-Reduce的目标是用于大规模数据集（大于1TB）的并行运算，在面对节点失效的情况时能够保证大量的文件和数据依然可用。初始化时数据被分割成许多分块的小数据，这些数据都是以<key, value>的形式存储。用户程序会分配一个Master进程和许多Worker进程。

任务开始时，Master将用户程序的工作分成两种类型的任务（Map任务和Reduce任务），并将这些任务分配给相应的Workers。

Master的责任是给Map Workers和Reduce Workers分配相应的任务，检测是否有Workers进程死掉，将Map任务处理后的结果通知给Reduce任务。

Map Worker在得到Master的Map工作任务的指令后，Map Workers开始工作。Map Workers从input中获取分块的数据，并通过其中数据存储的方式对数据进行处理，并将处理的中间结果（<key, value>形成的链表）存放在本地磁盘中。

得到Master的有关Map处理结果的通知后，Reduce任务开始工作。Reduce工人把每个Map Workers的本地磁盘中的内容进行整合，形成有序的最终结果，并将结果写入到输出文件中。

假如一个Workers进程失败了，该Workers的任务就被分配给其他的Workers进程。如果一个Master进程失败了，那么整个Map-Reduce任务就会终止。

Hadoop是一个分布式系统基础架构。用户可以充分利用集群的威力进行高速运算和存储。Hadoop的框架核心的设计就是：HDFS和Map Reduce。HDFS为海量的数据提供了存储，则Map Reduce为海量的数据提供了计算。

Hadoop实现了一个分布式文件系统（HDFS）。HDFS有高容错性的特点，部署在低廉硬件上，提供高吞吐量来访问应用程序的数据，适合那些有着超大数据集的应用程序。

1. **描述数据一致性理论。**

CAP理论：

1. 强一致性（C）。系统在执行某项操作后仍然处于一致状态。在分布式系统中，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（A）。每一个操作总是能够在一定的时间内返回结果。
3. 分区容错性（P）。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。

CAP理论是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，而不能同时满足三个条件。

|  |  |  |
| --- | --- | --- |
| **选择** | **特点** | **例子** |
| C、A | 两阶段提交、缓存验证协议 | 传统数据库 |
| C、P | 悲观加锁 | 分布式加锁 |
| 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：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。

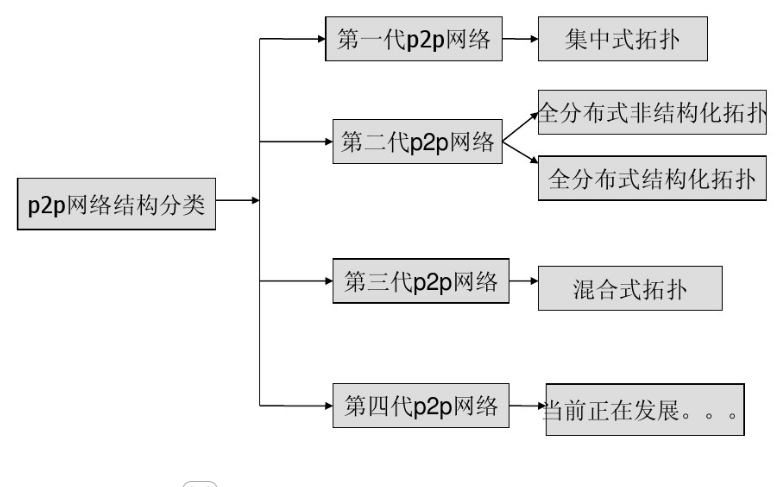
放弃C：这里所说的放弃一致性，并是不完全放弃数据的一致性，而是放弃**数据的强一致性**，而保留**数据的最终一致性**。

1. **简述P2P网络，阐述第一代P2P和第二代P2P的区别。**

Peer to Peer对等网络是一种分布式网络，网络的参与者共享他们所拥有的一部分硬件资源，这些共享资源需要由网络提供服务和内容，能被其它对等节点直接访问而无需经过中间实体。在此网络中的参与者既是资源提供者，又是资源获取者。

第二代与第一代区别：

1. 没有集中的目录服务器，但是拓扑结构有意义。
2. 使用**DHT技术**，有较好的可伸缩性和查询效率。
3. 使用精确的定位算法和特殊的路由协议使得搜索效率提高。



