Apache Kylin 框架介绍

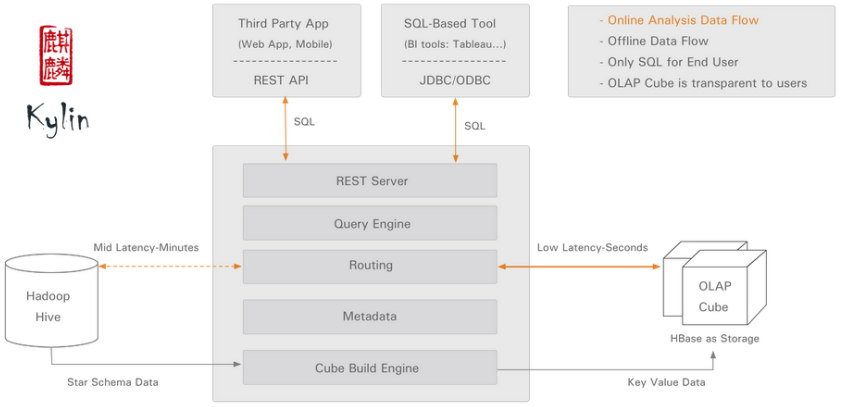
作者 ZanderXu (/u/8869887cd4c6) + 关注
2017.01.16 16:14 字数 3393 阅读 12 评论 0 喜欢 1
(/u/8869887cd4c6)

1. Apache Kylin 是什么？

Apache Kylin™是一个开源的分布式分析引擎，提供Hadoop之上的SQL查询接口及多维分析（OLAP）能力以支持超大规模数据，最初由eBay Inc. 开发并贡献至开源社区。它能在亚秒内查询巨大的Hive表。

2. Apache Kylin框架介绍

Apache kylin 能提供低延迟（sub-second latency）的秘诀就是预计算，即针对一个星型拓扑结构的数据立方体，预计算多个维度组合的度量，然后将结果保存在hbase中，对外暴露JDBC、ODBC、Rest API的查询接口，即可实现实时查询。



Kylin 框架

如上图所示，Kylin从Hadoop Hive中获取数据，然后经过Cube Build Engine，将Hive中的数据Build成一个OLAP Cube保存在HBase中。用户执行SQL查询时，通过Query引擎，将SQL语句解析成OLAP Cube查询，然后将结果返回给用户。

3. Apache Kylin核心概念

1.表(table)：This is definition of hive tables as source of cubes，在build cube 之前，必须同步在 kylin中。
2.模型(model)：模型描述了一个星型模式的数据结构，它定义了一个事实表（Fact Table）和多个查找表（Lookup Table）的连接和过滤关系。
3. Cube 描述：描述一个Cube实例的定义和配置选项，包括使用了哪个数据模型、包含哪些维度和度量、如何将数据进行分区、如何处理自动合并等等。
4.Cube实例：通过Cube描述Build得到，包含一个或者多个Cube Segment。
5.分区**(Partition)：**用户可以在Cube描述中使用一个DATA/STRING的列作为分区的列，从而将一个Cube按照日期分割成多个segment。
6.立方体段**(cube segmetn)：**它是立方体构建（build）后的数据载体，一个 segment 映射hbase中的一张表，立方体实例构建（build）后，会产生一个新的segment，一旦某个已经构建的立方体的原始数据发生变化，只需刷新（fresh）变化的时间段所关联的segment即可。
7.聚合组：每一个聚合组是一个维度的子集，在内部通过组合构建cuboid。

8.作业(job)：对立方体实例发出构建（build）请求后，会产生一个作业。该作业记录了立方体实例build时的每一步任务信息。作业的状态信息反映构建立方体实例的结果信息。如作业执行的状态信息为RUNNING 时，表明立方体实例正在被构建；若作业状态信息为FINISHED ，表明立方体实例构建成功；若作业状态信息为ERROR ，表明立方体实例构建失败！

3.1 DIMENSION & MEASURE的种类

- Mandotary：强制维度，所有cuboid必须包含的维度。
- Hierarchy：层次关系维度，维度之间具有层次关系性，只需要保留一定层次关系的cuboid即可。
- Derived：衍生维度，在lookup 表中，有一些维度可以通过它的主键衍生得到，所以这些维度将不参加cuboid的构建。
- Count Distinct(HyperLogLog)：直接进行count distinct是很难去计算的，一个近似的算法HyperLogLog可以保持错误率在一个很低的范围内。
- Count Distinct(Precise)：将基于RoaringBitMap进行计算，目前只支持int和BigInt。

