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。

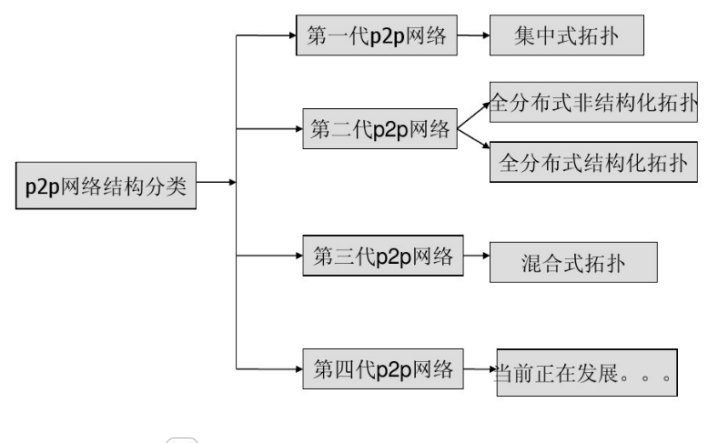
**Hadoop**是一个分布式系统基础架构。用户可以充分利用集群的威力进行高速运算和存储。Hadoop的框架核心的设计就是：HDFS和Map Reduce。HDFS为海量的数据提供了存储，则Map Reduce为海量的数据提供了计算。

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. **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：**

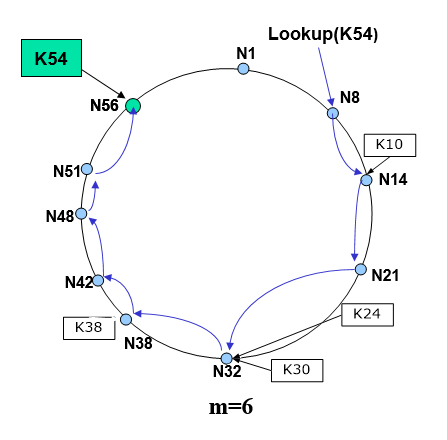
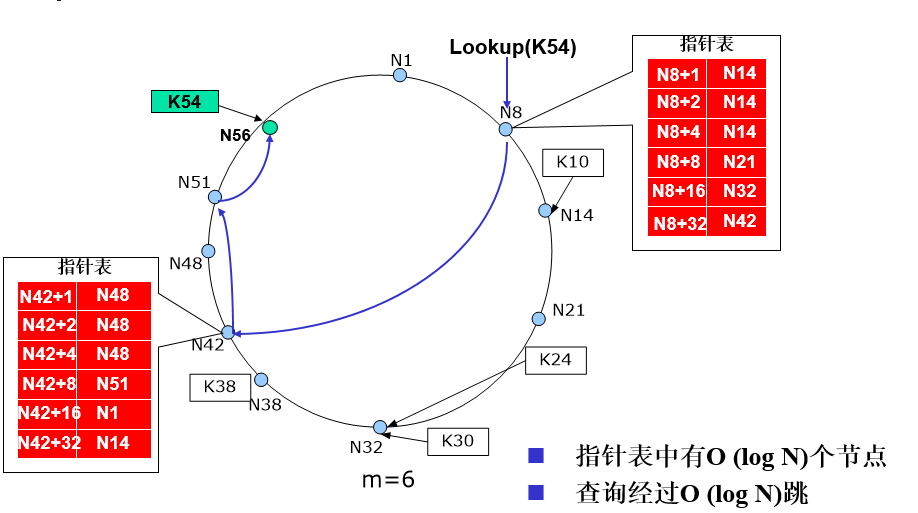
1. 采用环形拓扑(Chord环)
2. 其核心思想就是要解决在P2P应用中遇到的基本问题：如何在P2P网络中找到存有特定数据的节点
3. Chord使用一致性哈希作为哈希算法，在Chord协议中将其规定为SHA-1。

**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

改进前：改进后：

**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。

1. **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 |