CSDN 云计算频道 为IT技术人员提供有关云计算的最新资讯和技术!

Facebook 数据仓库揭秘：RCFile 高效存储结构

本文介绍了 Facebook 公司数据分析系统中的 RCFile 存储结构，该结构集行存储和列存储的优点于一身，在 MapReduce 环境下的大规模数据分析中扮演重要角色。

Facebook 曾在 2010 ICDE（IEEE International Conference on Data Engineering）会议上介绍了数据仓库 Hive。Hive 存储海量数据在 Hadoop 系统中，提供了一套类数据库的数据存储和处理机制。它采用类 SQL 语言对数据进行自动化管理和处理，经过语句解析和转换，最终生成基于 Hadoop 的 MapReduce 任务，通过执行这些任务完成数据处理。图 1 显 示了 Hive 数据仓库的系统结构。

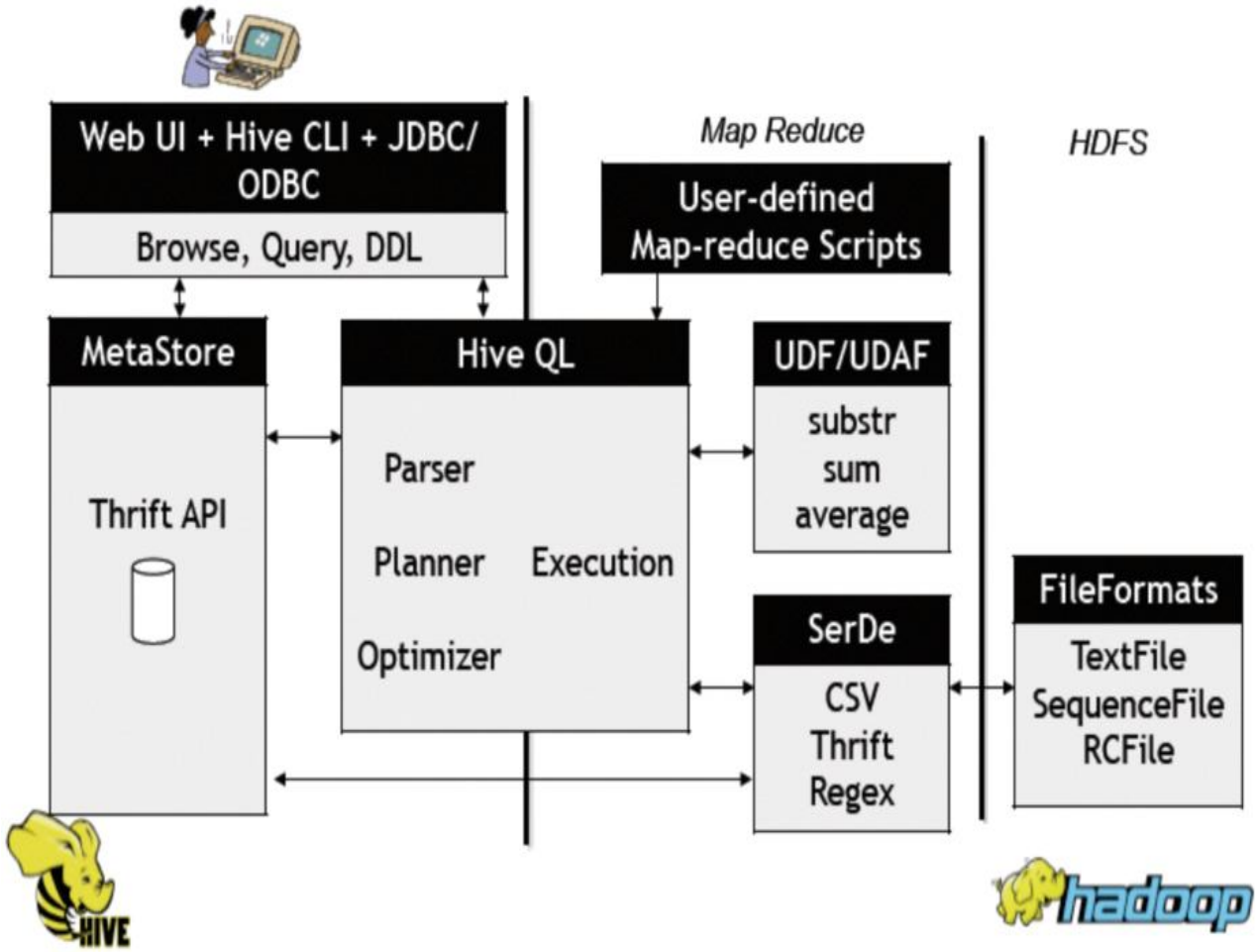

图 1 Hive 数据仓库的系统结构

基于 MapReduce 的数据仓库在超大规模数据分析中扮演了重要角色，对于典型的 Web 服务供应商，这些分析有助于它们快速理解动态的用户行为及变化的用户需求。数据存储结构是影响数据仓库性能的关键因素之一。Hadoop 系统中常用的文件存 储格式有支持文本的 TextFile 和支持二进制的 SequenceFile 等，它们都属于行存储方式。Facebook 工程师发表的 RCFile: A Fast and Spaceefficient Data Placement Structure in MapReducebased Warehouse Systems 一文，介绍了一种高效的数据存储结构——RCFile（Record Columnar File），并将其应用于 Facebook 的数据仓库 Hive 中。与传统数据库的数据存储结构相比，RCFile 更有效地满足了基于 MapReduce 的 数据仓库的四个关键需求，即 Fast data loading、Fast query processing、Highly efficient storage space utilization 和 Strong adaptivity to highly dynamic workload patterns。

数据仓库的需求

