高技术通讯 2010 年 第 20 卷 第 7 期:719 ~ 726

doi:10.3772/j.issn.1002-0470.2010.07.011

基于 MapReduce 实现空间查询的研究①

张书彬②*** 韩冀中③* 刘志勇* 王 凯***

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院研究生院 北京 100190)

摘 要 为了解决原有单机空间数据库管理系统在存储能力、计算能力和可扩展能力上的不足,在分布式文件系统的基础上设计并实现了一种基于并行计算框架 MapReduce 的空间矢量数据管理系统 Meadow,并重点讨论了利用 MapReduce 并行处理空间查询的数据分割方法、副本避免方法及关键算法的设计策略,最后给出了定量的分析和论证。实验表明,相对于单机 Oracle Spatial,利用 MapReduce 并行处理空间查询的方法具有良好的性能和近似线性的加速比。实验结果也验证了 MapReduce 在诸如空间数据管理系统这种基于小规模集群的计算密集型应用中同样具有良好的性能。

关键词 空间数据管理系统, MapReduce, 空间查询, 分布式系统

0 引 言

人类活动中约 75% ~ 80% 的信息与地理空间位置有关[1]。地理信息系统(geographic information system, GIS)作为获取、存储、处理、管理、分析和显示地理空间数据及其属性信息的重要工具,近年来得到了广泛关注和迅速发展。空间数据管理系统是 GIS 系统的核心构件,其性能在很大程度上决定了整个 GIS 系统的性能与可扩展能力。

为了提供高效的空间数据管理能力,人们在单机数据库管理系统的基础上先后提出了多种空间数据管理系统结构,目前较为流行的是"关系型数据库(RDBMS) + 空间数据引擎(SDE)"和"扩展对象关系型数据库(Extended ORDBMS)"两种结构。在"RDBMS + SDE"结构中,空间数据存储在 RDBMS 中;当用户访问数据时,SDE 从 RDBMS 中读取数据并处理,将结果返回给用户。这类系统的典型代表有 ESRI 的 ArcSDE 和 MapInfo 的 SpatialWare 等。ORDBMS 支持抽象数据类型及其相关操作的定义,因此,用户可以增加空间数据类型及相关函数,而且可以使用新增的空间数据类型和扩展的结构化查询语言(SQL)来操作空间数据。Oracle 的 Oracle Spatial、IBM 的 DB2 Spatial Extender 和 PostgreSQL 的 PostGIS 等都属于该类型。

当地理空间数据集规模较小时,利用单机空间数据库管理系统即可提供良好的服务。但是,随着数字测绘技术、GPS 技术、图像处理技术、传感器技术以及遥感技术的迅速发展,地理空间信息获取技术日渐成熟,地理空间数据规模迅速扩大,已成为海量数据来源之一。这使得单节点系统在性能、可扩展性和可靠性等方面的不足凸显出来。虽然采用 RAID、NAS 等技术可以在一定程度上缓解系统的存储压力,但是仍然很难满足 GIS 应用日益增长的对计算能力和 I/O 能力的需求。使用并行数据库系统来解决空间数据的存储和查询处理问题似乎是一种很好的选择[2]。但是,现有并行数据库系统价格昂贵,且存在一些不足,如非自动的数据分布、低效的通信、糟糕的负载平衡和缺乏有效的容错机制等。此外,并行数据库系统在并行空间查询处理方面进展缓慢,有些空间查询难以并行处理。

随着云计算技术的发展,大规模数据处理成为未来的趋势,Google 等公司在利用集群并行处理大数据集方面取得了很大的成功。MapReduce[3] 是 Google 提出的并行编程模型和分布式计算框架,用于基于大规模集群的数据密集型应用。除了被应用到搜索引擎中,MapReduce 还被广泛应用于分布式查询、分布式排序、日志分析、文档归类以及机器学习等应用中[4]。文献[5]扩展了 MapReduce 模型,以

① 863 计划(2009AA12Z226),973 计划(2007CB310805)和国家自然科学基金(60752001)资助项目。
② 男,1979 年生,博士;研究方向:空间数据管理,并行与分布式算法;E-mail: zhangshubin@ict.ac.cn
③ 通讯作者,E-mail: hjz@ict.ac.cn
(收稿日期:2009-03-06)

便更好地处理关系数据。此外,MapReduce 编程模型也被扩展到其它体系结构,例如多核结构[6],Cell 结构[7]。

本文在分布式文件系统的基础上设计并实现了基于 MapReduce 的空间矢量数据管理系统 Meadow,并针对使用 MapReduce 处理空间查询所遇到的关键问题和相关算法展开了研究。Meadow 的空间查询处理方法不依赖于空间索引,尤其适合于大规模空间数据处理和实时分析。本文的算法设计策略也可以用于其他并行处理架构。实验表明 MapReduce 在空间数据管理系统这种基于小规模集群的计算密集型应用中也具备良好性能。

