consistent hash原理，优化及实现

2016-11-26 | 分布式

引言

在分布式环境中，由于数据量庞大，往往需要对数据做分区，分区有两种：一种是range分区，另一种是hash分区。顾名思义，hash分区是采用hash算法，将数据划分到不同的分区中，采用传统的hash算法能有效地将数据划分到不同的分区，但是，传统的hash算法在机器上下线时，由于hash映射的变化，会导致大量的数据发生迁移。本文以分布式缓存为场景，分析了传统hash算法的缺点，并讨论了consistent hash如何解决该问题，以及consistent hash的优化和实现。

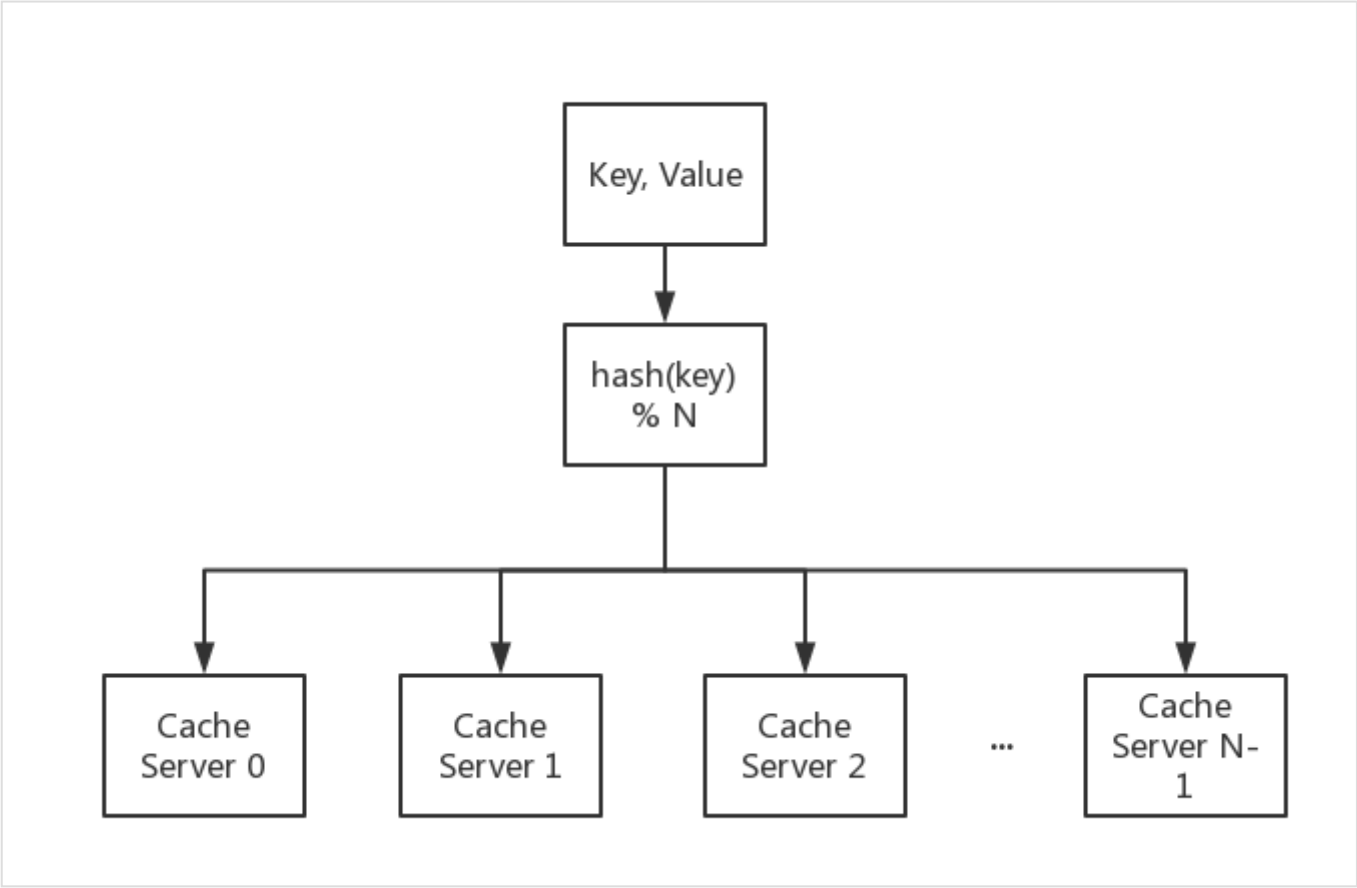
本文的按如下组织：

- 传统hash算法的缺点
- consistent hash

本文收录在我的github中papers项目，papers项目旨在学习和总结分布式系统相关的论文。

传统hash算法的缺点

以分布式缓存场景为例，如下：



对于传统hash分区方法，针对某个(key,value)对，其存储的缓存节点下标可以用hash(key) % N，在正常情况下，能良好地工作。但是，当机器宕机宕机下线时，可能会涉及到大量的数据的迁移，因为机器数量减少为N-1，对应的hash取模后，大量的(key,value)对到Cache Server的映射发生改变，导致大量的(key,value)数据在Cache Server间迁移，以一个例子说明这个问题：

假设hash函数为y = x + 1，系统中有如下(key,value)对，机器数量为5台

```
1  (0,1)
2  (1,2)
3  (2,3)
4  (3,4)
5  (4,5)
```

根据hash函数取模，得到(key,value)到Cache Server的如下映射：

```
1  (0,1) -> Cache Server 1
2  (1,2) -> Cache Server 2
3  (2,3) -> Cache Server 3
4  (3,4) -> Cache Server 4
5  (4,5) -> Cache Server 5
```