基于 Facebook 系统特征和用户数据的分析，在 MapReduce 计算环境下，数据仓库对于数据存储结构有四个关键需求。

Fast data loading

对于 Facebook 的产品数据仓库而言，快速加载数据（写数据）是非常关键的。每天大约有超过 20TB 的数据上传到 Facebook 的数据仓库，由于数据加载期间网络和磁盘流量会干扰正常的查询执行，因此缩短数据加载时间是非常必要的。

Fast query processing

为了满足实时性的网站请求和支持高并发用户提交查询的大量读负载，查询响应时间是非常关键的，这要求底层存储结构能够随着查询数量的增加而保持高速的查询处理。

Highly efficient storage space utilization

高速增长的用户活动总是需要可扩展的存储容量和计算能力，有限的磁盘空间需要合理管理海量数据的存储。实际上，该问题的解决方案就是最大化磁盘空间利用率。

Strong adaptivity to highly dynamic workload patterns

同一份数据集会供给不同应用的用户，通过各种方式来分析。某些数据分析是例行过程，按照某种固定模式周期性执行；而另一些则是从中间平台发起的查 询。大多数负载不遵循任何规则模式，这需要底层系统在存储空间有限的前提下，对数据处理中不可预知的动态数据具备高度的适应性，而不是专注于某种特殊的负 载模式。

MapReduce 存储策略

要想设计并实现一种基于 MapReduce 数据仓库的高效数据存储结构，关键挑战是在 MapReduce 计算环境中满足上述四个需求。在传统数据库 系统中，三种数据存储结构被

广泛研究，分别是行存储结构、列存储结构和 PAX 混合存储结构。上面这三种结构都有其自身特点，不过简单移植这些数据库导向的 存储结构到基于 MapReduce 的数据仓库系统并不能很好地满足所有需求。

行存储

如图 2 所示，基于 Hadoop 系统行存储结构的优点在于快速数据加载和动态负载的高适应能力，这是因为行存储保证了相同记录的所有域都在同一个集群 节点，即同一个 HDFS 块。不过，行存储的缺点也是显而易见的，例如它不能支持快速查询处理，因为当查询仅仅针对多列表中的少数几列时，它不能跳过不必要 的列读取；此外，由于混合着不同数据值的列，行存储不易获得一个极高的压缩比，即空间利用率不易大幅提高。尽管通过熵编码和利用列相关性能够获得一个较好 的压缩比，但是复杂数据存储实现会导致解压开销增大。

