(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 107346331 B
(45)授权公告日 2019.08.20

(21)申请号 201710482965.2
(22)申请日 2017.06.22
(65)同一申请的已公布的文献号
申请公布号 CN 107346331 A
(43)申请公布日 2017.11.14
(73)专利权人 武汉大学
地址 430072 湖北省武汉市武昌区珞珈山武汉大学
(72)发明人 余啸 刘进 吴思尧 崔晓晖 张建升 井溢洋
(74)专利代理机构 武汉科皓知识产权代理事务所(特殊普通合伙) 42222
代理人 魏波
(51)Int.Cl.
G06F 16/20(2019.01)
G06F 16/23(2019.01)
审查员 朱琦

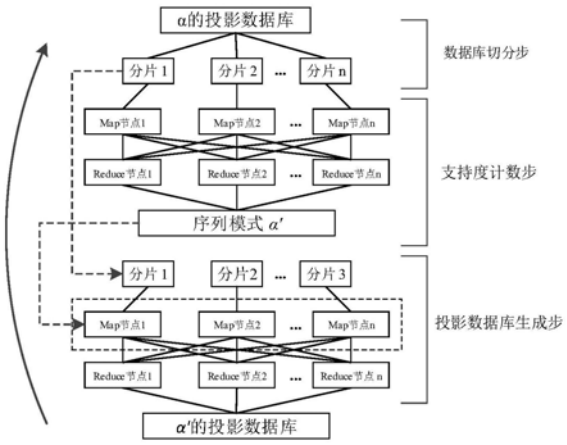
权利要求书1页 说明书12页 附图3页

(54)发明名称

一种基于Spark云计算平台的并行序列模式挖掘方法

(57)摘要

本发明公开了一种基于Spark云计算平台的并行序列模式挖掘方法,针对现有的串行化序列模式挖掘算法在处理海量数据时计算能力低效的问题和现有的基于Hadoop的并行序列模式挖掘算法具有高IO开销和负载不平衡的问题,设计了合理的投影序列数据库切分策略,最大限度的解决了负载不平衡的问题。在此基础上根据MapReduce编程框架的特性,对原始PrefixSpan算法进行了并行化,利用Spark云计算平台的大规模并行计算能力提高了海量数据序列模式挖掘效率。本发明的技术方案具有简单、快速的特点,能够较好地提高序列模式挖掘的效率。



CN 107346331 B

1.一种基于Spark云计算平台的并行序列模式挖掘方法，其特征在于：包括数据库切分、支持度计数和投影数据库生成三步骤，且三步迭代执行，直到没有新的序列模式产生为止；

所述数据库切分，具体实现包括以下子步骤：

步骤1.1：在第一次执行时，将原始数据库切分成相同大小的数据库分片，使每个数据库分片中的包含的序列个数近似相等；将数据库分片从HDFS中导入RDD中，接下来的所有MapReduce任务从RDD中读取数据库分片或生成的序列模式，并将该任务生成的投影数据库或序列模式存入RDD中；

步骤1.2：在后续迭代执行时，将投影数据库切分成相同大小的数据库分片，使每个数据库分片中的包含的序列个数近似相等；将投影数据库分片存入RDD中，接下来的所有MapReduce任务从RDD中读取投影数据库分片或生成的序列模式，并将该任务生成的投影数据库或序列模式存入RDD中；

所述支持度计数，利用一个MapReduce任务发现序列模式；其具体实现包括以下子步骤：

步骤2.1：在第一次执行时，调用第一个flatMap函数从序列数据库片段中读取每条序列，其中序列以<LongWritable偏移量， Text 序列>键值对的形式存储；调用另一个flatMap函数将序列切分为项，产生<项，1>键值对；拥有相同键的键值对被合并传递给Reduce节点，Reduce节点调用ReducebyKey()函数计算<项，1>键值对的支持度，输出支持度大于等于设定的最小支持度的键值对；这些键值对的键即为1-序列模式，值即为该1-序列模式的支持度计数；删除原始序列数据库中的非1-序列模式，形成新的序列数据库，后续的MapReduce任务都基于此新的序列数据库进行操作；

步骤2.2：在后续迭代执行时，每一个Map节点首先调用flatMap函数从投影数据库片段$S_i|_\alpha$中读取每一行后缀序列，然后再调用另一个flatMap函数将后缀序列中的第一项切分出来，将这一项b加入到前缀α后产生<$\alpha+b$， 1>键值对；拥有相同键的键值对被合并传递给Reduce节点；最后每一个Reduce节点调用ReducebyKey()函数计算<$\alpha+b$， 1>键值对的支持度，输出支持度大于等于设定的最小支持度的键值对；