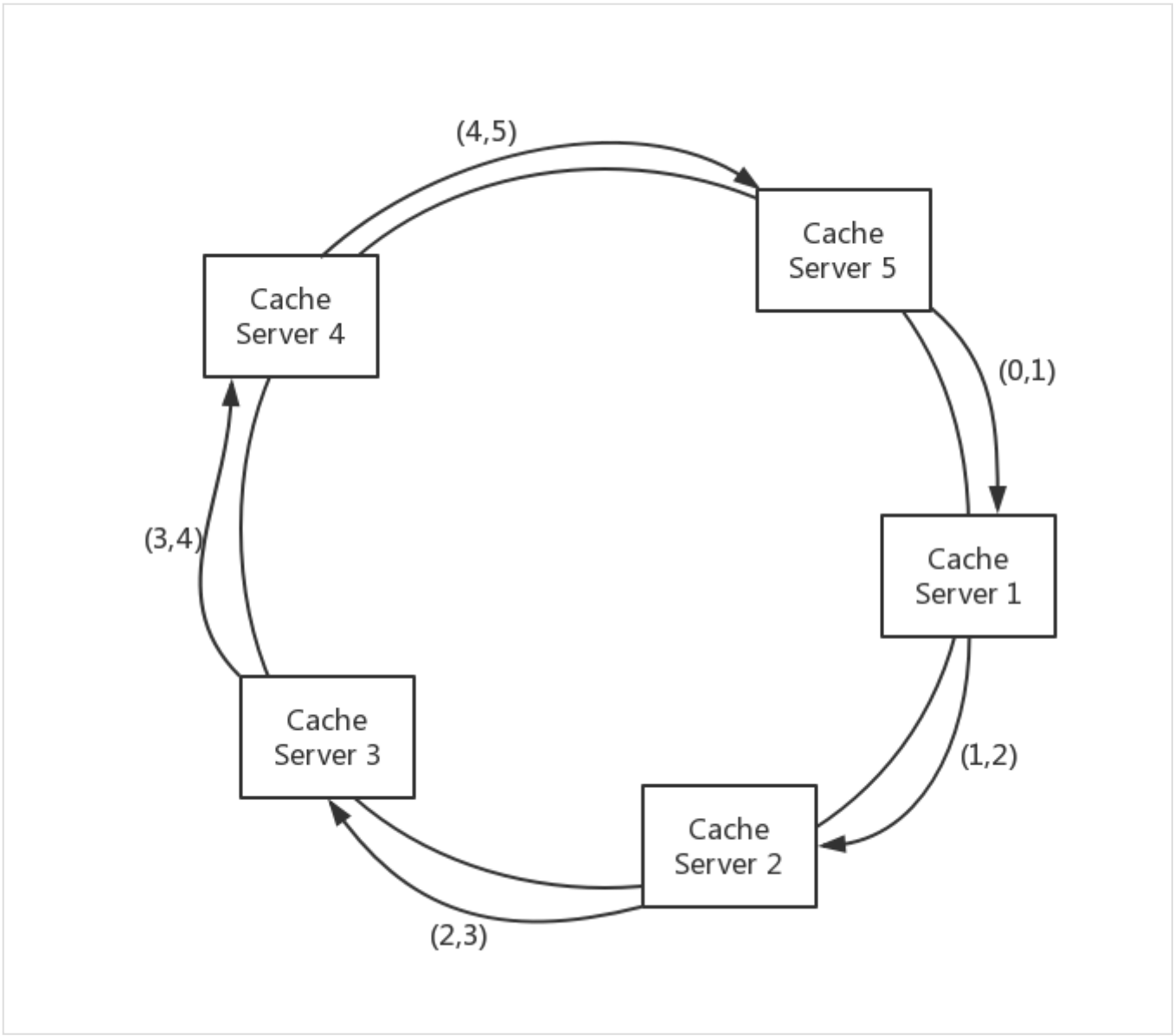
假设其中Cache Server 1宕机，(key, value)到Cache Server重新映射如下：

```
1  (0,1) -> Cache Server 2
2  (1,2) -> Cache Server 3
3  (2,3) -> Cache Server 4
4  (3,4) -> Cache Server 2
5  (4,5) -> Cache Server 4
```

可以看出，上述极端例子中，所有的(key,value)对都发生了迁移，这个开销是相当大的。在分布式系统中，由于机器数量众多，机器发生故障是常态，因此，采用传统的hash算法是不合适的。