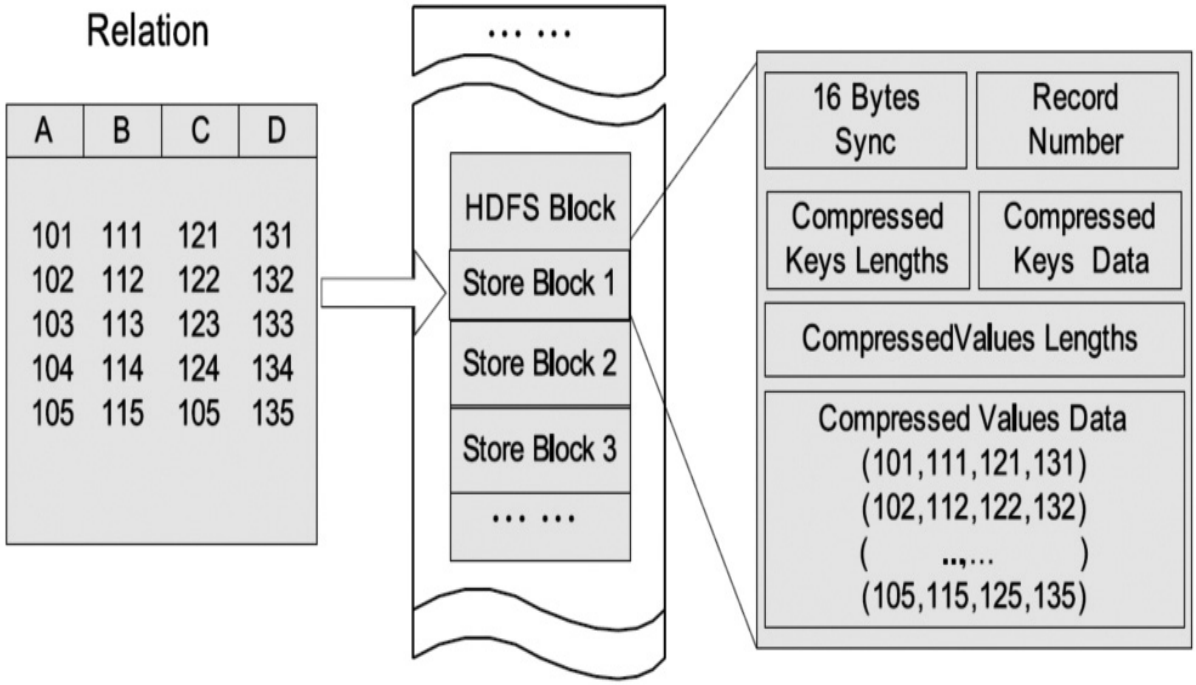

图 2 HDFS 块内行存储的例子

列存储

图 3 显示了在 HDFS 上按照列组存储表格的例子。在这个例子中，列 A 和列 B 存储在同一列组，而列 C 和列 D 分别存储在单独的列组。查询时列存储能够避 免读不必要的列，并且压缩一个列中的相似数据能够达到较高的压缩比。然而，由于元组重构的较高开销，它并不能提供基于 Hadoop 系统的快速查询处理。列 存储不能保证同一记录的所有域都存储在同一集群节点，例如图 2 的例子中，记录的 4 个域存储在位于不同节点的 3 个 HDFS 块中。因此，记录的重构将导致通过 集群节点网络的大量数据传输。尽管预先分组后，

多个列在一起能够减少开销，但是对于高度动态的负载模式，它并不具备很好的适应性。除非所有列组根据可能的 查询预先创建，否则对于一个查询需要一个不可预知的列组合，一个记录的重构或许需要 2 个或多个列组。再者由于多个组之间的列交叠，列组可能会创建多余的列 数据存储，这导致存储利用率的降低。