3.2 Cube Action种类

- BUILD：给定一个分区列指定的时间间隔，对Cube进行Build，创建一个新的cube Segment。
- REFRESH：这个操作，将在一些分期周期内对cube Segment进行重新build。
- MERGE：这个操作将合并多个cube segments。这个操作可以在构建cube时，设置为自动完成。
- PURGE：清理一个Cube实例下的segment，但是不会删除HBase表中的Tables。

3.3 Job状态

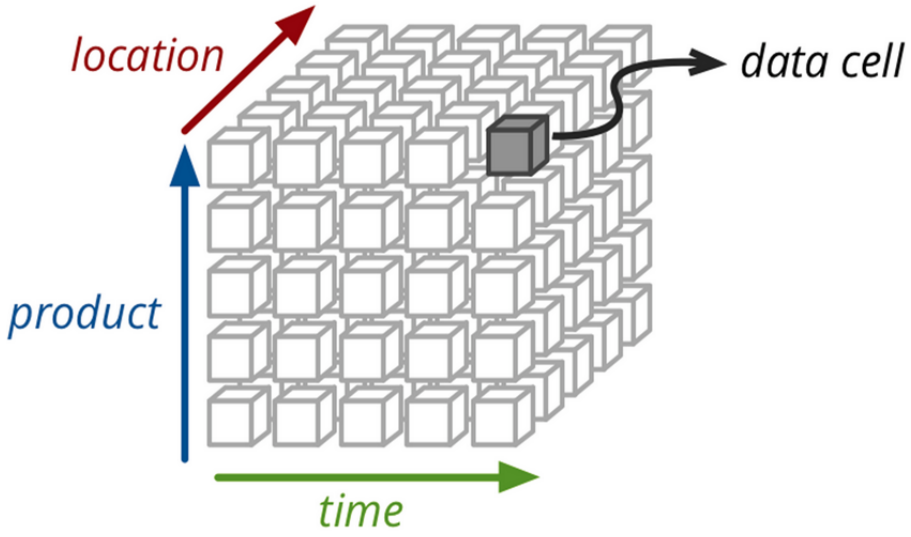
NEW：表示一个job已经被创建。 PENDING：表示一个job已经被job Scheduler提交，等待执行资源。
RUNNING：表示一个job正在运行。 FINISHED：表示一个job成功完成。
ERROR：表示一个job因为错误退出。 DISCARDED：表示一个job被用户取消。

3.4 Job执行

RESUME：这个操作将从失败的Job的最后一个成功点继续执行该Job。 DISCARD：无论工作的状态,用户可以结束它和释放资源。

4. Apache Kylin Cube 的构建过程

4.1 Cube的物理模型



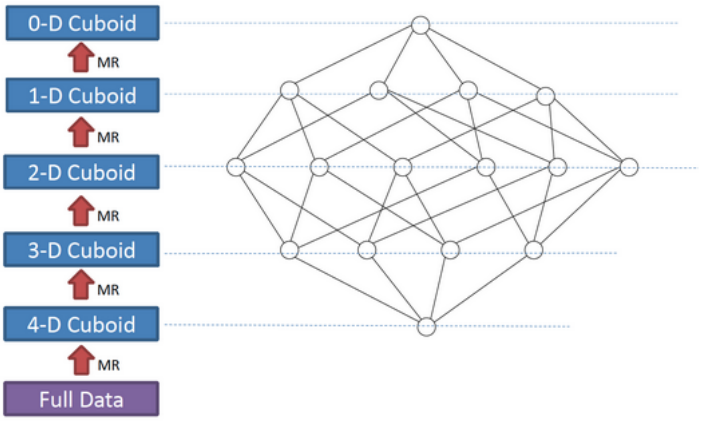
Cube物理模型

如上图所示，一个常用的3维立方体，包含：时间、地点、产品。假如data cell 中存放的是产量，则我们可以根据时间、地点、产品来确定产量，同时也可以根据时间、地点来确定所有产品的总产量等。
Apache Kylin就将所有（时间、地点、产品）的各种组合实现算出来，data cell 中存放度量，其中每一种组合都称为cuboid。估n维的数据最多有2^n个cuboid，不过Kylin通过设定维度的种类，可以减少cuboid的数目。