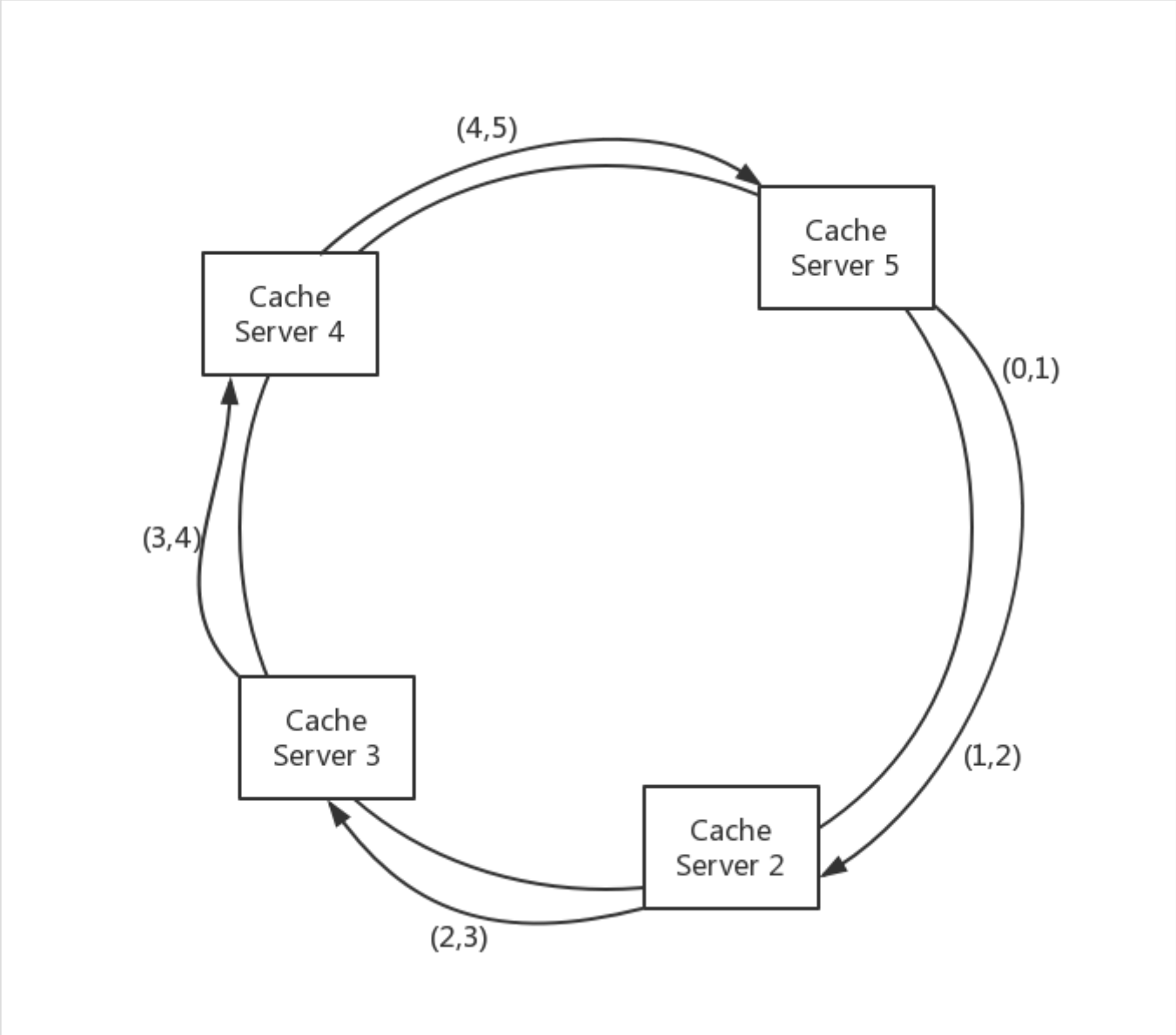
consistent hash

为了解决上述问题，引入了consistent hash算法，如下图：



在consistent hash中，有一个hash环，代表一个范围，例如，可以取值为[0,2^64-1]。对于，每个Cache Server会根据hash函数算出一个整数值，最终落到hash环的某个点上，如图中的Cache Server 1-5。每个(key,value)对存储在hash环上顺时针的下一个Cache Server，举个例子，假设hash(Cache Server i) = Hi，i=1..5，如果(1,2)的hash取值处于[H1,H2)之间的话，那么它会存储在Cache Server2上，以此类推。

因此，consistent hash能很好地应对Cache Server宕机情况，假设还是Cache Server 1宕机，如下图：



上图中Cache Server 1发生宕机的话，整个系统中只有(0,1)会从宕机的Cache Server 1迁移到Cache Server 2，比传统的hash算法会好很多。

但上述的consistent hash也有其缺点：

- Cache Server在hash环上的分布可能不均匀，导致Cache Server间的负载不均衡
- Cache Server的配置可能不同，所以能承受的负载也不同，而上述的consistent hash是没有考虑这个因素的

