“幽会”的Hash

分布式hash涉及三部分，分别是键值，键值对应的数据，和管理数据的服务器。分布式hash的作用是将键值映射到管理对应数据的服务器。

简单的方法是计算键值的hash值，再根据服务器数N求模，所以对应的服务器是 s = hash(key)%N。但是这种方法的问题是，如果服务器停止服务或者添加新的服务器那么大部分数据将需要进行迁移。解决这个问题的方法是：“幽会”的Hash - rendezvous-hashing。

之所以叫rendezvous，是因为计算hash值要结合键值和服务器标识符得出结果。

举个栗子：

设有4个服务器，标识符分别是S1，S2，S3，S4。计算公式是hash(key+Sx), 将得出的hash值进行排序（降序或升序皆可），排序结果S4，S3，S2，S1。排列第一的服务器作为匹配结果。

当添加服务器或者服务器停止服务时，执行同样的计算。

rendezvous-hashing优点在于，如果某一个服务器停止服务，那么其负责的数据将在概率上随机分配给其他服务器。不会形成雪崩式事故。此外各个服务器可以设置不同的权重，以反映出服务器不同的存储和计算能力。

我们常见的一致性hash是rendezvous-hashing的特例。

[1] 关于rendezvous-hashing，https://randorithms.com/2020/12/26/rendezvous-hashing.html

[2] 原始论文，https://www.eecs.umich.edu/techreports/cse/96/CSE-TR-316-96.pdf

[3] DynamoDB论文，https://dl.acm.org/doi/10.1145/1323293.1294281

[4] 一致性hash是 rendezvous-hashing的特例，https://en.wikipedia.org/wiki/Rendezvous\_hashing#Comparison\_with\_consistent\_hashing

[5] 多个hash算法比较， https://www.snia.org/sites/default/files/SDC15\_presentations/dist\_sys/Jason\_Resch\_New\_Consistent\_Hashings\_Rev.pdf

~~但这里仍旧有缺陷，假设栗子中的服务器S4停止服务，那么S4负责的数据将全部有S3接替。这有可能造成S3服务负载短时增加，形成雪崩式事故。这里可以将每个服务器分解为虚拟的多个服务器节点，如S4分为S41，S42，S43三个服务节点。S4\*负责的三部分数据将概率上随机分配给其他三个服务器。~~