所述投影数据库生成，利用一个MapReduce任务为每个在支持度计数步中产生的序列模式生成相应的投影数据库；其具体实现包括以下子步骤：

步骤3.1：每个Map节点调用flatMap() 函数读取在之前的以α为前缀的投影数据库中的后缀序列；

步骤3.2：每个map函数计算前缀$\alpha\phi$的后缀，以α为前缀的投影数据库中的后缀序列中第一次出现前缀$\alpha\phi$的后缀即为$\alpha\phi$的后缀；其中$\alpha\phi$是以α为前缀的序列模式；

步骤3.3：将Map节点产生的键值对传递给Reduce节点，Reduce节点对这些键值对不做任何的处理，生成最终的投影数据库。

2.根据权利要求1所述的基于Spark云计算平台的并行序列模式挖掘方法，其特征在于：步骤1.1中，数据库分片的数量与集群中Map节点数相同。

一种基于Spark云计算平台的并行序列模式挖掘方法

技术领域

[0001] 本发明属于序列模式挖掘技术领域，特别涉及一种基于Spark云计算平台的并行序列模式挖掘方法。

背景技术

[0002] (1)序列模式挖掘技术

[0003] [文献1]最早提出序列模式挖掘的概念。序列模式挖掘就是挖掘序列数据库中频繁出现的有序事件或子序列。序列模式挖掘作为数据挖掘研究领域中重要的研究内容之一，有着很广泛的应用需求，比如用户购买行为分析、生物序列分析、出租车频繁轨迹模式发现、人类移动行为模式分析。以下是序列模式挖掘中的一些术语的定义。

[0004] 定义1:对于一个集合$I=\{i_k,k=1,2,\cdots,m\}$是一个包含m个不同项的集合，称一个子集$X\subset I$为一个项集。

[0005] 定义2:序列是由多个项集组成的集合，记为$S=\langle s_1,s_2,\cdots,s_n\rangle$，其中$s_i\subset I$.某个具体的序列的长度等于序列包含项的数目。假定某个序列的长度为l，则称此序列是l-序列。

[0006] 定义3:序列数据库由<Sid,S>组成，其中第一列Sid表示序列号，第二列S表示序列的具体组成项集，每行表示一条序列记录。

[0007] 定义4:对于序列S支持度定义为序列S在全局序列数据库中出现的次数。已知最小支持度，如果序列S的支持度不低于最小支持度，那么序列S就是序列模式。长度为l的序列模式称为l-序列模式。

[0008] 定义5:给定两个序列$\alpha=\langle a_1,a_2,..a_n\rangle$，$\beta=\langle b_1,b_2,\cdots b_m\rangle$($m\leq n$)，β被称为α的前缀当且仅当$a_i=b_i\ (i\leq m-1,\ b_m\subset a_m)$，或$a_m-b_m=\Phi$.序列$\gamma=\langle a_m-b_m,a_{m+1},\cdots,a_n\rangle$被称为α相对于β的后缀。

[0009] 定义6:α是序列数据库D中的一个序列模式，α的投影数据库是以α为前缀的所有后缀的集合，记为$S|_\alpha$。

[0010] [文献2]提出了采用冗余候选模式的剪除策略和哈希树来实现候选模式快速访存的GSP算法。[文献3]提出了基于垂直数据表示的SPADE算法。[文献4]提出了基于投影数据库的PrefixSpan算法。这些传统的串行化算法虽然随着数据结构的优化和挖掘机制的改变，在性能上有一定提高，但在面对大规模数据集时算法的处理速度往往达不到人们的要求。直到20世纪初，计算机硬件的急速发展极大的推动了并行序列模式挖掘算法的研究。国内外学者相继提出了各种分布式序列模式挖掘算法。

[0011] [文献5]提出了基于树投影技术的两种不同的并行化算法来解决分布内存并行计算机的序列模式发现问题。[文献6]提出了通过语法序列树减少数据传输量的DMGSP算法。[文献7]提出了快速挖掘全局最大频繁项目集的FMGSP算法。但是由于分布式内存系统或网格计算系统这些并行平台并未提供容错机制，所以在这些并行平台上面实现的并行序列模式挖掘算法不具备容错性。此外，在这些平台上开发并行算法需要程序员具备大量的并行算法开发经验。