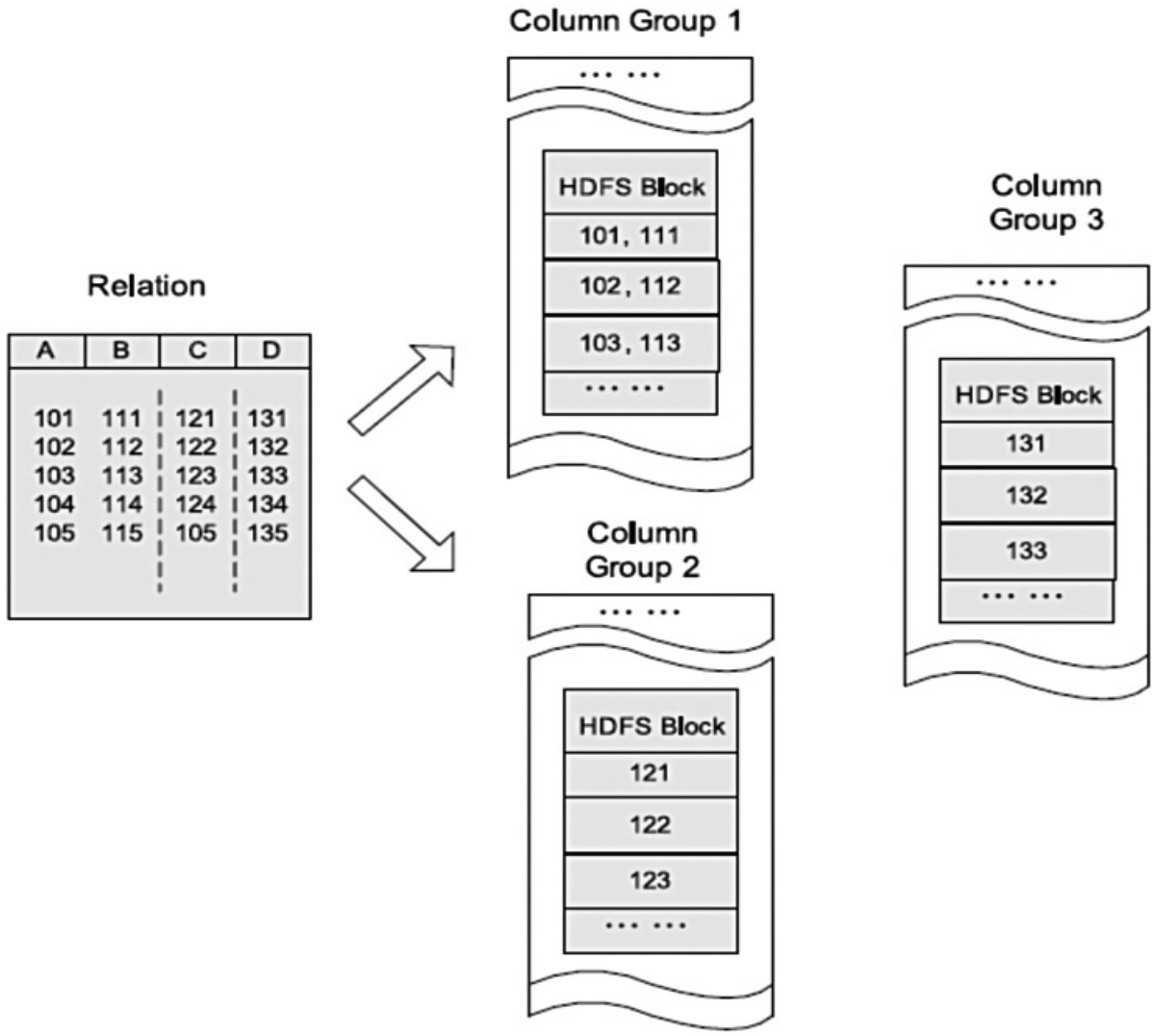

图 3 HDFS 块内列存储的例子

PAX 混合存储

PAX 存储模型（用于 Data Morphing 存储技术）使用混合存储方式，目的在于提升 CPU Cache 性能。对于记录中来自不同列的多个域，PAX 将它们放在一个磁盘页中。在每个磁盘页中，PAX 使用一个迷你页来存储属于每个列的所有域，并使用 一个页头来存储迷你页的指针。类似于行存储，PAX 对多种动态查询有很强的适应能力。然而，它并不能满足大型分布式系统对于高存储空间利用率和快速查询处 理的需求，原因在于：首先，PAX 没有数据压缩的相关工作，这部分与 Cache 优化关系不大，但对于大规模数据处理

系统是非常关键的，它提供了列维度数据 压缩的可能性；其次，PAX 不能提升 I/O 性能，因为它不能改变实际的页内容，该限制使得大规模数据扫描时不易实现快速查询处理；再次，PAX 用固定的页 作为数据组织的基本单位，按照这个大小，在海量数据处理系统中，PAX 将不会有效存储不同大小类型的数据域。本文介绍的是 RCF i l e 数据存储结构在 Hadoop 系统上的实现。该结构强调：第一，RCFile 存储的表是水平划分的，分为多个行组， 每个行组再被垂直划分， 以便每列单独存储；第二，RCFile 在每个行组中利用一个列维度的数据压缩，并提供一种 Lazy 解压（decompression）技术来在查询执行时 避免不必要的列解压；第三，RCFile 支持弹性的行组大小，行组大小需要权衡数据压缩性能和查询性能两方面。

RCFile 的设计与实现

RCFile（Record Columnar File）存储结构遵循的是“先水平划分，再垂直划分”的设计理念，这个想法来源于 PAX。它结合了行存储和列存储的优点：首先，RCFile 保证同一行 的数据位于同一节点，因此元组重构的开销很低；其次，像列存储一样，RCFile 能够利用列维度的数据压缩，并且能跳过不必要的列读取。图 4 是一个 HDFS 块内 RCFile 方式存储的例子。