1 研究背景以及相关研究

1.1 MapReduce 编程模型

Google 的 MapReduce 编程模型一经推出,就受到了广泛的关注,Paterson 在文献[8]中将 MapReduce 称为数据中心的指令。MapReduce 是一种非常适合并行的编程模型,它提供了一套方便的编写分布式应用程序的接口。MapReduce 编程模型表达并行算法的两个主要概念 Map(映射)和 Reduce(归约)来源于函数式编程语言,还有部分特性来源于矢量编程语言。MapReduce 编程模型可以如下表示:

$$Map:(k_1,v_1)\rightarrow[(k_2,v_2)]$$
$$Reduce:(k_2,[v_2])\rightarrow[k_3,v_3] \quad (1)$$

如式(1)所示,Map 函数将用户定义的应用逻辑循环运用于每个输入的键值对 (k_1, v_1) 上,并生成中间键值对列表 $[(k_2,v_2)]$。随后,Reduce 函数根据用户定义的逻辑处理具有相同中间键值 k_2 的所有中间值 $[v_2]$,并产生输出键值对列表 $[k_3, v_3]$。由编程模型可知,Map 操作和 Reduce 操作均可高度并行执行。

1.2 基于 MapReduce 的分布式存储计算框架

Google 的多数业务逻辑都可以用 MapReduce 编程模型表达。为此,Google 开发出了基于集群的一组分布式计算与数据管理平台,主要由分布式文件系统 Google File System (GFS)[9]、分布式结构化数据管理系统 Bigtable[10]、分布式锁服务器 Chubby[11]以及 MapReduce 分布式计算框架[3]等构件组成。

尽管不是所有的应用都可以用 Map 和 Reduce 函数表达,但与传统并行计算领域中广泛应用的 MPI/PVM 不同的是,MapReduce 分布式计算框架隐藏了更多集群物理结构以及节点/网络可用性方面的细节,并向用户提供了一套类似于函数语言的数据处理接口,将并行程序开发人员从复杂的系统细节中解放出来,使之将更多注意力放在应用逻辑的核心算法上。计算框架负责管理诸如任务分布、作业调度、容错、负载平衡、文件分布以及节点通信等问题,最终简化并行程序开发。

Meadow 是基于 Hadoop、HBase 和 Hive 等开源技术实现的。Hadoop[12]是由 Yahoo! 提供支持的基于 Google 技术的开源分布式平台,它支持大规模集群上的可靠的分布式计算和海量数据存储。Hadoop 主要由分布式文件系统 HDFS 和 MapReduce 框架组成。目前,Hadoop 已经在 Yahoo!、Facebook、AOL、IBM 和百度等众多公司和科研机构中广泛应用。此外,开源社区还在 HDFS 的基础上提供了与 Bigtable 作用相当的 HBase[13]以提供对结构化数据的分布式存储管理能力。由 Facebook 发起的 Hive 项目[14]是一种建立在 Hadoop 之上的数据仓库基础设施,它力图将 SQL 与 MapReduce 整合在一起,让用户以类似 SQL 的查询语言访问和处理保存在 HDFS 中的数据。

1.3 空间矢量数据查询

空间应用中较常见的对空间矢量数据的查询有空间选择查询[15]、空间连接查询[16-18]和最近邻查询[19]等。其中,空间选择查询和连接查询是最基本的查询操作。空间选择查询包括空间点查询和空间区域查询[20]。点查询获取包含给定点的所有空间对象,区域查询获取与指定空间区域存在相交或包含关系的所有空间对象。空间连接是根据空间谓词(如相交、距离范围内等)将两组空间对象结合在一起以产生第三组空间对象的操作。这是空间应用中的一个重要操作,空间应用经常需要基于某种空间关系结合两个空间数据集。空间连接查询相当于非空间关系数据库中的相等连接操作,但是,空间连接比相等连接要复杂得多:空间对象间没有自然全序关系,空间连接谓词比相等谓词更复杂,而且,空间对象通常也比相等连接中的关系对象要大得多。

由于空间查询多为计算密集型,因此,对空间查询的处理多采用两阶段的过滤-精炼策略。第一阶段,将空间对象用其最小外包矩形(minimum bounding rectangle,MBR)来表示。这样做是由于矩形间的关系判断或运算要比不规则空间对象间的关系判断或运算容易得多。这一阶段被称为过滤阶段,该阶段的结果集包含了 MBR 满足查询条件的候选者。第二阶段被称为精炼阶段,对过滤的结果集使用精确空间属性进行处理,这是一个计算代价较大的过

程,但在过滤阶段的帮助下,此阶段的输入集合只剩下较少的候选者。

由于单机空间数据库系统的 I/O 瓶颈,原有的空间查询处理算法多利用某种空间索引(如 R 树,R * 树)来减少 I/O 次数,加速过滤阶段。但在很多时候,空间计算的输入数据集是前次计算的临时结果集,或者尚未建立空间索引。此时,建立空间索引的高昂开销使得基于索引的空间查询处理方法耗时较多。基于数据分割和计算并行的 MapReduce 算法是解决大数据量空间查询问题的较好方案。但是,如何对空间数据进行有效分割,以及高效地进行 MapReduce 空间查询处理,是迫切需要解决的问题。