[0012] 云计算平台的出现为实现并行算法提供了新的方法和途径，使得高效低成本的从海量数据中进行序列模式挖掘成为可能。由Apache基金会所开发的Hadoop云计算平台由于其开源性、可扩展性、高容错性、使不具备丰富并行算法开发经验的程序员可以在Hadoop平台上轻松的开发并行程序，因此很多学者提出了基于Hadoop平台的并行序列模式挖掘算法。[文献8]提出了基于Hadoop的并行增量序列模式挖掘算法DPSP算法。[文献9]提出了基于Hadoop的并行闭序列挖掘算法-BIDE-MR算法。[文献10]提出了基于Hadoop的SPAMC算法。[文献11]提出了基于Hadoop的并行PrefixSpan算法。[文献12]提出了基于事务分解思想的基于Hadoop的PrefixSpan并行算法。[文献13]提出了基于数据库切分的基于Hadoop的DGSP算法。文献[8][9][10][11]提出的基于迭代式MapReduce任务的并行序列模式挖掘算法需要执行多个需要从HDFS中读取序列数据库的MapReduce任务，会产生很大的IO开销。文献[12][13]提出的基于非迭代式MapReduce任务的并行序列模式挖掘算法并不能有效的将计算任务均匀的分派到各个计算节点，造成了负载不均衡。

[0013] (2)Map-Reduce编程框架

[0014] Map-Reduce是一种编程框架，采用了概念"Map(映射)"和"Reduce(归约)"，用于大规模数据集(大于1TB)的并行运算，在[文献14]中提出。用户只需编写两个称作Map和Reduce的函数即可，系统能够管理Map或Reduce并行任务的执行以及任务之间的协调，并且能够处理上述某个任务失败的情况，并且同时能够保障对硬件故障的容错性。

[0015] 基于Map-Reduce的计算过程如下：

[0016] 1)用户程序中的Map-Reduce库首先将输入文件分成M个数据分片，每个分片的大小一般从16到64MB(用户可以通过可选的参数来控制每个数据片段的大小)，然后Map-Reduce库在机群中创建大量的程序副本。

[0017] 2)这些程序副本有一个特殊的程序-主控程序，副本中其它的程序都是由主控程序分配任务的工作程序。有M个Map任务和R个Reduce任务将被分配，主控程序将一个Map任务或Reduce任务分配给一个空闲的工作程序。

[0018] 3)被分配了Map任务的工作程序读取相关的输入数据片段，从输入的数据片段中解析出键-值(key,value)对，然后把键-值对传递给用户自定义的Map函数，Map函数将产生的中间临时键-值对保存在本地内存缓存中。

[0019] 4)缓存中的键-值对通过分区函数分成R个区域，之后周期性的写入到本地磁盘上。缓存的键-值对在本地磁盘上的存储位置将被回传给主控程序，由主控程序负责把这些存储位置再传给被分配了Reduce任务的工作程序。

[0020] 5)当被分配了Reduce任务的工作程序接收到主控程序发来的数据存储位置信息后，使用远程过程调用(remote procedure calls)从被分配了Map任务的工作程序所在主机的磁盘上读取这些缓存数据。当被分配了Reduce任务的工作程序读取了所有的中间数据后，通过对键进行排序后使得具有相同键的数据聚合在一起。由于许多不同的键会映射到相同的Reduce任务上，因此必须进行排序。如果中间数据太大无法在内存中完成排序，那么就要在外部进行排序。

[0021] 6)被分配了Reduce任务的工作程序遍历排序后的中间数据，对于每一个唯一的中间键-值对，被分配了Reduce任务的工作程序将这个键和它相关的中间值的集合传递给用户自定义的Reduce函数。Reduce函数的输出被追加到所属分区的输出文件。

[0022] 7)当所有的Map和Reduce任务都完成之后,主控程序唤醒用户程序。在这个时候,在用户程序里的对Map-Reduce调用才返回。

[0023] (3)Spark云计算平台

[0024] Spark是由UC Berkeley AMP实验室开发的开元通用并行云计算平台,Spark基于MapReduce思想实现的分布式计算,拥有Hadoop MapReduce所具有的优点;但是不同地方是运算中间输出结果能存储在内存中,从而不在需要读写分布式文件系统(HDFS),因此Spark能更好的运行数据挖掘与机器学习等需要迭代的MapReduce算法。Spark启用了内存分布数据集,它可以提供交互式查询,除此之外还可以将数据集缓存在内存中,提高数据集读写速率。实现计算过程中的数据集的重用,优化迭代工作负载。Spark底层可使用多种分布式文件系统如HDFS文件系统存放数据,不过更多的是与资源调度平台Mesos和YARN一起合作出现。