4.2 Cube构建算法介绍

4.2.1 逐层算法(Layer Cubing)

我们知道，一个N维的Cube，是由1个N维子立方体、N个(N-1)维子立方体、N*(N-1)/2个(N-2)维子立方体、.....、N个1维子立方体和1个0维子立方体构成，总共有2^N个子立方体组成，在逐层算法中，按维度数逐层减少来计算，每个层级的计算（除了第一层，它是从原始数据聚合而来），是基于它上一层级的结果来计算的。
比如，[Group by A, B]的结果，可以基于[Group by A, B, C]的结果，通过去掉C后聚合得来的；这样可以减少重复计算；当 0维度Cuboid计算出来的时候，整个Cube的计算也就完成了。



逐层算法

如上图所示，展示了一个4维的Cube构建过程。
此算法的Mapper和Reducer都比较简单。Mapper以上一层Cuboid的结果（Key-Value对）作为输入。由于Key是由各维度值拼接在一起，从其中找出要聚合的维度，去掉它的值成新的Key，并对Value进行操作，然后把新Key和Value输出，进而Hadoop MapReduce对所有新Key进行排序、洗牌（shuffle）、再送到Reducer处；Reducer的输

入会是一组有相同Key的Value集合，对这些Value做聚合计算，再结合Key输出就完成了一轮计算。
每一轮的计算都是一个MapReduce任务，且串行执行； 一个N维的Cube，至少需要N次MapReduce Job。

算法优点

- 此算法充分利用了MapReduce的能力，处理了中间复杂的排序和洗牌工作，故而算法代码清晰简单，易于维护；
- 受益于Hadoop的日趋成熟，此算法对集群要求低，运行稳定；在内部维护Kylin的过程中，很少遇到在这几步出错的情况；即便是在Hadoop集群比较繁忙的时候，任务也能完成。

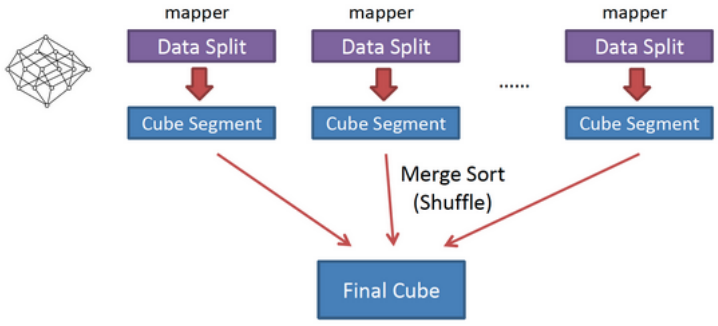
算法缺点

- 当Cube有比较多维度的时候，所需要的MapReduce任务也相应增加；由于Hadoop的任务调度需要耗费额外资源，特别是集群较庞大的时候，反复递交任务造成的额外开销会相当可观；
- 由于Mapper不做预聚合，此算法会对Hadoop MapReduce输出较多数据; 虽然已经使用了Combiner来减少从Mapper端到Reducer端的数据传输，所有数据依然需要通过Hadoop MapReduce来排序和组合才能被聚合，无形之中增加了集群的压力;
- 对HDFS的读写操作较多：由于每一层计算的输出会用做下一层计算的输入，这些Key-Value需要写到HDFS上；当所有计算都完成后，Kylin还需要额外的一轮任务将这些文件转成HBase的HFile格式，以导入到HBase中去；
- 总体而言，该算法的效率较低，尤其是当Cube维度数较大的时候；时常有用户问，是否能改进Cube算法，缩短时间。

4.2.2 快速Cube算法(Fast Cubing)

快速Cube算法（Fast Cubing）是麒麟团队对新算法的一个统称，它还被称作“逐段”(By Segment) 或“逐块”(By Split) 算法。

该算法的主要思想是，对Mapper所分配的数据块，将它计算成一个完整的小Cube 段（包含所有Cuboid）；每个Mapper将计算完的Cube段输出给Reducer做合并，生成大Cube，也就是最终结果；图2解释了此流程。