2 设计与实现

本节首先介绍 Meadow 原型系统的结构,然后重点介绍利用 MapReduce 并行处理空间查询的数据分割方法、副本避免方法及关键算法设计策略。

图 1 将我们设计实现的 Meadow 原型系统的结构与原有系统进行了对比。在 Meadow 中,空间应用的查询请求通过支持空间扩展的类 SQL 语句提交,并由任务适配层翻译成 MapReduce 程序。任务适配层是通过对 Hive 进行扩展实现的。具体的空间查询处理由 MapReduce 框架调用定制的空间算子库完成。底层由分布式文件系统 HDFS 和分布式数据管理系统 HBase 提供对空间数据的存储支持。Meadow 的存储和计算能力可以随着系统规模的增大而不断扩展。而在原有的"RDBMS + SDE"结构和"Extended ORDBMS"结构系统中,空间数据库管理系统则会成为系统性能的瓶颈。

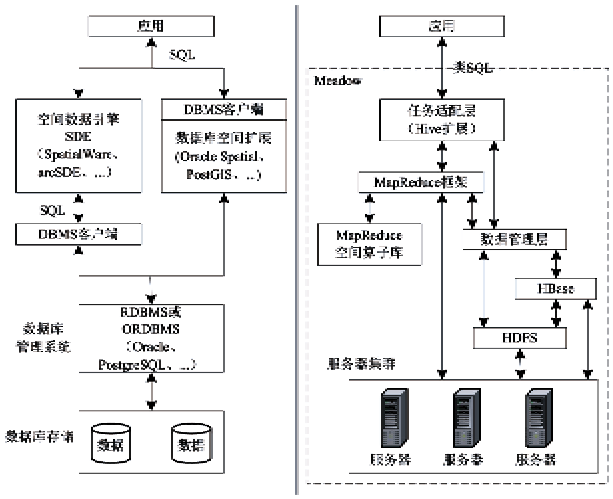

图 1 空间数据管理系统结构对比

本文重点描述和评价 Meadow 的空间查询处理能力。假定空间数据被保存到 HDFS 中,Meadow 将其转换为自定义的空间矢量数据存储格式,每行表达一个空间对象,列与列之间用分隔符区分,以方便 MapReduce 读取和处理。对象的空间属性均以 OGC 的 WKT 格式表达。

2.1 空间数据分割方法

原有空间查询处理算法多依赖于空间索引。但在很多情况下,数据集并未建立空间索引。一种解决方案是实时为数据集建立空间索引,然后使用基于索引的算法处理查询。但是,建立空间索引的代价较高,严重影响查询效率。在 Meadow 中,由于 MapReduce 和 HDFS 具有数据动态分布的特点,所以,不使用索引的查询处理算法效率可能更高。

数据分割并行在并行关系数据库中已经得到成功运用[21],下面以关系连接查询为例进行说明。对两个数据集 R 和 S 进行并行连接(表示为 $R \rhd\lhd S$)的基本思想是将 R 和 S 分割成 n 个桶,$R_1, R_2, \cdots, R_n$ 和 $S_1, S_2, \cdots, S_n$,对所有桶而言,如果 $i \neq j$,则 $R_i \cap R_j = \varnothing$ 且 $S_i \cap S_j = \varnothing$,那么,$R \rhd\lhd S \equiv \bigcup_{i=1}^{n}(R_i \cap S_i)$。这种分割方法具有"单次分配,单次连接(single-assignment single-join, SASJ)"的性质。每对 R 桶和 S 桶的连接被称为一个任务,只能被分配到一个节点上执行。

但是,由于空间对象具有空间范围,故对空间数据集而言,不存在数据独立的 SASJ 分割方法。此外,空间对象不均匀分布(数据倾斜)现象非常普遍。典型的空间数据分割方法是将整个空间区域分割成网格(分片),并将空间对象根据其位置分配到不同的网格中,然后将网格按照各种方法合并成桶。由于空间对象可能与多个网格相交,因此,空间数据分割方法要么是"多次分配,单次连接(multi-assignment single-join, MASJ)"的;要么是"单次分配,多次连接(single-assignment multi-join, SAMJ)"的。使用 SAMJ 方法时,各计算任务间需要通信以获取计算所需的空间对象。因为 Map/Reduce 任务间彼此独立,没有通信,所以,MASJ 方法更适合于 MapReduce 并行算法。

2.1.1 桶数目的确定

桶的最小数目由数个因素决定,高效的算法应保证各桶的空间对象可以完全放入内存,并使用内存算法处理空间查询。桶的最小数目可以如下进行估计:

$$P = \lceil (\| R \| + \| S \|) \times (1 + p) \times Size_{obj} / M \rceil \quad (2)$$