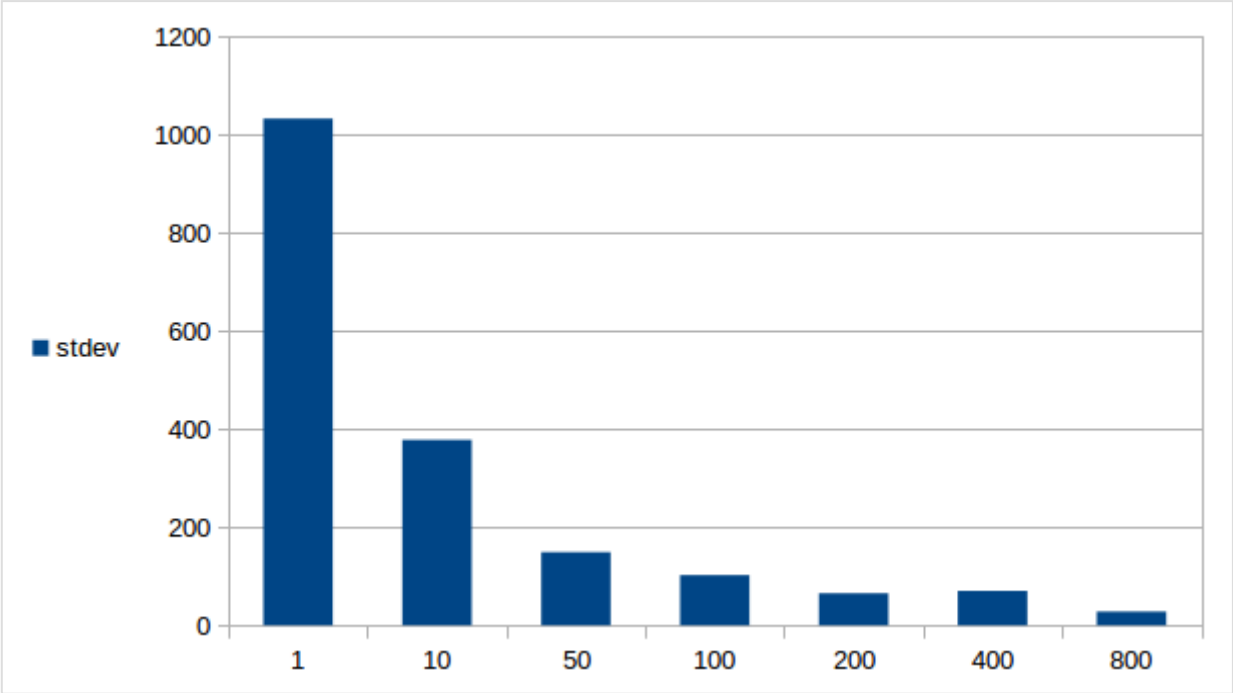
为了说明上述问题，我实现了上述的consistent hash算法，并且模拟了10000个key，10台机器的场景下各个server的负载情况如下

```
1  Node1:324
2  Node6:689
3  Node3:75
4  Node2:217
5  Node5:28
6  Node8:865
```

```
7   Node0:657
8   Node9:3157
9   Node4:2284
10  Node7:1704
```

可以看出机器之间的负载不均衡程度很高，为此，引入了虚拟节点的概念。一个实际的Server可以拆分成多个虚拟节点，虚拟节点根据hash值会散落在hash环的不同地方，这样，在虚拟节点个数较大时，负载就会趋向于均衡。

下面做了一组实验，模拟了虚拟节点的数量从1,10,50,100,200,400,800的时候，各个Cache Server的负载均衡情况：



其中横坐标是每个Server的虚拟节点的数量，纵坐标是每个Server负载量的标准偏差，反映了Server之间负载的不均衡度，从图中看出，当虚拟节点的数量增加时，Server之间的不均衡度下降了。

虚拟节点除了用来降低Server的不均衡度之外，还可以用来表示每个Server的容量情况，例如，对于负载能力为1的Server，可以给它分配1个虚拟节点，而对于负载能力为10的Server，可以给它分配10个虚拟节点，从而为异构的Server提供相应的支持。

本文的代码在consistent_hash.cpp，使用方法在ReadMe。