哈希分布数据的另一个缺点是，一旦某数据特征值的数据严重不均，容易出现数据倾斜（data skew）问题。例如，某系统中以用户 id 做哈希分数据，当某个用户 id 的数据量异常庞大时，该用户的数据始终由某一台服务器处理，假如该用户的数据量超过了单台服务器处理能力的上限，则该用户的数据不能被处理。更为严重的是，无论如何扩展集群规模，该用户的数据始终只能由某一台服务器处理，都无法解决这个问题。在这种情况下只能重新选择需要哈希的数据特征，例如选择用户 id 与另一个数据维度的组合作为哈希函数的输入，这样做则需要完全重新分布数据，在工程实践中可操作性不高。另一种极端的思路是，使用数据的全部而不是某些维度的特征计算哈希，这样数据将被完全打散在集群中。 然而实践中有时并不这样做，这是因为这会使得每个数据之间的关联性完全消失。如果系统处理的每条数据之间没有任何逻辑上的联系，例如一个给定关键词的查询系统，每个关键词之间并没有逻辑上的联系，则可以使用全部数据做哈希的方式解决数据倾斜问题。另外一种常见的改进算法是引入虚节点（virtual node）的概念，系统初始时就创建许多虚节点， 虚节点的个数一般远大于未来集群中机器的个数，将虚节点均匀分布到一致性哈希表上，其功能与哈希算法中的节点相同。为每个节点分配若干虚节点。操作数据时，首先通过数据的哈希值找到对应的虚节点，进而查找元数据找到对应的真实节点。使用虚节点改进有多个优点。首先，一旦某个节点不可用，该节点将使得多个虚节点不可用，从而使得多个相邻的真实节点负载失效节点的压力。同理，一旦加入一个新节点，可以分配多个虚节点，从而使得新节点可以负载多个原有节点的压力，从全局看，较容易实现扩容时的负载均衡。