[0025] RDD(弹性分布式数据集)是Spark的核心,RDD是分布于各个计算节点存储于内存中的数据对象集合,RDD允许用户在执行多个查询时显式地将工作集缓存在内存中,后续的查询能够重用工作集,这极大地提升了查询速度。RDD分布在多个节点上,并可以对其进行并行处理。RDD是可扩展、弹性的,在运算过程中,内存小于RDD时,可以转存到磁盘上,确保内存足够继续运算。RDD是已被分区、只读的、不可变的并能够被并行操作的数据集合,只能通过在其它RDD执行确定的转换操作(如map、join、filter和group by)而创建,然而这些限制使得实现容错的开销很低。与分布式共享内存系统需要付出高昂代价的检查点和回滚不同,RDD通过Lineage来重建丢失的分区:一个RDD中包含了如何从其它RDD衍生所必需的相关信息,从而不需要检查点操作就可以重构丢失的数据分区。尽管RDD不是一个通用的共享内存抽象,但却具备了良好的描述能力、可伸缩性和可靠性,并能够广泛适用于数据并行类应用。

[0026] 有关文献:

[0027] [文献1]Agrawal R,Srikant R.Mining sequential patterns:The 11th International Conference on Data Engineering[C].Taipei:IEEE Computer Society,1995:3-141.

[0028] [文献2]Srikant R,Agrawal R.Mining sequential pattern:Generations and performance improvement[C]//proceedings of the 5th International Conference ExtendingDatabase Technology.Avignon:Lecture Notes in Computer Science,1996:.3-17.

[0029] [文献3]Zaki M.SPADE:An efficient algorithm for mining frequent sequences[J].Machine Learning,2001.41(2):31-60.

[0030] [文献4]Pei J,Han J,Pinto H.PrefixSpan mining sequential patterns efficiently by prefix-projected pattern growth[C]//proceedings of the 17th International Conference on Data Engineering.Washington,IEEE Transactions on Data Engineering,2004.16(1):1424-1440.

[0031] [文献5]Gurainikv,Gargn,Vipink.Parallel tree Projection algorithm for sequence mining[C]//proceedings of the 7th International European Conference on Parallel Processing.London,2001:310-320.

[0032] [文献6]龚振志,胡孔法,达庆利,张长海.DMGSP:一种快速分布式全局序列模式挖掘算法[J].东南大学学报,2007.16(04):574-579.

[0033] [文献7]Zhang Changhai,Hu Kongfa,Liu Haidong.FMGSP:an efficient method of mining global sequential patterns[C].//proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery.Los Alanitos IEEE Computer Society.2007:761-765.

[0034] [文献8]J.Huang,S.Lin,M.Chen,“DPSP:Distributed Progressive Sequential Pattern Mining on the Cloud,”Lecture Notes in Computer Science,pp.27-34,2010.

[0035] [文献9]D.Yu,W.Wu,S.Zheng,Z.Zhu,“BIDE-Based Parallel Mining of Frequent Closed Sequences with MapReduce,”In:Proceedings of the 12th International Conference on Algorithms and Architecturesfor Parallel Processing,pp.177-186 2012.

[0036] [文献10]Chun-Chieh Chen,Chi-Yao Tseng,Chi-Yao Tseng,“Highly Scalable Sequential Pattern Mining Based on MapReduce Model on the Cloud,”In2013IEEE International Congress on Big Data,pp.310-317,2013.

[0037] [文献11]P.N.Sabrina,“Miltiple MapReduce and Derivative projected database:new approach for supporting prefixspan scalability,”IEEE,pp.148-153,Nov.2015.

[0038] [文献12]X.Wang,“Parallel sequential pattern mining by transcationdecompostion,”IEEE Fuzzy Systems and Knowledge Discovery(FSKD),2010Seventh International Conference on,vol.4,pp.1746-1750.

[0039] [文献13]X.Yu,J.Liu,C.Ma,B.Li,“A MapReduce reinforeced distributed sequential pattern mining algorithm,”Algorithms and Architectures for Parallel Processing,vol.9529,pp.183-197,Dec.2015.

[0040] [文献14]Jeffrey Dean and Sanjay Ghemawat.Map-Reduce:Simplified data processing on large Cluster[C]//proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation.New York:ACM Press,2004:137-149.

