一致性Hash

范佚伦

Agenda

- 从Hash开始
- Distributed Hash Table
- 一致性Hash
- 一致性Hash的应用
- 其他分布式Hash算法

从Hash开始

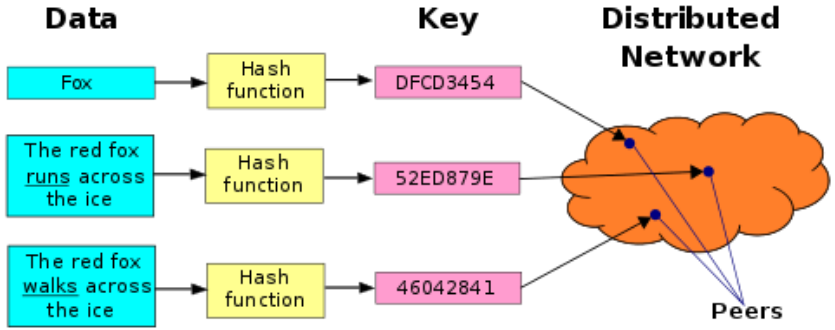
- Hash算法是将数值从大范围映射到小范围的函数
  - 是对原数据的一种有损压缩
  - 例MOD、MD5、SHA-1

- HashTable是通过hash(key)表示元素存储位置的数据结构
  - 查找速度快
  - 散列均匀

- DHT(Distributed Hash Table)是分布式环境下的HashTable
  - 一种分布式存储方式
  - 在分布式系统中提供了HashTable一样的功能

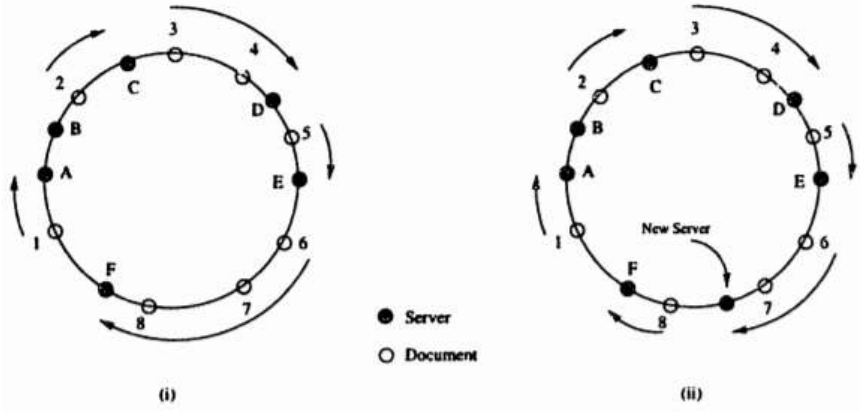
Distributed Hash Table

- 应用：缓存、分布式存储、P2P等
- 特点
  - 去中心化（Decentralization）：无中心协调机制
  - 容错性（Fault tolerance）：节点加入/退出不影响整体系统
  - 可扩展性（Scalability）：随着节点增多，系统效率不会下降
- 难点
  - 数据如何分割（Keyspace partitioning）
  - 如何快速查找（Overlay network）
- 实现
  - Kademlia（应用于BitTorrent）
  - Chord、Apache Cassandra
  - ……



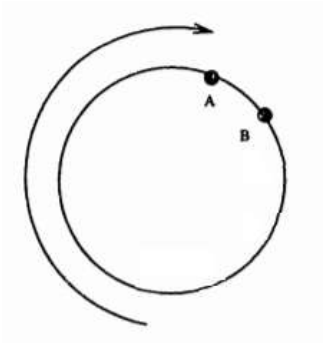
一致性Hash

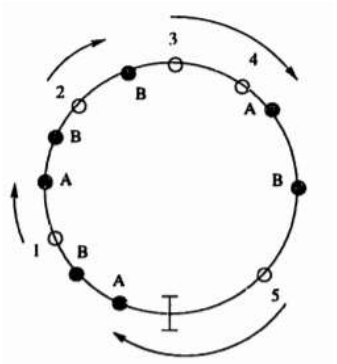
- 一种适用于分布式系统的Hash算法
  - 解决了DHT中数据如何分割的问题，但不一定要应用于DHT☺
- 起源：David Karger, 1997，解决分布式缓存热点问题
  - Karger D , Lehman E , Leighton T , et al. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web[C]// Twenty-ninth Acm Symposium on Theory of Computing. ACM, 1997.
- 一致性Hash的特点：
  - Balance, Monotonicity, Spread, Load
- 基于环的实现
  - 机器与数据同时散列在一个环中
  - 数据顺时针方向查找机器



