基于HBase的高效结构化数据查询方法研究

沙学府 陆保国 何锡点
(中国电子科技集团公司第二十八研究所 江苏南京 210000)

摘要:随着数据时代的到来,大数据量的即时查询技术成为研究的焦点和热点之一。虽然HBase凭借其分布式、列存储等诸多优点,目前在业界广泛使用,但由于其键值对存储的特性,不支持二级索引以及Join、Group by等数据操作,使得其在大数据量的结构化数据即时查询上存在局限性。而Hive作为分布式的关系型数据仓库,支持结构化数据的SQL查询,但它依赖底层的MapReduce计算框架,数据存取时的磁盘I/O时间消耗很大,数据量在GB级别时查询速度受限。针对以上问题,提出一个基于关系型数据库的HBase解决方案,以增加与提高HBase上结构化查询的能力。

关键词:大数据 HBase RDBMS 混合数据库 Hive
中图分类号:TP311.13 **文献标识码**:A **文章编号**:1007-9416(2015)05-0085-02

1 引言

随着数据时代的到来，Hadoop作为可在廉价PC 上部署的高可扩展性的分布式存储计算框架,成为数据时代的主流。Hadoop生态圈的其他产品,继承Hadoop框架的优势,也在各自的特长领域被业界广泛使用。在数据读写领域,传统关系型数据库在数据量较大时,查询速度明显减慢,而集群的硬件成本太高,所以,大数据量的即时查询技术成为数据时代研究的焦点和热点之一。

HBase[1],是一种面向列存储的NoSQL系统,其数据存储依赖hadoop的分布式文件系统HDFS。HBase能够在普通硬件设备上存储数以百亿行的大数据量表，目前在企业中广泛使用。譬如Facebook(消息,大数据实时数据分析),Trend Micro, Adobe, Twiter, Yahoo!与淘宝。虽然HBase能够存储数百TB海量数据,具有高并发、支持随机访问、高可扩展性的优点[2],并支持结构化和半结构化数据存储,但是HBase使用键值对的数据模型,不支持sql查询,不支持二级索引,不支持如Join、跨行、跨表事物、Order by与Group by。在大量结构化数据的多表查询上,优势受限。相比之下,Hadoop生态圈的Hive可以支持传统关系型数据库的主流SQL语言,但是Hive是建立在Hadoop基础之上的分布式数据仓库,其查询依赖于MapReduce计算框架,数据存取时的磁盘I/O时间消耗很大,数据量在GB级别时查询速度也不快。因此,本文结合关系型数据库在处理结构化数据上的SQL优势,设计并构建了一个基于关系型数据库的HBase解决方案,以增加与提高HBase上结构化查询的能力。

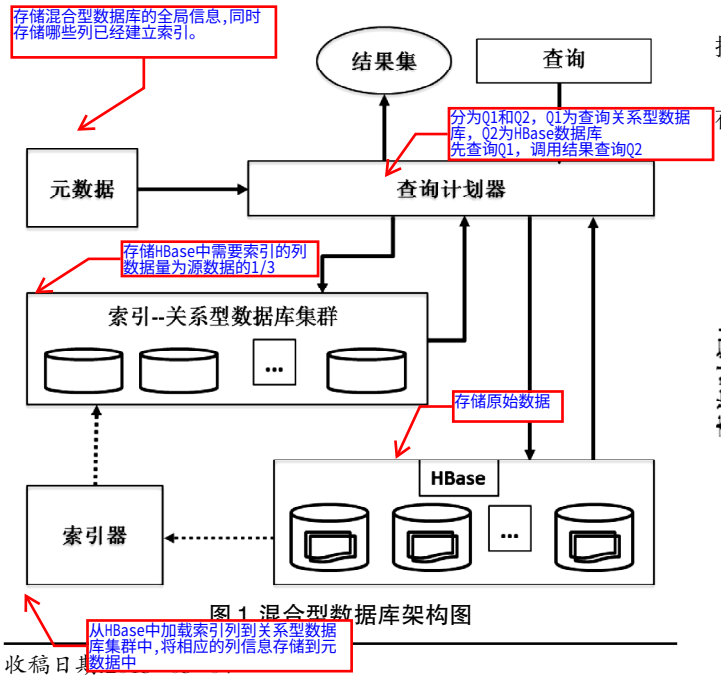

图1 混合型数据库架构图

2 体系架构设计

本文构建的基于关系型数据库的HBase(以下简称“混合型数据库”)架构如图1所示。

其中,关系型数据库作为提高HBase结构化查询能力的索引,数据本身存储在HBase中。需要建立索引的数据存储在关系型数据库中。数据库中的每条记录都包含一个指向HBase的“指针”。关系型数据库返回查询结果集,查询计划器根据“指针”查询HBase,根据查询结果集和HBase查询结果进行合并,返回给用户。

本文中的混合型数据库可支持:(1)丰富SQL查询语法:支持更复杂的结构化查询;(2)提高HBase上结构化查询的性能表现,如Join, Order by, Group by;(3)减少硬编码,提高生产率;(4)高可扩展性;(5)通用解决方案。