1. **Chord算法原理**

特点：

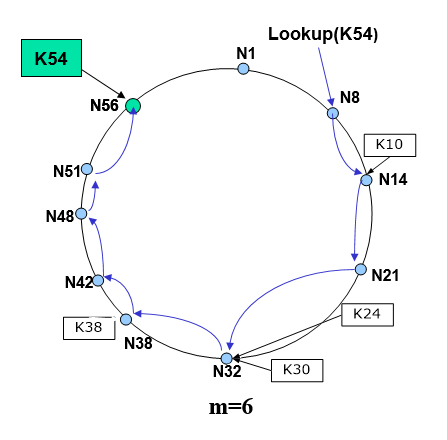
1. 采用环形拓扑(Chord环)
2. 其核心思想就是要解决在P2P应用中遇到的基本问题：如何在P2P网络中找到存有特定数据的节点
3. Chord使用一致性哈希作为哈希算法，在Chord协议中将其规定为SHA-1。

加入过程：

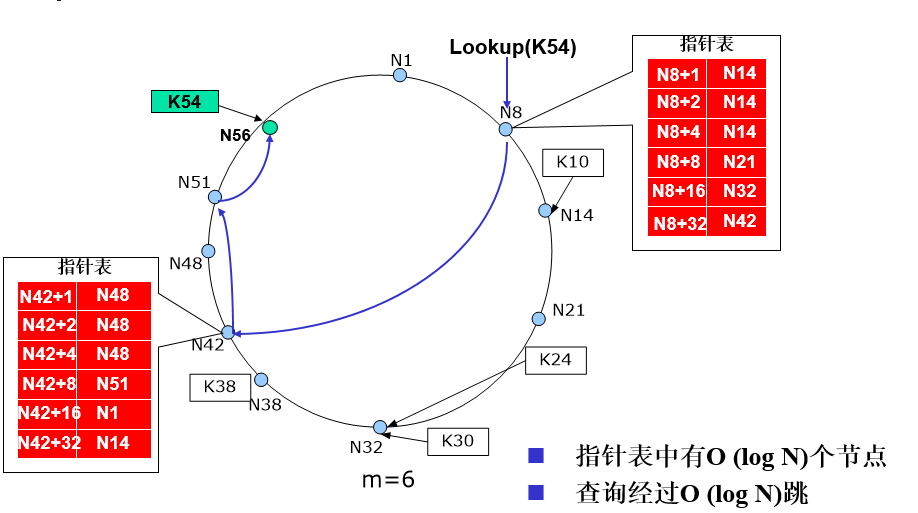
1. Hash算法：SHA-1
2. Hash节点IP地址得到m位节点ID（表示为NID）
3. Hash内容关键字－>m位K（表示为KID）
4. 节点按ID从小到大顺序排列在一个逻辑环上
5. 键值对<K, V>存储在后继节点上。即NIDi < K < NIDi+1时，键值对存在NIDi+1上。也就是从K开始顺时针方向距离K最近的节点

改进前：

1. 每个节点仅维护其后继节点ID、IP地址等信息
2. 查询消息通过后继节点指针在圆环上传递
3. 直到查询消息中包含的K落在某节点ID和它的后继节点ID之间
4. 速度太慢 O(N)，N为网络中节点数



改进后：Chord中每个节点上都有一张局部指针表，表中存放着2m个节点信息（m是二进制位数）。



查询过程（改进后）

如上图所示，假设当前节点是Nc，要查找Ko 。

1. 查找当前节点是否有要查找的Ko。
2. 在指针表里面查找Ko的位置。如果找到，即Nc + 2i = Ko （i∈[0,m-1]），则直接转到对应节点Nc + 2i；若没有找到，则查找临界节点。若Nc + 2i < Ko < Nc + 2i+1 i∈[0,m-1），转到Nc + 2i 对应的节点处继续查找；若Ko> Nc + 2m-1 ，即最后一项的ID < Ko，则在Nc + 2m-1对应节点继续查找。并重复1)

节点加入

1. 新节点N事先知道某个或者某些节点，并且通过这些节点初始化自己的指针表，也就是说，新节点N将要求已知的系统中某节点为它查找指针表中的各个表项
2. 在其他节点运行探测协议后，新节点N将被反映到相关节点的指针表和后继节点指针中
3. 新结点N的第一个后继结点将其维护的小于N节点的ID的所有K交给该节点维护

优点：

1. 算法简单
2. 负载平衡：所有的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：查询过程的通信开销和节点维护的状态随着系统总节点数增加成对数关系(O (log N)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。