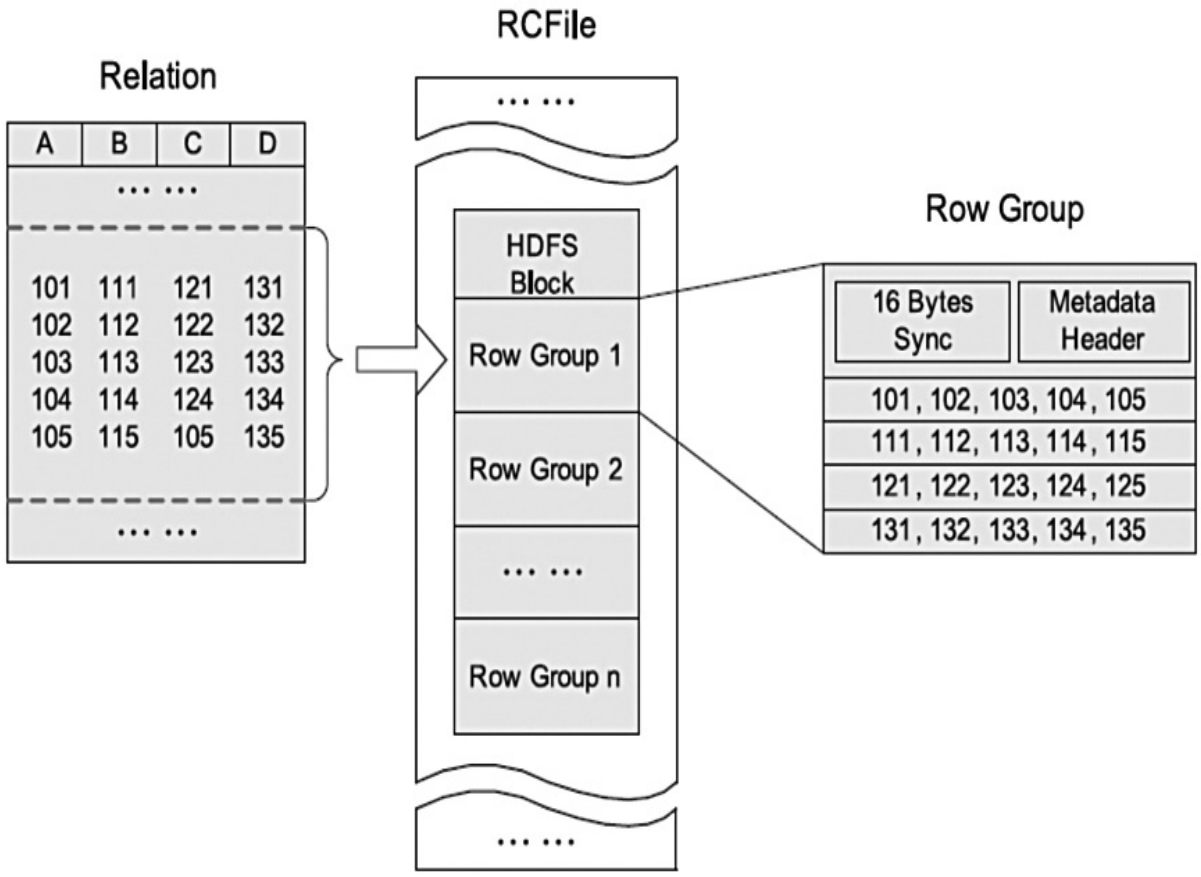

图 4 HDFS 块内 RCFile 方式存储的例子

数据格式

RCFile 在 HDFS 分布式文件系统之上设计并实现，如图 4 所示，RCFile 按照下面的数据格式来存储一张表。

RCFile 基于 HDFS 架构，表格占用多个 HDFS 块。

每个 HDFS 块中，RCFile 以行组为基本单位来组织记录。也就是说，存储在一个 HDFS 块中的所有记录被划分为多个行组。对于一张表，所有行组大小都相同。一个 HDFS 块会有一个或多个行组。

一个行组包括三个部分。第一部分是行组头部的同步标识，主要用于分隔 HDFS 块中的两个连续行组；第二部分是行组的元数据头部，用于存储行组单元的 信息，包括行组中的记录数、每个列的字节数、列中每个域的字节数；第三部分是表格数据段，即实际的列存储数据。在该部分中，同一列的所有域顺序存储。从图 4 可以看出，首先存储了列 A 的所有域，然后存储列 B 的所有域等。

压缩方式

RCFile 的每个行组中，元数据头部和表格数据段分别进行压缩。

对于所有元数据头部，RCFile 使用 RLE（Run Length Encoding）算法来压缩数据。由于同一列中所有域的长度值都顺序存储在该部分，RLE 算法能够找到重复值的长序列，尤其对于固定的域长度。

表格数据段不会作为整个单元来压缩；相反每个列被独立压缩，使用 Gzip 压缩算法。RCFile 使用重量级的 Gzip 压缩算法，是为了获得较好的压 缩比，而不使用 RLE 算法的原因在于此时列数据非排序。此外，由于 Lazy 压缩策略，当处理一个行组时，RCFile 不需要解压所有列。因此，相对较高的 Gzip 解压开销可以减少。

尽管 RCFile 对表格数据的所有列使用同样的压缩算法，不过如果使用不同的算法来压缩不同列或许效果会更好。RCFile 将来的工作之一可能就是根据每列的数据类型和数据分布来自适应选择最好的压缩算法。

数据追加

RCFile 不支持任意方式的数据写操作，仅提供一种追加接口，这是因为底层的 HDFS 当前仅仅支持数据追加写文件尾部。数据追加方法描述如下。

RCFile 为每列创建并维护一个内存 column holder，当记录追加时，所有域被分发，每个域追加到其对应的 column holder。此外，RCFile 在元数据头部中记录每个域对应的元数据。