其中，$\| R \|$ 和 $\| S \|$ 分别表示数据集 R 和 S 的势，p 表示空间对象复制系数(见3.2节)，M 表示各节点的平均内存大小，$Size_{obj}$ 表示空间对象的平均大小。

2.1.2 分片合并方法

原有的空间数据分割方法多使用轮询方法(round-Robin，RR)和哈希方法(Hash)将分片合并成桶。轮询方法首先将分片按行编号，然后使用轮询算法将分片映射到不同的桶。该方法最初被用于处理并行关系连接中的数据倾斜问题，后被 Patel 用于空间数据分割[18]。哈希方法与轮询方法一样，分片被按行编号，然后用哈希算法将分片映射到不同的桶中。这种方法也被 Patel 在文献[18]中用于空间数据分割。

但是，在二维空间中，使用轮询方法和哈希方法可能将位于不同行上的相邻分片映射到同一个桶，不利于解决空间数据倾斜问题。空间填充曲线在将二维空间映射为一维的过程中可以尽可能地保证距离不变性，利用空间填充曲线的这一特点，本文将 Z 序方法(Z-order)和 Hilbert 序方法(Hilbert-order)用于 Meadow 系统中的分片合并。Z 序方法首先将每个分片按照对应的 Z 曲线值编码并排序，然后用轮询的方法将分片合并成桶。Hilbert 序方法将每个分片用对应的 Hilbert 曲线值编码并排序，然后使用轮询的方法将分片合并成桶。

上述方法利用空间对象的分布特点和空间填充曲线的原理，使各桶的对象数更加均匀。Z 序方法计算方便，与 Hilbert 序方法性能相当，并且，从分片的 Z 编码可以很容易地计算出该分片的行列号，这一特点在算法设计中很有帮助。Hilbert 序的性能最好，但计算量稍大。Meadow 使用了 Z 序方法实现分片合并，具体分析见本文第3.2节。

2.2 副本避免方法

与 SASJ 方法相比，MASJ 方法的一个特点是存在数据复制问题，即在不同任务(桶)的输入和输出数据集中可能有相同的对象。这些对象会被重复处理，不仅降低了算法性能，而且，影响计算结果的正确性。

有两种方法可以将副本从结果集中去除：副本消除方法和副本避免方法。副本消除方法需要将查询结果汇总并排序，然后消除副本。这种方法只能在主要查询操作完成后执行，因而计算代价较高。

为了避免额外的计算步骤，Meadow 使用了副本避免方法：每对空间对象 T_R 和 T_S，如果均被复制到多个桶中，那么，T_R 和 T_S 的空间关系判断只应在 T_R 和 T_S 均被复制的某个特定的桶中进行。为此，在各计算任务中增加一个判断操作，原有方法为参考点方法[22]，该方法计算出 T_R 和 T_S 的 MBR 的特定公共点，并只在该点所属的任务中执行计算。Meadow 使用了一种改进方法，称之为参考分片法。该方法根据对象的分片信息，计算出 T_R 和 T_S 的最小公共分片，并只在该分片所在的桶中进行空间关系判断。

如图2所示，整个图层被分割成16个分片。T_R 和 T_S 的最小公共分片是 *Tile* 3。因此，两者的空间关系判断只会在 *Tile* 3 所在的桶进行。与参考点方法相比，参考分片法的优点是充分利用了数据分割时获取的对象空间位置信息，计算量小，而且，可以和2.4节介绍的分条扫描法很好地配合起来。

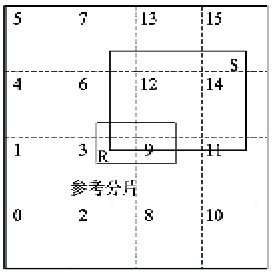

图2 参考分片法

2.3 空间选择查询的并行化

空间选择查询是最重要的一种空间查询操作，它可以作为其他空间查询操作(如最近邻查询和空间连接查询)的基础。典型的空间选择查询，包括点查询和区域查询：

- 点查询

$$Q(P) = \{O \mid O.contains(P), O \in M\} \quad (3)$$

给定查询点 P 和空间对象集 M，空间点查询需要找出 M 中所有包含点 P 的空间对象。

- 区域查询

$$Q(R) = \{O \mid O.G \cap R.G \neq \varnothing \| R.contains(O), O \in M\} \quad (4)$$

给定多边形查询区域 R 和空间对象集 M，空间区域查询查找 M 中所有与 R 相交或被 R 包含的空间对象。查询区域为矩形时的区域查询也被称为窗口查询。

MapReduce 并行处理空间选择查询的数据流图如图3所示。以 HDFS 中输入文件分块或 HBase 表

分块作为 Map 任务划分依据,使用空间选择查询算子(点查询算子或区域查询算子)作为 Map 函数,k_1 为空间对象在文件中的行号或在 HBase 中的 ID,v_1 为该对象的所有属性值。各 Map 任务分别对不同的文件块或表分块中的对象执行空间选择查询,Map 任务的计算结果直接放入 HBase 或 HDFS 作为最终结果。由于不需要对 Map 任务的结果进行归约运算,因此,空间选择查询处理作业不需要 Reduce 阶段。