一致性Hash

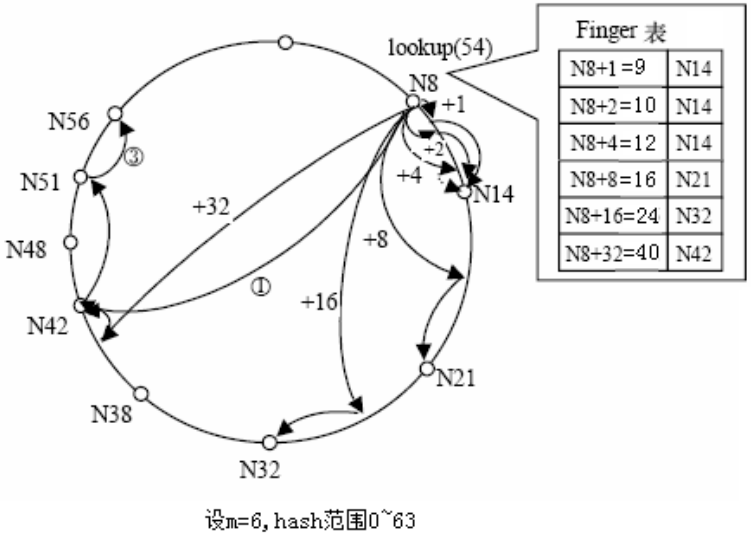
- 存在的问题
  - Hash倾斜的问题
- 优化方案
  - 虚拟节点：每一个服务节点计算多个Hash



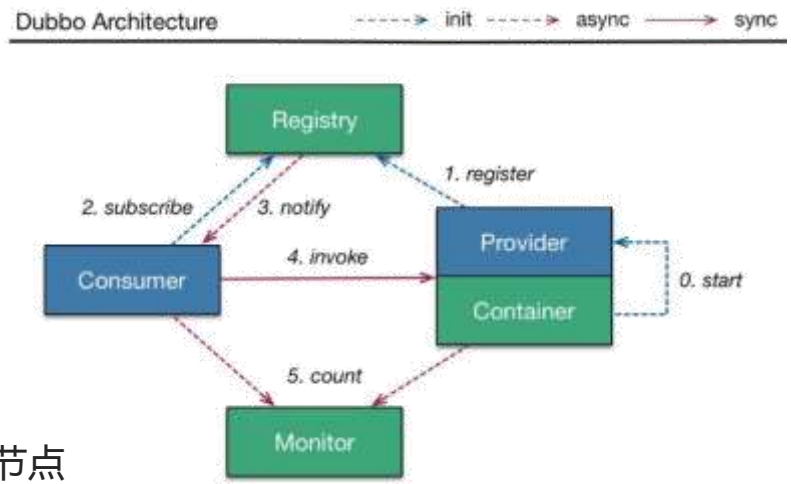


应用：Chord

- Chord是实现DHT的协议之一
  - Stoica I , Morris R , Karger D , et al. Chord:A scalable peer-to-peer lookup service for internet applications[C]
- 查找数据
  - 幂次逼近查询法 O(log(n))
- 节点加入/退出
  - 定位、通知、拷贝数据
  - Stabilization protocol
- 副本策略



应用：Dubbo



- Dubbo的负载均衡策略
  - 随机、轮询、最少活跃、一致性Hash
- Dubbo的一致性Hash策略
  - 默认对第一个参数求Hash值，160份虚拟节点
  - Consumer客户端实现一致性Hash路由
  - Hash算法采用md5
  - 虚拟节点采用TreeMap<Long, Invoker<T>>维护

其他分布式Hash算法

- Jump Consistent Hash
  - Lamping J , Veach E . A Fast, Minimal Memory, Consistent Hash Algorithm[J]. Computer Science, 2014.
  - 主要思想：
    - 每次增加节点， 只需要稳定的1/(n+1)数据跳变到新节点
  - 优点：零内存，快速，非常均匀
  - 缺点：只能在尾部增删节点

```
int32_t JumpConsistentHash(uint64_t key, int32_t num_buckets) {
    int64_t b = -1, j = 0;
    while (j < num_buckets) {
        b = j;
        key = key * 2862933555777941757ULL + 1;
        j = (b + 1) * (double(1LL << 31) / double((key >> 33) + 1));
    }
    return b;
}
```

小结

- 一致性Hash算法很好的解决了分布式系统中，服务器节点灵活伸缩带来的问题。
- DHT可以用一致性Hash算法来实现。
- 在诸多的分布式Hash算法中，一致性Hash是一个典型，但还有不少其他算法可以在不同适合的场景使用。

感谢您的观看