RCFile 提供两个参数来控制在刷写到磁盘之前，内存中缓存多少个记录。一个参数是记录数的限制，另一个是内存缓存的大小限制。

RCFile 首先压缩元数据头部并写到磁盘，然后分别压缩每个 column holder，并将压缩后的 column holder 刷写到底层文件系统中的一个行组中。

数据读取和 Lazy 解压

在 MapReduce 框架中，mapper 将顺序处理 HDFS 块中的每个行组。当处理一个行组时，RCFile 无需全部读取行组的全部内容到内存。

相反，它仅仅读元数据头部和给定查询需要的列。因此，它可以跳过不必要的列以获得列存储的 I/O 优势。例如，表 tbl(c1, c2, c3, c4)有 4 个列，做一次查询“SELECT c1 FROM tbl WHERE c4 = 1”，对每个行组，RCFile 仅仅读取 c1 和 c4 列的内容。在元数据头部和需要的列数据加载到内存中后，它们需要解压。元数据头部总会解压并在内存中维 护直到 RCFile 处理下一个行组。然而，RCFile 不会解压所有加载的列，相反，它使用一种 Lazy 解压技术。

Lazy 解压意味着列将不会在内存解压，直到 RCFile 决定列中数据真正对查询执行有用。由于查询使用各种 WHERE 条件，Lazy 解压非常有 用。如果一个 WHERE 条件不能被行组中的所有记录满足，那么 RCFile 将不会解压 WHERE 条件中不满足的列。例如，在上述查询中，所有行组中的列 c4 都解压了。然而，对于一个行组，如果列 c4 中没有值为 1 的域，那么就无需解压列 c1。

行组大小

I/O 性能是 RCFile 关注的重点，因此 RCFile 需要行组够大并且大小可变。行组大小和下面几个因素相关。

行组大的话，数据压缩效率会比行组小时更有效。根据对 Facebook 日常应用的观察，当行组大小达到一个阈值后，增加行组大小并不能进一步增加 Gzip 算法下的压缩比。

行组变大能够提升数据压缩效率并减少存储量。因此，如果对缩减存储空间方面有强烈需求，则不建议选择使用小行组。需要注意的是，当行组的大小超过 4MB，数据的压缩比将趋于一致。

尽管行组变大有助于减少表格的存储规模，但是可能会损害数据的读性能，因为这样减少了 Lazy 解压带来的性能提升。而且行组变大会占用更多的内存， 这会影响并发执行的其他 MapReduce 作业。考虑到存储空间和查询效率两个方面，Facebook 选择 4MB 作为默认的行组大小，当然也允许用户自行 选择参数进行配置。

小结

本文简单介绍了 RCFile 存储结构，其广泛应用于 Facebook 公司的数据分析系统 Hive 中。首先，RCFile 具备相当于行存储的数据加载 速度和负载适应能力；其次，RCFile 的读优化可以在扫描表格时避免不必要的列读取，测试显示在多数情况下，它比其他结构拥有更好的性能；再 次，RCFile 使用列维度的压缩，因此能够有效提升存储空间利用率。

为了提高存储空间利用率，Facebook 各产品线应用产生的数据从 2010 年起均采用 RCFile 结构存储，按行存储 （SequenceFile/TextFile）结构保存的数据集也转存为 RCFile 格式。此外，Yahoo 公司也在 Pig 数据分析系统中集成了 RCFile，RCFile 正在用于另一个基于 Hadoop 的数据管理系统 Howl（http://wiki.apache.org/pig /Howl）。而且，根据 Hive 开发社区的交流，RCFile 也成功整合加入其他基于 MapReduce 的数据分析平台。有理由相信，作为数据存储标准 的 RCFile，将继续在 MapReduce 环境下的大规模数据分析中扮演重要角色。

转发到新浪微博 转播到腾讯微博