HBase的辅助索引

<http://wiki.apache.org/hadoop/Hbase/SecondaryIndexing>

本文是关于实现HBase辅助索引不同方法的设计文档。

使用Coprocessor实现最终一致性的辅助索引

* 基本的想法是为主表中的每个索引建立一个额外的辅助表。为列族(family)添加coprocessor，并利用coprocessor处理列族上(或指定的列)的辅助索引。用WAL(write ahead log)来保证持久性，并使用共享队列保证辅助索引的更新在调用者看来是异步的。通常HBase的时间戳可以决绝所有的解决，并且可以保证操作是幂等的（多次重复操作与一次操作等价）。

当向主表进行Put操作时，会顺序执行以下过程（假设不同的索引更新到不同的辅助表）：

1。为主表生成WALEdit

2。为辅助表的更新生成特殊的WALEdit

3。将以上两个WALEdit同步到Hlog

4。将主表的更改应用到MemStore，并提交到RWCC(MVCC,MultiVersionConsistencyControl)

5。将辅助表的更改添加到共享工作队列

6。返回给客户端

共享队列是一个线程或线程池，通过应用标准的Put操作处理辅助表的更改。

当主表故障恢复时，主表的更改可以正常回放，辅助索引的更改需要通过共享队列应用到辅助表(服务器)上。

主要的问题在于如何处理WAL和回放

由于其他RS(RegionServer)可能会失效，辅助表可能处于离线状态，这可能导致长时间等待辅助表的更新。当收回旧的HLog日志时，如何才能保证所有辅助表已经获得更新？同时，我们需要避免过度开发幂等的，不只是积极的重新应用所有的操作。

是否需要跟踪每个Hlog和它正在进行的辅助表更新操作并阻止日志回收操作，直到所有的日志都被应用？

还是使用workers将辅助表的更改写入到WAL中，

工人申请二次编辑，编辑完成（和由此耐用的其他服务器上的，所以不需要要重播如果这个失败过）写回的WAL？

或者，可以把辅助表的更改绑定到每个上memstore，只有当所有的辅助表内的数据都写入完成后，memstore才可以做flush操作，这可以与现有的语义的日志回收相匹配...但有其他方面的影响，并不会真正解决过多的回放的问题。

其他悬而未决的问题：

建立二次表（自动自举？coprocessor初始化时创建？手动？）

读操作API

未来的工作：

通过API或Shell定义索引，而不是通过编码生成的每索引一个的coprocessor

现存表的索引创建（基于表的当前数据创建索引，并保持索引的最新状态）

辅助表数据更新的同步方式（如果你想利用性能并具有较强的一​​致性的指标，适用于次要更新）

在辅助表中存储主表数据以避免单次查询中的合并和正则化

使用乐观并发控制的辅助索引

由HBase / IndexedTable实现。

目前在这里：https://github.com/hbase-trx/hbase-transactional-tableindexed

辅助索引扫描

这一次，但我不知道它生活的地方了。