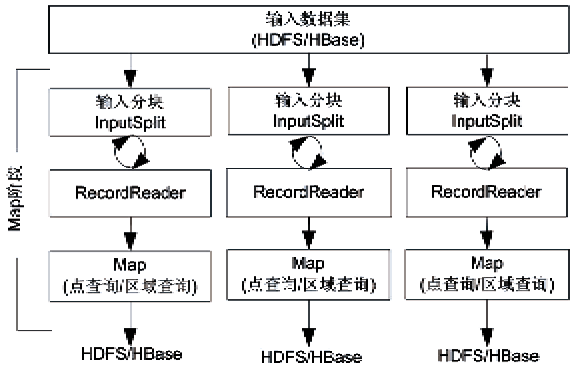

图 3　空间选择查询数据流图

2.4　空间连接查询的并行化

典型的空间连接查询的示例,包含"找出相距 200km 以内的城市对"和"找出所有跨越水系的道路"。通过变换空间谓词,空间连接查询可以演变出很多重要的空间分析算法,如交、并、差、剪切、被剪切和缓冲分析等,这类查询计算开销较大。空间连接查询的形式化描述为

$$SJ(R,S)=\{(r,s)\mid r.join(s),r\in R,s\in S\} \tag{5}$$

Meadow 系统空间连接查询处理的数据流图如图 4 所示。在 Map 阶段,以 HDFS 输入文件分块或 HBase 表分块作为 Map 任务划分依据,使用对象同质化算子和空间数据分割算子作为 Map 函数,k_1 为每个空间对象在文件中的行号或在 HBase 中的 ID,v_1 为该对象的属性值。Map 任务首先对来自两个不同数据源的数据进行同质化,使来自不同数据源的对象具备相同的属性,同时加入数据源属性,标识出该对象所属数据源,然后对同质化后的对象进行空间分割,分割后得到的各对象的分片信息也被作为对象属性保存。对于每个空间对象,以其对应的桶编号作为 Map 结果的 k_2 值,以对象属性值作为 v_2 值。由于使用的是 MASJ 方法,因此,每个空间对象可能产生一个或多个(k_2, v_2)对。

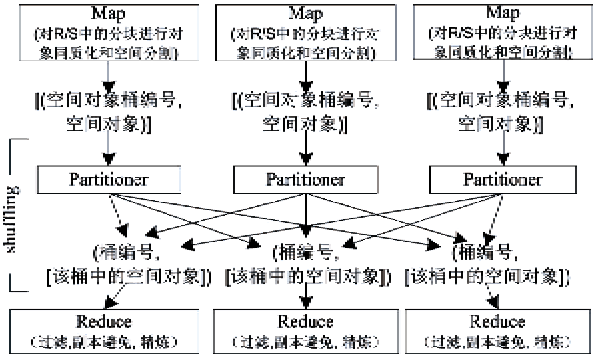

图 4　空间连接查询数据流图

MapReduce 框架会将同一个桶中的对象,即具有相同桶编号 k_2 的空间对象,交由同一个 Reduce 任务处理。各 Reduce 任务分为过滤和精炼两阶段进行。为了加快过滤阶段,在各 Reduce 任务的过滤阶段使用了一种新的基于分条的平面扫描算法,将整个图层分成多个水平条状区,利用 v_2 中的对象分片信息将桶内的所有空间对象映射到不同的分条中,然后,对各分条同时进行平面扫描以加速过滤阶段。该扫描算法充分利用在对象分片信息中隐含的空间位置信息,减少扫描计算量,从而加速过滤阶段。在过滤的过程中,使用参考分片法避免产生副本。具体的空间连接算法见文献[23]。

3　性能评估

3.1　测试环境与测试数据集

测试平台是由 7 个节点组成的集群。节点型号为 Dell Power Edge SC430,CPU 为 Intel Pentuim IV 2.8GHz,内存为 2GB,硬盘为 80GB 7200 转的 SATA 硬盘。操作系统为 RedHat AS4.4,Hadoop 的版本为 0.19.0,Java 虚拟机版本为 1.6.0。一个节点作为 NameNode/JobTracker,其余节点作为 DataNode/TaskTracker。TaskTracker 被配置为可以同时执行 2 个 Map 任务和 2 个 Reduce 任务。

由于本文目的是讨论如何使用 MapReduce 并行处理空间查询,而且,目前单机空间数据库系统更为常见,因此,对比系统采用性能较好的单机版 Oracle 11g 系统。