发明内容

[0041] 为了解决串行化序列模式挖掘算法在处理海量数据时计算能力低效的问题和现有的基于Hadoop的并行序列模式挖掘算法具有高IO开销和负载不平衡的问题,本发明提供了一种基于Spark云计算平台的并行序列模式挖掘方法。

[0042] 本发明所采用的技术方案是:一种基于Spark云计算平台的并行序列模式挖掘方法,其特征在于:包括数据库切分、支持度计数和投影数据库生成三步骤,且三步迭代执行,直到没有新的序列模式产生为止;

[0043] 所述数据库切分,具体实现包括以下子步骤:

[0044] 步骤1.1:在第一次执行时,将原始数据库切分成相同大小的数据库分片,使每个数据库分片中的包含的序列个数近似相等;将数据库分片从HDFS中导入RDD中,接下来的所

有MapReduce任务从RDD中读取数据库分片或生成的序列模式，并将该任务生成的投影数据库或序列模式存入RDD中；

[0045] 步骤1.2：在后续迭代执行时，将投影数据库切分成相同大小的数据库分片，使每个数据库分片中的包含的序列个数近似相等；将投影数据库分片存入RDD中，接下来的所有MapReduce任务从RDD中读取投影数据库分片或生成的序列模式，并将该任务生成的投影数据库或序列模式存入RDD中；

[0046] 所述支持度计数，其具体实现包括以下子步骤：

[0047] 步骤2.1：在第一次执行时，调用第一个flatMap函数从序列数据库片段中读取每条序列，其中序列以<LongWritable偏移量，Text序列>键值对的形式存储；调用另一个flatMap函数将序列切分为项，产生<项，1>键值对；拥有相同键的键值对被合并传递给Reduce节点，Reduce节点调用ReducebyKey()函数计算<项，1>键值对的支持度，输出支持度大于等于设定的最小支持度的键值对；这些键值对的键即为1-序列模式，值即为该1-序列模式的支持度计数；删除原始序列数据库中的非1-序列模式，形成新的序列数据库，后续的MapReduce任务都基于此新的序列数据库进行操作；

[0048] 步骤2.2：在后续迭代执行时，每一个Map节点首先调用flatMap函数从投影数据库片段$S_i|_\alpha$中读取每一行后缀序列，然后再调用另一个flatMap函数将后缀序列中的第一项切分出来，将这一项b加入到前缀α后产生<α+b，1>键值对；拥有相同键的键值对被合并传递给Reduce节点；最后每一个Reduce节点调用ReducebyKey()函数计算<α+b，1>键值对的支持度，输出支持度大于等于设定的最小支持度的键值对；

[0049] 所述投影数据库生成，利用一个MapReduce任务为每个在支持度计数步中产生的序列模式生成相应的投影数据库；其具体实现包括以下子步骤：

[0050] 步骤3.1：每个Map节点调用flatMap()函数读取在之前的以α为前缀的投影数据库中的后缀序列；

[0051] 步骤3.2：每个map函数计算前缀α′的后缀，以α为前缀的投影数据库中的后缀序列中第一次出现前缀α′的后缀即为α′的后缀；其中α′是以α为前缀的序列模式；

[0052] 步骤3.3：将Map节点产生的键值对传递给Reduce节点，Reduce节点对这些键值对不做任何的处理，生成最终的投影数据库。

[0053] 本发明设计了合理的投影序列数据库切分策略，最大限度的解决了负载不平衡的问题。在此基础上根据MapReduce编程框架的特性，对原始PrefixSpan算法进行了并行化，利用Spark云计算平台的大规模并行计算能力提高了海量数据序列模式挖掘效率。本发明的技术方案具有简单、快速的特点，能够较好地提高序列模式挖掘的效率。

附图说明

[0054] 图1为本发明实施例的流程图；

[0055] 图2为本发明实施例第一次执行支持度计数步示意图；

[0056] 图3为本发明实施例第一次执行投影数据库生成步示意图；

[0057] 图4为本发明实施例第二次执行支持度计数步示意图；

[0058] 图5为本发明实施例第二次执行投影数据库生成步示意图；

[0059] 图6为本发明实施例第三次执行支持度计数步示意图。

具体实施方式

[0060] 为了便于本领域普通技术人员理解和实施本发明，下面结合附图及实施例对本发明作进一步的详细描述，应当理解，此处所描述的实施示例仅用于说明和解释本发明，并不用于限定本发明。