快速Cube算法

与旧算法相比，快速算法主要有两点不同

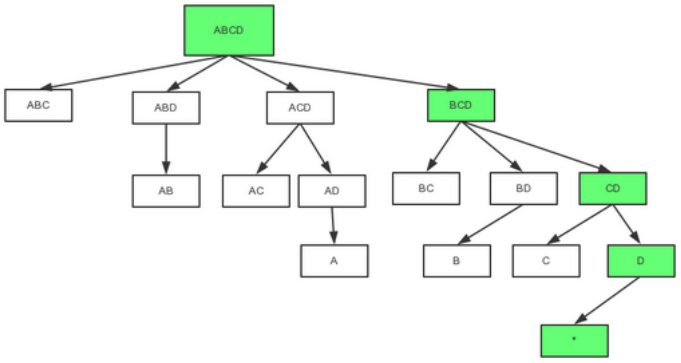
- Mapper会利用内存做预聚合，算出所有组合；Mapper输出的每个Key都是不同的，这样会减少输出到Hadoop MapReduce的数据量，Combiner也不再需要；
- 一轮MapReduce便会完成所有层次的计算，减少Hadoop任务的调配。

子立方体生成树的遍历

值得一提的还有一个改动，就是子立方体生成树(Cuboid Spanning Tree)的遍历次序；在旧算法中，Kylin按照层级，也就是广度优先遍历(Broad First Search)的次序计算出各个Cuboid；在快速Cube算法中，Mapper会按深度优先遍历（Depth First Search）来计算各个Cuboid。深度优先遍历是一个递归方法，将父Cuboid压栈以计算子Cuboid，直到没有子Cuboid需要计算时才出栈并输出给Hadoop；最多需要暂存N个Cuboid，N是Cube维度数。

采用DFS，是为了兼顾CPU和内存：

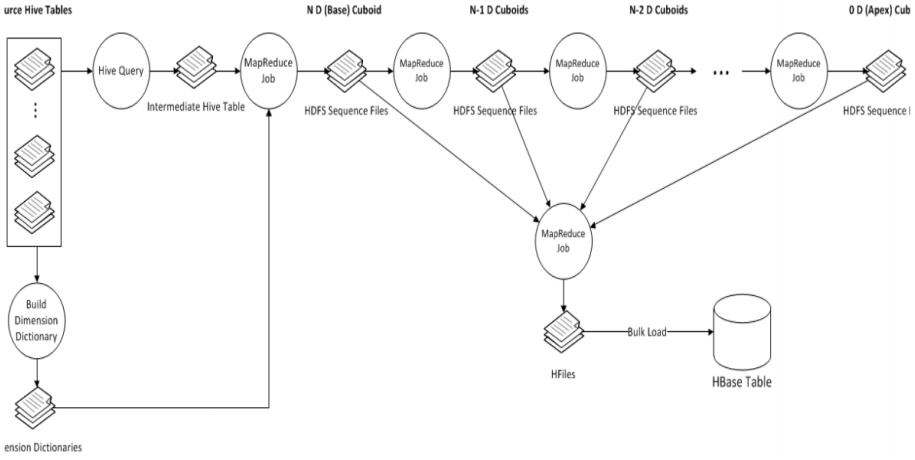
- 从父Cuboid计算子Cuboid，避免重复计算；
- 只压栈当前计算的Cuboid的父Cuboid，减少内存占用。



立方体生成数的遍历过程

上图是一个四维Cube的完整生成树；按照DFS的次序，在0维Cuboid 输出前的计算次序是 ABCD -> BCD -> CD -> D -> ， *ABCD, BCD, CD和D需要被暂存；在*被输出后，D可被输出，内存得到释放；在C被计算并输出后，CD就可以被输出； ABCD最后被输出。