实验使用了两个数据集,均来自 TIGER/Line 文件[24]。Road 数据集包含加州的道路信息,Hydrography 数据集包含加州的水系对象,如河流、运河、湖泊等。表 1 列出了这两个数据集的基本参数。Oracle Spatial 为这两个数据集建立空间索引的时间分

别为 411.40s 和 61.35s。

表 1 实验数据集

数据类型	对象数目	数据表大小	对象平均点数
Road	2092079	529MB	3.87
Hydrography	373950	135MB	10.77

3.2 空间数据分割方法性能比较

图 5 和图 6 反映了对 Road 数据集使用 4 种数据分割方法(桶数目为 4 或 16)时,分片数目、桶数目和分片到桶的映射方法等分割方法参数对各桶的对象数目变异系数和复制系数的影响。变异系数是指各桶对象数目的最大差值与均值的比率,复制系数是指分割后对象数目增量与原数目的比值。这两个系数越小,说明分割方法性能越好。

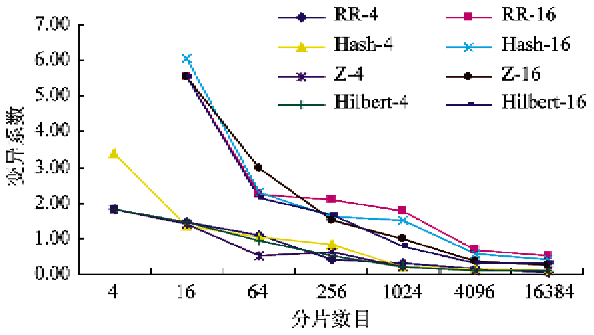

图 5 变异系数

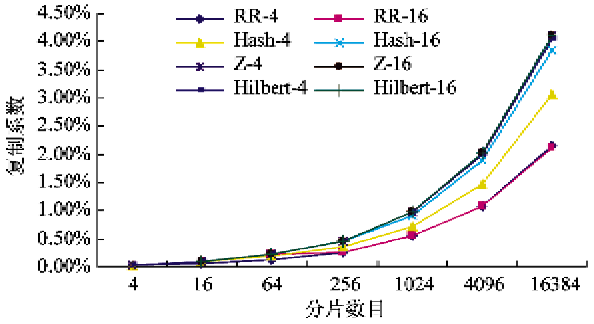

图 6 复制系数

由图 5 可知,首先,就对变异系数的影响而言,Z 序方法和 Hilbert 序方法要优于哈希方法和轮询方法;其次,所有方法的性能随着分片数目的增加而提高,这是由于随着分片数目的提高,较密集的区域被分割成更多的分片,从而使得各桶对象分布更均匀;最后,对于给定分片数目,桶数目越小,各桶对象分布越均匀。

图 6 表明分片数目或桶数目越大,复制系数越高。这是由于对象更易跨越多个分片或桶。而且,对于该数据集,即使分片数目非常大,数据复制也非常有限。因此,可以使用较大的分片数目来保证各任务负载均衡,也可以提高桶数目来获得更高的任务并行度。

根据图 5 和图 6 的结论,且由于 Z 序方法比 Hilbert 方法计算起来更简单,Meadow 系统使用 Z 序方法对空间连接查询的数据集进行空间分割。此外,为了使各任务负载更均衡,使用了较大的分片数目,本文实验所用的分片数为 16384。桶数目下限由公式(2)确定,其中,空间对象平均大小是指过滤阶段空间对象键值对的平均大小。

3.3 Meadow 空间选择查询性能

索引可以加速 Oracle Spatial 空间点查询处理,在查询处理过程中基本不需要读硬盘,其响应时间为 100ms 量级。而 Meadow 需要读取并处理硬盘中的空间对象,故对 Hydrography 数据集进行随机点查询时,在单节点的情况下,Meadow 耗时 109.92s,节点数为 6 时,则需 24.43s。但是,Oracle Spatial 需要耗时 61.35s 为该数据集建立空间索引。

对更常见的窗口选择查询的性能分析使用 Road 数据集,并随机生成了 3 个不同大小的查询窗口,其大小与数据集 MBR 的比例分别为 38.86%、4.11%、1.03%。图 7 中 Query 1、2、3 代表上述 3 个窗口查询。在该图中,Oracle Spatial 的数据为单节点且数据集有索引时的查询性能。

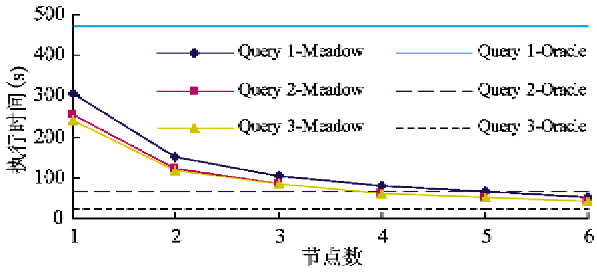

图 7 窗口选择查询性能

由图 7 可知,第一,Meadow 查询性能与查询窗口的大小有一定关系,其原因为查询窗口较大时,过滤结果集和精炼开销也较大。第二,Meadow 窗口选择查询性能与节点数基本呈线性关系。第三,Oracle Spatial 窗口选择查询性能与窗口大小密切相关,窗口越小,性能越好。这是由于 Oracle Spatial 查询处理依赖于索引,查询窗口越小,过滤结果集越小,硬盘 I/O 越少。第四,尽管 Meadow 未使用空间索引,

