

文

副本更新策略 (/a/1190000004480546)

xixicat (https://segmentfault.com/u/xixicat) 5 天前发布

分布式理论系列

- 从ACID到CAP到BASE (https://segmentfault.com/a/1190000004468442)
- 2PC到3PC到Paxos到Raft到ISR (https://segmentfault.com/a/1190000004474543)
- 复制、分片和路由 (https://segmentfault.com/a/1190000004485355)
- 副本更新策略 (https://segmentfault.com/a/1190000004480546)
- 负载均衡算法及手段 (https://segmentfault.com/a/1190000004492447)

序

本文主要摘要了一些主要的副本更新策略。

1、同时更新

- 类型A：没有任何协议，可能出现多个节点执行顺序交叉导致数据不一致情况。
- 类型B：通过一致性协议唯一确定不同更新操作的执行顺序，从而保证数据一致性

2、主从式更新

多个副本之间存在一个主副本（Master Replica），其他副本为从副本，这种称为主从更新策略。所有对数据的更新首先提交到主副本，再由主副本通知从副本进行数据更新。如果同时产生多个数据更新操作，由主副本决定不同更新操作的顺序。

类型A：同步方式

主副本等待所有从副本更新完成之后才确认更新操作完成，这样确保数据的强一致性，但是会存在较大的请求延时，尤其是在多副本跨数据中心的情形下，因为请求延时取决于最慢的那个副本的更新速度。

类型B：异步方式

主副本在通知从副本更新之前即可确认更新操作。假设主副本还没有通知任何其他从副本就发生崩溃，那么数据一致性可能会出现问题，一般首先在另外的可靠存储位置将这次更新操作记录下来，以防这种情况发生。

- 1）所有读请求都通过主副本来响应，任意一个副本接收到读请求后转发为主副本，可以保证强一致，但是本来可以由距离近的副本响应的操作又得转发给距离较远的主副本，增加了请求延时，*Google的Chubby采用这种方式。*
- 2）任意一个副本都可以响应读请求，请求延时大大降低，但是可能导致读不一致，因为有些副本可能还存在旧版本的数据，*Zookeeper就是采用这种方法获得低延时，但牺牲了一致性。*

类型C：混合方式

同步混合异步，主副本首先同步更新部分从副本，然后确认更新操作完成，其他副本通关异步方式获得更新，*Kafka就是采用这种混合方式来维护数据副本的不一致性。*

- 1）读操作至少要从一个同步更新的节点读出，类似RW协议，可保证强一致性，但是请求延时加大
- 2）读操作不要求一定从至少一个同步更新节点读出，那么会出现类型B的第2种不一致性情形。

3、任意节点更新

不区分主从副本，任意节点都可以接收请求，然后又它去通知其他副本进行更新。

类型A：同步通知其他副本

存在和主从更新的类型A的情况，除此之外，为了识别出是否存在不同客户端向不同副本发送对同一数据的更新操作，还需要额外付出更多的请求延时

类型B：异步通知其他副本

存在主从更新的B方式问题。

Dynamo/Cassandra/Riak同时采取了主从式更新的类型C（同步+异步），以及任意节点更新的策略。

参考

- 大数据技术系列----副本更新策略 (http://blog.csdn.net/ni_guang2010/article/details/48493507)

5 天前发布 (/a/1190000004480546)

1 推荐

收藏

你可能感兴趣的文章

说说分布式事务(一) (<https://segmentfault.com/a/1190000004474144>) 74 浏览

kbengine开源分布式游戏服务端引擎 (<https://segmentfault.com/a/1190000000663501>) 2 收藏, 971 浏览

理解 CAP 理论 - 分布式数据库相关理论 Part1 (<https://segmentfault.com/a/1190000002802523>) 6 收藏, 980 浏览

本文采用 知识共享署名 3.0 中国大陆许可协议 (<http://creativecommons.org/licenses/by/3.0/cn>), 可自由转载、引用, 但需署名作者且注明文章出处。


讨论区

请先 [登录](#) () 后评论

本文隶属于专栏

xixicat (<https://segmentfault.com/blog/xixicat>)

spring boot , docker and so on

 xixicat (<https://segmentfault.com/u/xixicat>)
作者

关注专栏

分享扩散:

...