**可行性报告**

1. **技术依据**

**1、小组成员背景：**

**彭昀：学过C语言、掌握linux的一些基本命令、接触过Python**

**黄奕桐：学过C语言、掌握linux的一些基本命令、接触过Python**

**张圣明：学过C语言、熟练掌握linux相关知识、有服务器学习经验**

**胡清泳：学过C语言、掌握linux的一些基本命令、接触过C++**

**杭逸哲：学过C语言、掌握linux的一些基本命令**

**2、去中心化协议Kademlia**

**Kademlia is a distributed hash table for decentralized peer-to-peer computer networks designed by Petar Maymounkov and David Mazières in 2002. It specifies the structure of the network and the exchange of information through node lookups. Kademlia nodes communicate among themselves using UDP. A virtual or overlay network is formed by the participant nodes. Each node is identified by a number or node ID. The node ID serves not only as identification, but the Kademlia algorithm uses the node ID to locate values (usually file hashes or keywords). In fact, the node ID provides a direct map to file hashes and that node stores information on where to obtain the file or resource.**

**When searching for some value, the algorithm needs to know the associated key and explores the network in several steps. Each step will find nodes that are closer to the key until the contacted node returns the value or no more closer nodes are found. This is very efficient: Like many other DHTs, Kademlia contacts only O(log(n)) nodes during the search out of a total of n nodes in the system.**

**Further advantages are found particularly in the decentralized structure, which increases the resistance against a denial-of-service attack. Even if a whole set of nodes is flooded, this will have limited effect on network availability, since the network will recover itself by knitting the network around these "holes".**

**Kademlia uses a "distance" calculation between two nodes. This distance is computed as the exclusive or (XOR) of the two node IDs, taking the result as an integer number. Keys and Node IDs have the same format and length, so distance can be calculated among them in exactly the same way. The node ID is typically a large random number that is chosen with the goal of being unique for a particular node (see UUID). It can and does happen that geographically widely separated nodes—from Germany and Australia, for instance—can be "neighbors" if they have chosen similar random node IDs.**

**Exclusive or was chosen because it acts as a distance function between all the node IDs. Specifically:**

**the distance between a node and itself is zero**

**it is symmetric: the "distances" calculated from A to B and from B to A are the same**

**it follows the triangle inequality: given A, B and C are vertices (points) of a triangle, then the distance from A to B is shorter than (or equal to) the sum of the distance from A to C and the distance from C to B.**

**These three conditions are enough to ensure that exclusive or captures all of the essential, important features of a "real" distance function, while being cheap and simple to calculate.[1]**

**Each Kademlia search iteration comes one bit closer to the target. A basic Kademlia network with 2n nodes will only take n steps (in the worst case) to find that node.**

**3、负载均衡算法**

**启发式随机采样算法**

**http://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CDFD&dbname=CDFD0911&filename=1011062707.nh&v=MTgzMTZQSVI4ZVgxTHV4WVM3RGgxVDNxVHJXTTFGckNVUkxLZlkrZHVGQ2ptVzd6S1ZGMjZIN08rSE5iTXFKRWI=**

1. **理论依据**

**去中心化：**

**DHT技术:**

**DHT全称叫分布式哈希表(Distributed Hash Table)，是一种分布式存储方法。它提供的服务类似于hash表，键值对存储在DHT中，任何参与该结构的节点能高效的由键查询到 数据值。这种键值对的匹配是分布式的分配到各个节点的，这样节点的增加与删减只会 带来比较小的系统扰动。这样也可以有效地避免“中央集权式”的服务器(比 如:tracker)的单一故障而带来的整个网络瘫痪。**

**DHT技术本质上强调以下特性:**

**离散性:构成系统的节点并没有任何中央式的协调机制。**

**伸缩性:即使有成千上万个节点，系统仍然应该十分有效率。**

**容错性:即使节点不断地加入、离开或是停止工作，系统仍然必须达到一定的可靠度。**

**其关键技术为:任一个节点只需要与系统中的部分节点(通常为O(logN)个)沟通，当成员改变的时候，只有一部分的工作(例如数据或键的发送，哈希表的改变等)必须要 完成。基本上，DHT技术就是一种映射key和节点的算法以及路由的算法。 DHT的结构:关键值空间分区(keyspace partitioning)和延展网络(overlay network)**

**关键值空间分区是指每一个节点掌管部分键空间。**

**延展网络是指一个连接各个节点的抽象网络，它能使每个节点找到拥有特定键的节点。每个节点或者存储了该键，或者储存有离这个键更近(这个距离由具体算法定义)的节点链接。**

**当这些组件都准备好后,一般使用分布式散列表来存储与读取的方式如下所述。假设关键值空间是一个 160 位长的字符串集合。为了在分布式散列表中存储一个文件,名称为 filename 且内容 data,我们计算 filename 的 SHA1 散列值(一个 160 位的关键值k)并 将消息 put(k,data)送给分布式散列表中的任意参与节点。此消息在延展网络中被转送, 直到抵达在关键值空间分区中被指定负责存储关键值 k 的节点。而 (k,data)即存储在该 节点。其他的节点只需要重新计算 filename 的散列值 k,然后提交消息 get(k)给分布式 散列表中的任意参与节点,以此来找与 k 相关的数据。此消息也会在延展网络中被转送 到负责存储 k 的节点。而此节点则会负责传回存储的数据 data。**

1. **创新点**

**传统的FastDFS由调度服务器、存储服务器和客户端三部分组成，其中调度服务器是整个系统运行的关键部分，这一部分一旦遭到攻击，将对整个系统产生很大的影响。故将FastDFS去中心化可以大大增强它的安全稳定性。然而，去中心化文件系统存在着效率不如中心化文件系统的问题，尤其是面对海量小文件存储的时候（如glusterfs）。而FastDFS本身就有着小文件存储的优势，将FastDFS改为去中心化系统再加上适当的负载均衡优化，它的性能（尤其是小文件存储性能）会高于其他的去中心化分布式文件系统。既增加文件系统的安全性，又保障文件传输效率，这就是我们这次实验的创新点。**