但只要节点数足够多,Meadow 仍能提供与 Oracle Spatial 相当或更优的窗口查询性能。此外,若将 Oracle Spatial 为数据集建立空间索引所用的 411.40s 考虑进去,Meadow 性能明显优于 Oracle Spatial。

3.4 Meadow 空间连接查询性能

本节对表 1 的两数据集进行空间连接查询性能测试。假定 Meadow 系统计算节点数为 N,各节点可同时执行 R 个 Reduce 任务,Reduce 任务总数为 r。

图 8 显示了参数 r 对 Meadow 空间连接性能的影响。当 $r \leqslant 6$ 时,随着 r 的增加,性能近似线性提高。这是由于 Reduce 任务越多,并行度越高。但当 $6 < r < 12$ 时,性能出现起伏,这是由于各节点的 Reduce 任务数不均衡,执行两个 Reduce 任务的节点决定了整体性能。$r = 12$ 时的性能最优,只需 205.32s,此时,所有 Reduce 任务可与 Map 任务同时运行,节约了耗时较多的异地 I/O 时间。但当 $r > 12$ 时,仅 12 个 Reduce 任务可以同时执行,剩余的 Reduce 任务需要等待被调度,即并行度并未增加。等待执行的 Reduce 任务的 Shuffle 阶段无法与 Map 任务同时进行,性能反而降低。当 Reduce 任务较多时,性能略有改善,但仍低于 $r = 12$ 时的性能。

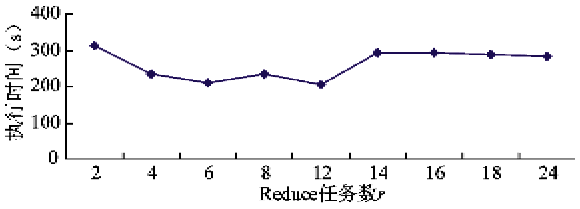

图 8 Reduce 任务数对空间连接性能的影响($N=6$)

图 9 显示了节点数对 Meadow 空间连接性能的影响。图 9 中,Oracle Spatial 的数据为单节点且数据集有索引情况下的性能。我们将 Reduce 任务总数设为 $N \times 2$。可以看到,随着节点数 N 的增加,Meadow 空间连接性能基本呈线性提高。由算法分析可知,Meadow 空间连接查询的线性加速性不会随着系统规模加大而明显改变。由于空间连接查询具有计算和 I/O 密集的特点,故尽管利用空间索引来加速计算,Oracle Spatial 仍需要 1056.17s。这仅比单机 Meadow 系统的性能略优。6 节点 Meadow 系统空间连接性能约为 Oracle Spatial 的 5.14 倍。若考虑 Oracle 建立空间索引的时间,则 6 节点 Meadow 系统空间连接的性能约为 Oracle Spatial 的 7.30 倍。可见,Meadow 性能更好,尤其是在数据集未建立空间索引的情况下。

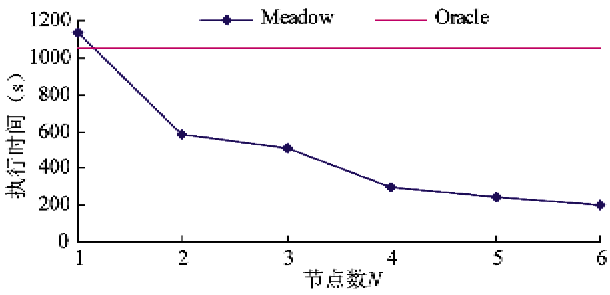

图 9 节点数对空间连接性能的影响

4 结 论

在分布式文件系统的基础上,本文设计并实现了基于 MapReduce 的空间矢量数据管理系统 Meadow。该系统利用集群的并行计算和 I/O 能力满足空间应用日益增长的性能需求。从空间查询处理方法来看,与原有空间数据库系统最大的不同是 Meadow 不依赖于空间索引,而是在数据分割的基础上,充分利用了集群的 I/O 并行和计算并行能力,将空间查询处理与 MapReduce 架构很好地结合起来。

Meadow 系统的空间查询处理过程不依赖于空间索引,故尤其适合于大规模空间数据处理和实时分析。本文介绍了两种新的空间数据分割方法,以及新的副本避免方法。此外,本文以空间选择查询和空间连接查询为例,阐述了如何利用 MapReduce 并行编程模型实现相应的空间查询算法。测试结果表明,相对于 Oracle Spatial,利用 MapReduce 处理空间查询具有良好的性能和系统可扩展性,尤其是当数据集无索引时,这种情况在应用中很常见。本文实验也验证了 MapReduce 在空间数据管理系统这种小规模集群和计算密集型应用中的有效性。