缺点：拓扑失配。可能本来的资源节点离自己很近，但却要通过逻辑节点绕一圈过来。

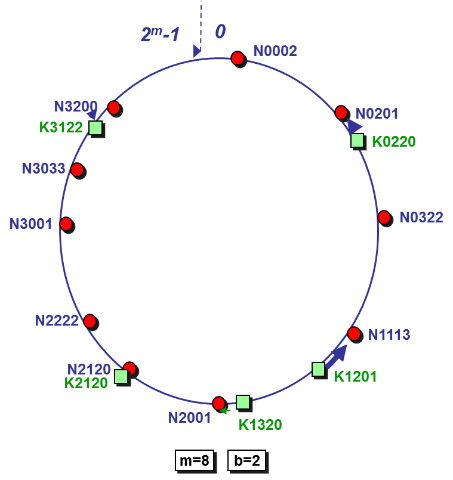
**简述Pastry算法。**

特点：

1. 考虑网络的本地性，解决物理网络和逻辑网络的拓扑失配问题。
2. 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。
3. 节点ID分布采用环形结构

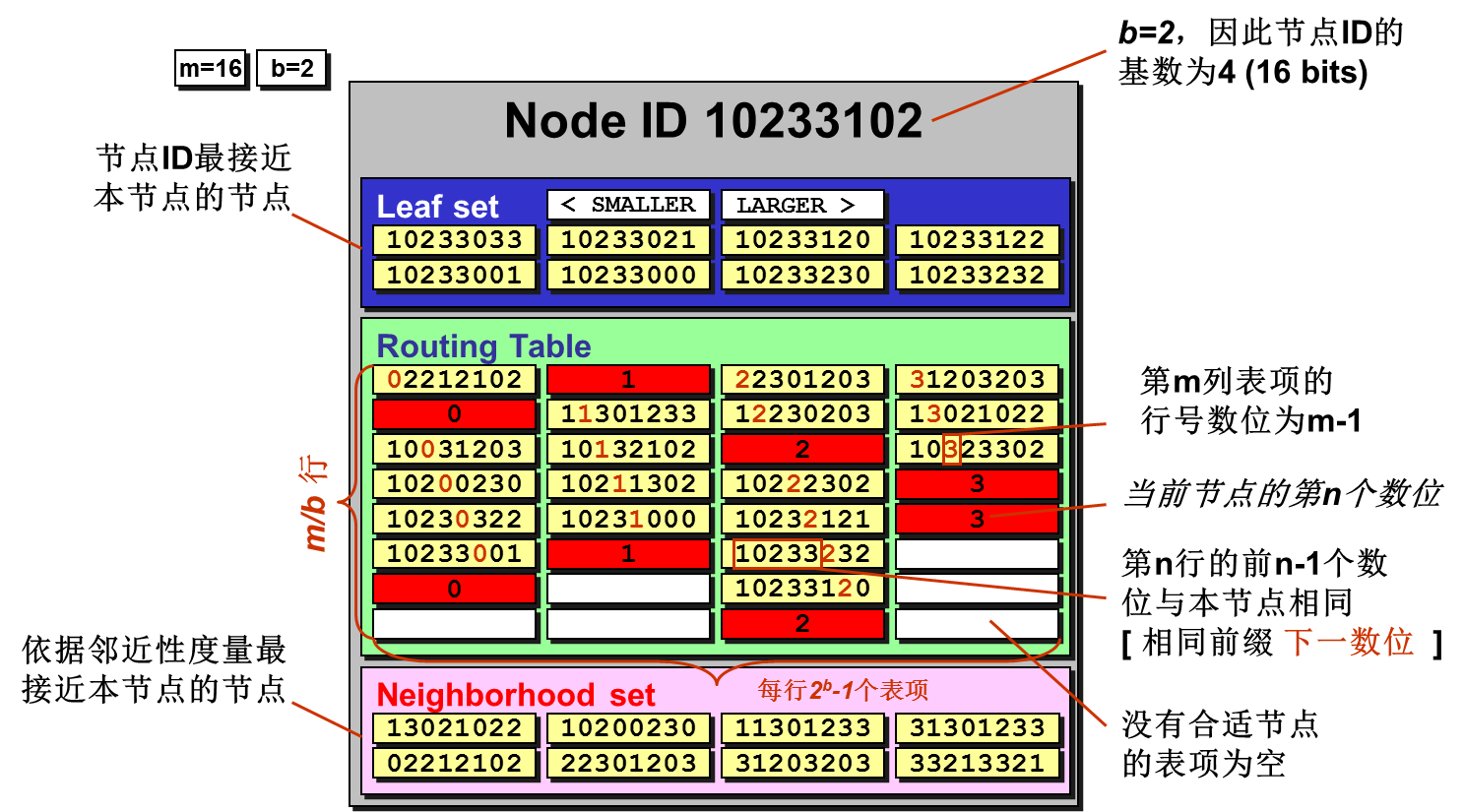
Hash表分布规则：

1. Hash算法：SHA-1
2. Hash节点IP地址－>m位节点ID(表示为NID)
3. Hash内容关键字－>m位K(表示为KID)
4. NID和KID是以2b为基的数，共有m/b个数位，b是一个配置参数
5. 节点按ID从小到大顺序排列在一个逻辑环上
6. <K, V>存储在NID与KID数值最接近的节点上



