phoenix-hbase

http://www.cnblogs.com/kekukekro/p/6339587.html

phoenix>CREATE TABLE "t1" (pk VARCHAR PRIMARY KEY,"cf".A VARCHAR, "cf".B VARCHAR, "cf".C VARCHAR);

hbase>put't1', 'row1', 'cf:A', 'value1'

put 't1', 'row1', 'cf:B', 'value2'

put 't1', 'row1', 'cf:C', 'value3'

1)Secondary Indexing

Phoenix支持Mutable Index和ImmutableIndex，

两种索引适合不同的场景，如Immutable Index非常适合time-seriesdata，数据一旦写入便不再修改，只能追加。

Mutable Index

phoenix>CREATE TABLE my\_table (k VARCHAR PRIMARY KEY, v1 VARCHAR, v2 BIGINT);

为列v1查询建立索引

CREATE INDEX my\_index ON my\_table (v1);

为列v1，v2查询建立索引

CREATE INDEX my\_index ON my\_table (v1) INCLUDE (v2);

Immutable Index

要使用此索引形式，必须在建表时指定IMMUTABLE\_ROWS=true，如：

CREATE TABLE my\_table (k VARCHAR PRIMARY KEY, v VARCHAR) IMMUTABLE\_ROWS=true;

注意：只有当查询cloumns完全匹配索引时，基于索引的查询才能生效。例如：

create table usertable (id varchar primary key, firstname varchar, lastname varchar);

create index idx\_name on usertable (firstname);

查询语句：

select id, firstname, lastname from usertable where firstname = 'foo';

由于索引只有firstname，而查询cloumns还包括了lastname，因此上述查询索引并未生效，可将索引调整如下：

create indexidx\_name on usertable (firstname) include(lastname);

2）Paged Queries

SELECT \* FROMTEST LIMIT 1000;

SELECT title, author, isbn, description

FROM library

WHERE published\_date > 2010

AND (title, author, isbn) > (?, ?, ?)

ORDER BY title, author, isbn

LIMIT 20

3）Sequences

创建序列

CREATE SEQUENCEmy\_sequence;

CREATE SEQUENCE my\_sequence START WITH -1000;

CREATE SEQUENCE my\_sequence INCREMENT BY 10;

CREATE SEQUENCE my\_schema.my\_sequence START 0 CACHE10;

插入数据

UPSERT INTOmy\_schema(MYKEY, MYCOLUMN) VALUES( NEXT VALUE FORmy\_schema.my\_sequence, 'foo');

删除序列

DROP SEQUENCEmy\_sequence;

DROP SEQUENCE IF EXISTS my\_schema.my\_sequence;

4）Salted Tables

在使用连续的row key时，避免RegionServer出现Hotspotting是HBase使用过程中的通用问题。详细描述及解决方案参考http://blog.sematext.com/2012/04/09/hbasewd-avoid-regionserver-hotspotting-despite-writing-records-with-sequential-keys/。

其问题大致可描述为：HBase按照row key的字典序的形式存储记录，这可以非常快速的通过raw key定位数据，以及数据的startkey和endkey范围。在很多应用场景中，很多时候采用自增或自减的序列作为HBase的row key，比如001，002，003……或者499，498，……

在这种情况下，下一个序列必须由当前序列和步长计算得出，那么如果Region由很多个RegionServer提供服务，这是没有问题的。但很显然在HBase的架构中，一个Region只有一个RegionServer提供服务。因此，当一个Region达到它预先设置的一个最大范围时，将分裂成为两个较小的Region，并由两个RegionServer接管，无论怎样分裂对Region的写入负载总是落在某一个RegionServer的主机上，若需要全局序列实现自增，显然就失去了集群负载均衡的作用，下图是一个负载情况：

Phoenix <wbr>on <wbr>HBase

找到了问题原因，那么解决方案就相对容易一下，例如可以salt row keys with a prefix，即为row key做一些添加剂，如增加前缀：

new\_row\_key = (++index % BUCKETS\_NUMBER) + original\_key

这里简单的根据数据的原始自增或自减的记录或者ID作为index，对需要分片的bucket去余进行划分，得到一个新的row key：

Phoenix <wbr>on <wbr>HBase

那么HBase集群相对之前的负载将会大有改善，如下图所示：

Phoenix <wbr>on <wbr>HBase

Phoenix中以Salted Table的方式支持上述接近方案，可以做如下定义，其中SALT\_BUCKETS是1~256的自然数：

CREATE TABLE table (a\_key VARCHAR PRIMARY KEY, a\_col VARCHAR) SALT\_BUCKETS = 20;

5）Configuration andTuning

ApachePhoenix作为HBase的一个连接驱动，在客户端做了很多定制优化。因此，应用程序可以对Client端的hbase-site.xml进行参数调优。详细属性说明参考：http://phoenix.incubator.apache.org/tuning.html

这里以bin/psql.py客户端脚本执行调优为例，修改$PHOENIX/phoenix-assembly/target目录下的phoenix-4.0.0-incubating-client.jar的hbase-default.xml文件，增加如下内容以实现将客户端默认缓存从100MB调整为1GB：

phoenix.query.maxServerCacheBytes

1024000000

hbase-site.xml中添加

<property>

<name>hbase.regionserver.wal.codec</name>

<value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</value>

</property>

<property>

<name>hbase.master.loadbalancer.class</name>

<value>org.apache.phoenix.hbase.index.balancer.IndexLoadBalancer</value>

<description>Phoenix订制的索引负载均衡器</description>

</property>

<property>

<name>hbase.coprocessor.master.classes</name>

<value>org.apache.phoenix.hbase.index.master.IndexMasterObserver</value>

<description>Phoenix订制的索引观察者</description>

</property>

<property>

<name>hbase.regionserver.wal.codec</name>

<value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</value>

<description>Enables custom WAL edits to be written, ensuring proper writing/replay of the index updates. This codec supports the usual host of WALEdit options, most notably WALEdit compression.</description>

</property>

<property>

<name>hbase.region.server.rpc.scheduler.factory.class</name>

<value>org.apache.hadoop.hbase.ipc.PhoenixRpcSchedulerFactory</value>

<description>Factory to create the Phoenix RPC Scheduler that uses separate queues for index and metadata updates</description>

</property>

<property>

<name>hbase.rpc.controllerfactory.class</name>

<value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</value>

<description>Factory to create the Phoenix RPC Scheduler that uses separate queues for index and metadata updates</description>

</property>

<property>

<name>index.builder.threads.max</name>

<value>10</value>

<description>主表更新操作建立索引的最大线程数,Default,10</description>

</property>

<property>

<name>index.builder.threads.keepalivetime</name>

<value>60</value>

<description>主表更新操作建立索引的最大线程超时时间 ,Default,60</description>

</property>

<property>

<name>index.writer.threads.max</name>

<value>10</value>

<description>将索引写到索引表的最大线程数,Default,10</description>

</property>

<property>

<name>index.writer.threads.keepalivetime</name>

<value>60</value>

<description>将索引写到索引表的最大线程超时时间,Default,60</description>

</property>

<property>

<name>hbase.htable.threads.max</name>

<value>147</value>

<description>同时最多有这么多线程往索引表写入数据 ,Default,2,147,483,647</description>

</property>

<property>

<name>hbase.htable.threads.keepalivetime</name>

<value>60</value>

<description>多线程往索引表写入数据超时时间,Default,60</description>

</property>

<property>

<name>index.tablefactory.cache.size</name>

<value>10</value>

<description>缓存10个往索引表写数据的线程,Default,10</description>

</property>