下一步,我们将对 Meadow 的空间查询算子库进行优化,并补充新的空间查询和分析算子。此外,我们将进一步扩展 Hive,使之与 HBase 更好地结合起来,提高系统的实用性。

参考文献

[1] Mckee L. Building the GSDI. Wayland, USA: The Open GIS Consortium, 1996

[2] Patel J, Yu J, Kabra N, et al. Building a scalable geo-spatial DBMS: technology, implementation, and evaluation. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, 1997. 336-347

[3] Dean J, Ghemawat S. MapReduce: simplified data processing

on large clusters. In: Proceedings of 6th Symposium on Operating System Design and Implementation, San Francisco, CA, 2004.137-150
[4] Wikipedia. MapReduce. http://en.wikipedia.org/wiki/Map/reduce, 2008
[5] Yang H, Dasdan A, Hsiao R L, et al. Map-reduce-merge: simplified relational data processing on large clusters. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, New York, NY, USA,2007. 1029-1040
[6] Ranger C, Raghuraman R, Penmetsa A, et al. Evaluating mapreduce for multi-core and multiprocessor systems. In: Proceedings of the 13st International Conference on High-Performance Computer Architecture, Phoenix, Arizona, USA, 2007. 13-24
[7] Kruijf M D, Sankaralingam K. MapReduce for the Cell BE architecture: [Technical Report]. University of Wisconsin, Computer Sciences Technical Report CS-TR-2007, 2007. vol. 1625
[8] Patterson D A. Technical perspective: the data center is the computer. *Communications of the ACM*, 2008, 51:105-105
[9] Ghemawat S, Gobioff H, Leung S T. The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA,2003. 29-43
[10] Chang F, Dean J, Ghemawat S, et al. Bigtable: a distributed storage system for structured data. In: Proceedings of the 7th symposium on Operating Systems Design and Implementation, Seattle, WA, USA,2006. 205-218
[11] Burrows M. The Chubby lock service for loosely-coupled distributed systems. In: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, Seattle, WA, USA,2006. 335-350
[12] The Apache Software Foundation. Hadoop. http://hadoop.apache.org/core/, 2006
[13] The Apache Software Foundation. HBase. http://hadoop.apache.org/hbase/, 2008
[14] The Apache Software Foundation. Hive. http://hadoop.apache.org/hive/, 2008
[15] Guttman A. R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, Boston, MA, USA,1984. 47-57
[16] Gunther O, Ulm F. Efficient computation of spatial joins. In: Proceedings of the 9th International Conference on Data Engineering, Vienna, Austria, 1993. 50-59
[17] Brinkhoff T, Kriegel H P, Seeger B. Efficient processing of spatial joins using R-trees. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 1993. 237-246
[18] Patel J M, DeWitt D J. Partition based spatial-merge join. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada, 1996. 259-270
[19] Roussopoulos N, Kelley S, Vincent F. Nearest neighbor queries. In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA,1995. 71-79
[20] Kriegel H P, Brinkhoff T, Schneider R. Efficient spatial query processing in geographic database systems. *Data Engineering Bulletin*, 1993, 16: 10-15
[21] DeWitt D, Gray J. Parallel database systems: the future of high performance database systems. *Communications of the ACM*, 1992, 35: 85-98
[22] Dittrich J, Seeger B. Data redundancy and duplicate detection in spatial join processing. In: Proceedings of the 16th International Conference on Data Engineering, San Diego, CA, USA,2000. 535-546
[23] Zhang S, Han J, Liu Z, et al. Parallelizing spatial join with MapReduce. In: Proceedings of the 2009 IEEE International Conference on Cluster Computing, New Orleans, Louisiana, USA,2009
[24] U.S. Bureau of the Census. TIGER/Line files(TM), 2007 technical documentation. Washington, DC, USA,2007

Research on implementing spatial queries based on MapReduce

Zhang Shubin[*,**], Han Jizhong[*], Liu Zhiyong[*], Wang Kai[*,**]
([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**] Graduate University of Chinese Academy of Sciences, Beijing 100190)

Abstract

In consideration of the problem that single-node spatial DBMSs have inabilities in storage, computing and scalability, the authors designed and implemented the Meadow, a spatial vector data management system, which was built on the distributed file system and the MapReduce framework. This paper focuses on new spatial data partitioning methods and the duplication avoidance strategy in Meadow, and introduces some MapReduce-based algorithms for processing spatial queries. A quantitative analysis and an argumentation are given finally. The experiment results show that, compared with Oracle Spatial, parallel processing of spatial queries with MapReduce has a good performance and an approximate linear speedup. The performance evaluation also verifies that MapReduce performs well for computing-intensive applications in small-scale clusters.

Key words: spatial data management system, MapReduce, spatial queries, distributed system