3 混合型数据库技术方案

本文的混合型数据库技术方案包括HBase数据存储、关系型数据库集群、索引器、元数据、查询计划器五大部分的设计。

(1)HBase数据存储:用来存储原始数据。(2)关系型数据库集群:存储HBase中需要索引的列。实际上,索引只有源数据数据量的约

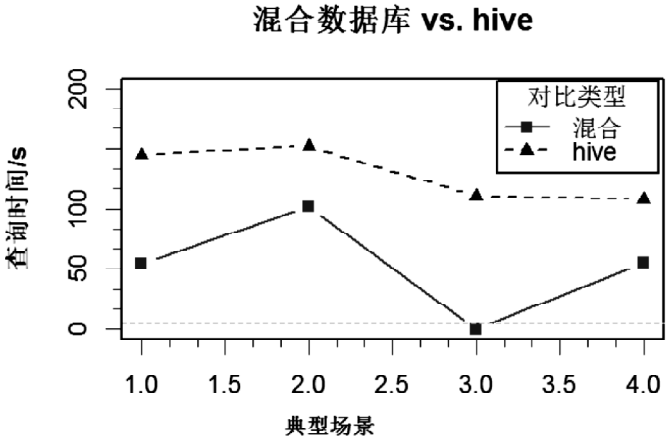

图2 混合型数据库架构的查询结果

收稿日期:
作者简介:沙学府(1987—),男,安徽无为人,硕士,毕业于哈尔滨工业大学,研究方向:大数据。

表 1 vehicle

Id	Bigint，primary key
vehicle_num	Varchar(50)
vehicle_type	Varchar(24)
fleet_name	Varchar(100)
group_id	smallint
drivername	Varchar(50)
Mobile	Varchar(50)
Remark	Varchar(250)
Dummy	int

表 2 gps_location_log

Id	Bigint，primary key
vehicle_num	Varchar(50)
longitude	Double
latitude	double
location	smallint
gps_time	Varchar(50)

表 3

	Hive 机构	混合型数据库架构	Hbase 架构
单张大表执行简单查询（不使用 Group by，Order by 操作）	108s	55.2s	7.8s

1/3，在大多数场景下，能够被存储在一个单节点的数据库中，为了使方案更通用，本文采用如DB2 DPF的集群提供以实现解决方案的高可扩展性。(3)索引器：从HBase中加载索引列到关系型数据库集群中，将相应的列信息存储到元数据中，类似于从HBase加载数据到关系型数据库中的导入工具。(4)元数据：存储混合型数据库的全局信息，同时存储哪些列已经建立索引。(5)查询计划器。查询计划的具体过程如下：

一、接受查询请求，根据元数据重写查询为Q1和Q2两部分。Q1为关系型数据库集群中的查询，Q2为HBase中的查询语句；二、将Q1发送到关系型数据库，接受查询结果，包括HBase的rowkey信息；三、根据rowkey信息，在HBase中执行Q2，以提高性能，每次查询固定数量的rowkeys到HBase中，并获得结果集；四、与Q1中的查询结果集进行合并。

由于查询被重写为Q1和Q2两部分，所以混合型数据库的查询时间为：

QueryCost(total) = QueryCost(Q1) + QueryCost(Q2).

4 混合型数据库查询和维护

4.1 混合型数据库查询

针对不同的查询场景，查询路径是不相同的，查询计划器会根据查询开销进行选择。具体场景分为以下三种。

(1)场景1：查询完全在关系型数据库中执行。查询语句实例如下：

Select T1.a, sum(T1.b) from T1 where T1.b > 100 group by T1.a;

Select T1.a from T1,T2 where T1.a=T2.a and T1.b like '%ss'.

(2)场景2：查询部分在关系型数据库中，部分在HBase中。查询语句实例如下：

Select T1.* from T1,T2 where T1.a = T2.a .

(3)场景3：查询完全在HBase中进行，即全表扫描。

4.2 索引维护

本文索引维护提供两中IUD操作模型的解决方案：

(1)索引直接写入：将数据直接先写入HBase，再更新关系型数据库，如果任何一步失败，进行回滚；(2)索引延迟写入：将数据写入HBase并记录，到固定时间窗口或数据达到阀值，将索引数据写入关系型数据库。这需要用户允许数据的延迟更新。

5 实验环境与数据

该实验使用1台DB2 DPF v97fp5。使用14个Hadoop节点，Hadoop版本为0.20.2。使用14个HBase节点，HBase版本为0.90.3。使用14个Hive节点，Hive版本为0.70.1。实验数据有9.8GB，5000w行记录。实验场景有4个，主要对比Hive on hadoop 架构、本文基于关系型数据库的HBase架构，以及不使用join，group by，order by等操作时候的单表HBase查询。实验使用vehicle和gps_location_log两张数据表，具体结构如表1—表2所示：

基于Hive架构和本文混合型数据库架构的查询结果如图2所示。

具体四个对比场景如下：(1)场景1：小表join大表；(2)场景2：大表join大表；(3)场景3：大表执行Group by查询；(4)场景4：单张大表执行简单查询(不使用Group by，Order by操作)。

其中，HBase架构本身不支持场景1—3，但是支持场景4的对比，所以在场景4中还加入了HBase单独使用时候的查询速度。具体见下表3。

分析实验结果可知，HBase本身可以支持的大数据量单表简单查询，其查询速度在三者中最快；但是，在对大数据量的结构化数据做HBase本身不支持的Join、Group by等复杂操作时，本文提出的混合型数据库查询速度较快。

参考文献

[1]http://hbase.apache.org/.
[2]http://www.searchtb.com/2011/01/understanding-hbase.html.