节点维护状态表：

1. 路由表
2. 邻居节点集
3. 叶子节点集



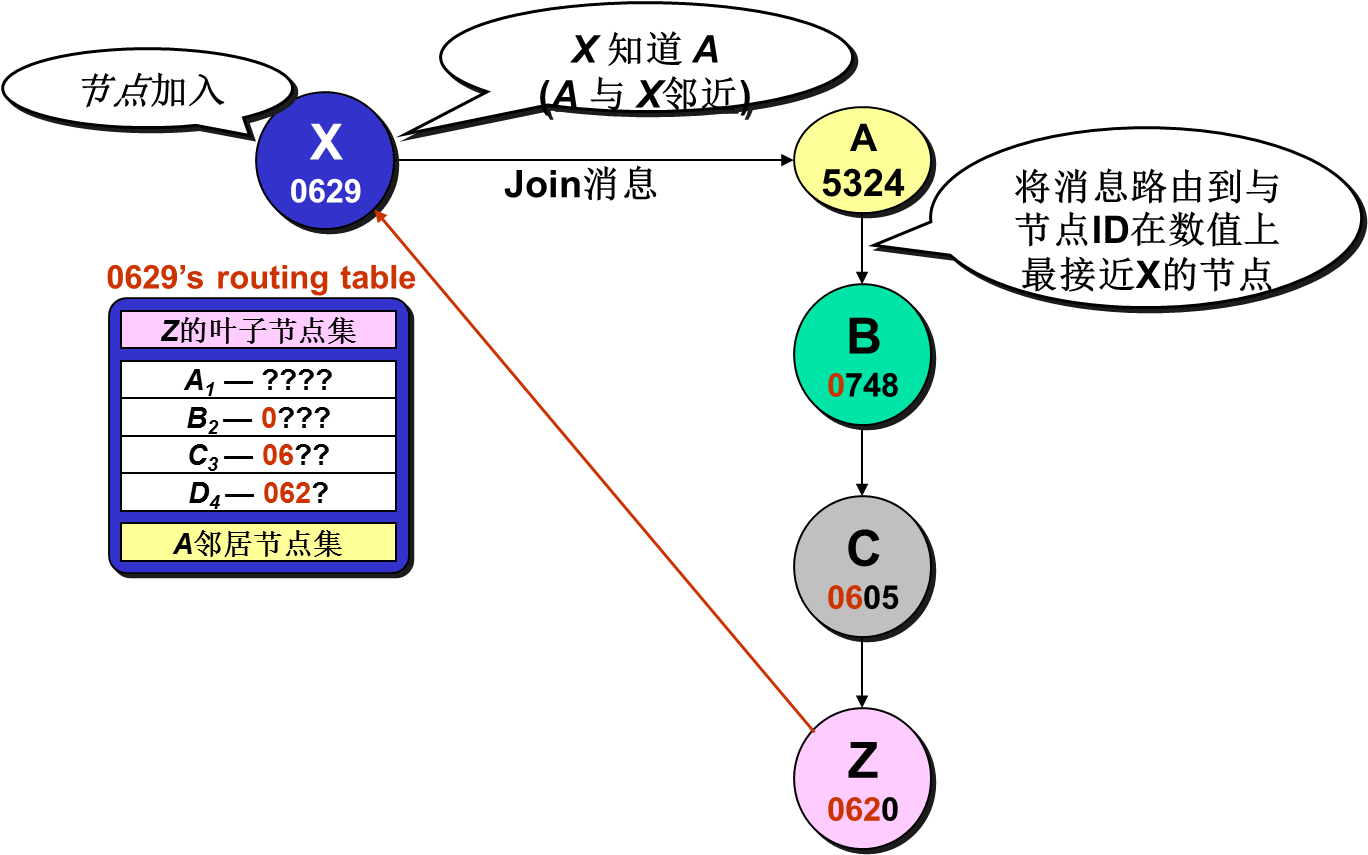
查询过程：

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，则查询消息直接被转发到目的节点，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），否则进行下一步；
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**邻居节点集合（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。