4.3 Cube构建流程



Cube构建流程

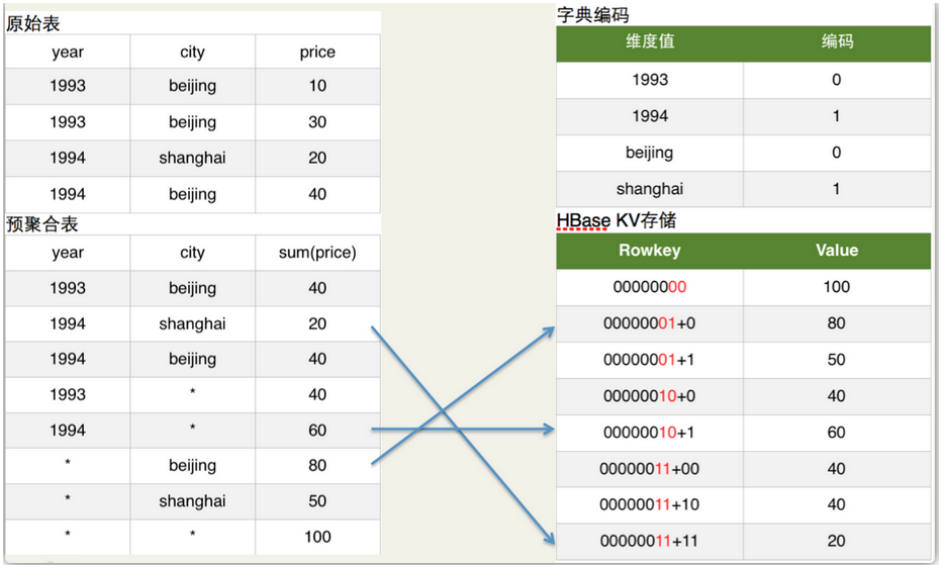
主要步骤如下：

1. 构建一个中间平表(Hive Table)：将Model中的fact表和look up表构建成一个大的Flat Hive Table。
2. 重新分配Flat Hive Tables。
3. 从事实表中抽取维度的Distinct值。
4. 对所有维度表进行压缩编码，生成维度字典。
5. 计算和统计所有的维度组合，并保存，其中，每一种维度组合，称为一个Cuboid。

6. 创建HTable。
7. 构建最基础的Cuboid数据。
8. 利用算法构建N维到0维的Cuboid数据。
9. 构建Cube。
10. 将Cuboid数据转换成HFile。
11. 将HFile直接加载到HBase Table中。
12. 更新Cube信息。
13. 清理Hive。

5. Apache Kylin Cube 的存储

简单的说Cuboid的维度会映射为HBase的Rowkey，Cuboid的指标会映射为HBase的Value。



原始表

year	city	price
1993	beijing	10
1993	beijing	30
1994	shanghai	20
1994	beijing	40

预聚合表

year	city	sum(price)
1993	beijing	40
1994	shanghai	20
1994	beijing	40
1993	*	40
1994	*	60
*	beijing	80
*	shanghai	50
*	*	100

字典编码

维度值	编码
1993	0
1994	1
beijing	0
shanghai	1

HBase KV存储

Rowkey	Value
00000000	100
00000001+0	80
00000001+1	50
00000010+0	40
00000010+1	60
00000011+00	40
00000011+10	40
00000011+11	20

Cube映射成HBase存储

如上图原始表所示：Hive表有两个维度列year和city，有一个指标列price。如上图预聚合表所示：我们具体要计算的是year和city这两个维度所有维度组合（即4个cuboid）下的sum(priece)指标，这个指标的具体计算过程就是由MapReduce完成的。
如上图字典编码所示：为了节省存储资源，Kylin对维度值进行了字典编码。图中将beijing和shanghai依次编码为0和1。

如上图HBase KV存储所示：在计算cuboid过程中，会将Hive表的数据转化为HBase的KV形式。Rowkey的具体格式是cuboid id + 具体的维度值（最新的Rowkey中为了并发查询还加入了ShardKey），以预聚合表内容的第2行为例，其维度组合是（year，city），所以cuboid id就是00000011，cuboid是8位，具体维度值是1994和shanghai，所以编码后的维度值对应上图的字典编码也是11，所以HBase的Rowkey就是0000001111，对应的HBase Value就是sum(priece)的具体值

6. Apache Kylin 如何将SQL转换成HBase Scan查询

还是以上面的例子进行解释，假
设查询SQL如下：

```
select year, sum(price)
from table
where city = "beijing"
group by year
```

SQL 转换成 SCAN

这个SQL涉及维度year和city，所以其对应的cuboid是00000011，又因为city的值是确定的beijing,所以在Scan HBase时就会Scan Rowkey以00000011开头且city的值是beijing的行，取到对应指标sum(price)的值，返回给用户。

Apache Kylin (/nb/9110262)

举报文章 © 著作权归作者所有

ZanderXu (/u/8869887cd4c6)
写了 6455 字，被 0 人关注，获得了 1 个喜欢
(/u/8869887cd4c6)

+关注

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持

♡ 喜欢 (/sign_in) | 1

更多分享
(http://cwb.assets.jianshu.io/notes/images/848875(

登录 (/sign_in)后发表评论

评论

智慧如你，不想发表一点想法 (/sign_in)咩~