节点加入：

1. 由于A与X在邻近性度量上接近，所以使用A的邻居节点集来初始化X的邻居节点集；
2. 由于Z的节点ID与X最相近，因此使用Z的叶子节点集来初始化X的叶子节点集；
3. X将Join消息经过的第i个节点的路由表的第i行作为自己路由表的第i行，因为Join消息经过的第i个节点与X的前i个数位相同；
4. 向其他相关节点通告自己的到来：新节点向邻居节点集、叶子节点集和路由表中的每个节点发送自己的状态，以更新这些节点的状态表。



总结：

1. 逻辑网络路由跳数
2. 路由表开销；
3. 路由本地性：状态表（路由表、邻居节点集、叶子节点集）中的表项选择在邻近性度量上与本节点相近的节点
4. 稳健性：只有在个叶子节点完全失效时才会路由失败
5. **简述CDN网络的概念，DNS重定向技术。**

内容分发网络的目的通过在现有的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，提供用户访问网站的响应速度。从技术上全面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。

Akamai和Lime Light主要区别：

1. Akamai主要使用DNS重定向。Lime Light使用IP Anycast技术。
2. Akamai在全球建立部署了几千个节点，以物理节点距离用户最近而著称；通过对各节点间的路由优化，实现信息的分发；主要提供WEB、下载、流媒体等加速服务。
3. Lime Light通过在全球最重要的城市建立“超级节点”，且所有“超级节点”间采用自有光纤直连，实现全球多运营商网络的互联互通；如同在全世界建立了一个自有的、具有互联互通的“BGP”网络。Lime Light在每个“超级节点”部署了成千上万的服务器、存储、网络设备。形成了分布式的全球“信息”存储中心。通过“超级节点”网络结构的使用，Lime Light能够更好地满足和支持富媒体全球分布式分发，特别适合于为大资料库、大文件、海量观众提供高效率分发服务。

|  |  |  |  |
| --- | --- | --- | --- |
| **对比项** | | **Akamai** | **Lime Light** |
| **基础建设** | 节点数目 | 号称5000多个 | 22个 |
| 布点级别 | 二三级节点 | 超核心节点 |
| 覆盖策略 | 小节点分布覆盖 | 大节点集中覆盖 |
| 节点使用属性 | 租用 | 自建 |
| 边缘节点与源站的距离 | 较远 | 较近 |
| 边缘节点与用户的距离 | 较近 | 较远 |
| **数据分发** | 小文件分发能力 | 可实现，但受限于各节点之间网络 | 可实现 |
| 大文件分发能力 | 可基本实现，但全网分发时间较长 | 可实现，分发时间较快 |
| 分发成本 | 较高 | 低 |
| **数据存储** | 海量数据存储能力 | 可实现，但受限于存储成本，具有海量存储的节点不多 | 可实现 |
| WEB数据存储能力 | 可实现 | 可实现 |

DNS重定向：

1. 用户向域名解析器发送域名解析请求，请求解析某域名，以获取内容服务器的IP地址；
2. 域名解析请求中包含了域名解析器的IP地址，并被发送到该域名的域名服务器；
3. 该域名的域名服务器通过返回一个CNAME类的DNS记录，将此域名解析请求重定向到CDN网络的域名服务器；
4. CDN网络的域名服务器对该域名解析请求进行智能解析，根据域名解析器的IP地址、网络的拥塞状况、各CDN网络边缘节点服务器的负载情况等，将最合适的CDN网络边缘节点服务器的IP地址返回给域名解析器，域名解析器再转发给用户。
5. 用户直接访问CDN边缘节点



内容路由技术

CDN内容路由，就是为了保证它负载均衡问题。包括IP anycast 和DNS重定向两个技术。

1. **简述物联网的概念，阐述物联网五层架构。**

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。

感知层：由各种传感器以及传感器网关构成，是物联网识别物体，采集信息的来源，其主要功能是识别物体，采集信息。

接入层：用来读取识别到的物体信息，并将读到的内容连到Internet。

网络层：由各种私有网络、互联网、有线和无线通信网、网络管理系统和云计算平台等组成，负责传递和处理感知层获取的信息。

中间件层：用来管理物联网上的资源和网络通讯，连接多个独立应用程序或独立系统，使它们之间能交换信息。

应用层：是物联网和用户（包括人、组织和其他信息）的接口，它与行业需求结合，实现物联网的智能应用。

物联网案例：

牲畜溯源、精准农业、机场防入侵、智能电网、医疗保健