[0061] 本发明设计的基于Spark云计算平台的序列模式挖掘算法的流程见附图1，所有步骤可由本领域技术人员采用计算机软件技术实现流程自动运行。该方法主要包括三个步骤：数据库切分步、支持度计数步和投影数据库生成步。这三步迭代执行，直到没有新的序列模式产生。

[0062] 实施例具体实现过程如下：

[0063] 步骤1，数据库切分；

[0064] 为了实现较好的负载平衡，在第一次执行该步时，将原始数据库切分成相同大小的数据库分片(分片数最好与集群中Map节点数相同)，使每个数据库分片中包含的序列近乎相等。为了减小IO开销并且充分的利用集群内存，将这些数据库分片从HDFS中导入RDD中，接下来的所有MapReduce任务从RDD中读取数据库分片、投影数据库分片或生成的序列模式，并将该任务生成的投影数据库或序列模式存入RDD中。

[0065] 本实施例在第一次执行数据库切分步时，设定将原始序列数据库划分为n＝3个数据库分片。

[0066] 原始序列数据库内容如下表1：

[0067] 表1

[0068]

序列号	序列
S_1	〈(c d) (e f g)〉
S_2	〈h〉
S_3	〈(a b)a c〉
S_4	〈c g〉
S_5	〈a c g h〉
S_6	〈g a〉

[0069] 划分得到的数据库分片1、2、3分别如下表2、3、4：

[0070] 表2

[0071]

序列号	序列

[0072]

S_1	<(c d) (e f g)>
S_2	<h>

[0073] 表3

[0074]

序列号	序列

S_3	<(a b)a c>
S_4	<c g>

[0075] 表4

[0076]

序列号	序列
S_5	<a c g h>
S_6	<g a>

[0077] 步骤2,支持度计数;

[0078] 在第一次执行支持度计数步时,该步首先调用第一个flatMap函数从序列数据库片段中读取每条序列,其中序列以<LongWritable offset,Text sequence>键值对的形式存储。随后调用另一个flatMap函数将序列切分为项,产生<项,1>键值对。拥有相同键的键值对被合并传递给Reduce节点,Reduce节点调用ReducebyKey()函数计算<项,1>键值对的支持度,输出支持度大于等于设定的最小支持度的键值对。这些键值对的键即为1-序列模式,值即为该1-序列模式的支持度计数。然后删除原始序列数据库中的非1-序列模式,形成新的序列数据库,后续的MapReduce任务都基于这个新的序列数据库进行操作。

[0079] 实施例设定最小支持度为2,Spark集群中包含3个Map节点和2个Reduce节点,第一次执行支持度计数步的具体执行过程参见图2,Map节点对数据库分片1产生键值对结果如下表5:

[0080] 表5

[0081]

输出结果
<*c*, 1>
<*d*, 1>
<*e*, 1>
<*f*, 1>
<*g*, 1>

[0082]

<*h*, 1>

[0083] Map节点对数据库分片2产生键值对结果如下表6:

[0084] 表6

[0085]

输出结果
<a,1>
<b,1>
<a,1>
<c,1>
<c,1>
<h,1>

[0086] Map节点对数据库分片3产生键值对结果如下表7：

[0087] 表7

[0088]

输出结果
<a,1>
<c,1>
<g,1>
<h,1>
<g,1>
<a,1>

[0089] Reduce节点合并具有相同的键的键值对，输出支持度大于等于2的键值对的结果如下表8：

[0090] 表8

[0091]

序列模式	支持度
a	3
c	4
g	4
h	2

[0092] 删除原始序列数据库中非1-序列模式后得到新的数据库如下表9：

[0093] 表9

[0094]

序列号	序列
S_1	<cg>
S_2	<h>
S_3	<a a c>
S_4	<c g>
S_5	<a c g h>
S_6	<g a>

[0095] 步骤3，投影数据库生成；

[0096] 在这一步中，利用一个MapReduce任务为每个在支持度计数步中产生的序列模式生成相应的投影数据库。需要说明的是，一个前缀(如前缀<a c>)的后缀是之前的前缀(如前缀<a>)的后缀的子序列。所以对前缀α′的构造无需扫描原始的序列数据库，只需扫描α的投影数据库即可，其中α′是以α为前缀的序列模式。在这个MapReduce任务中，每个Map节点调用flatMap()函数读取在之前的以α为前缀的投影数据库中的后缀序列。然后每个map函数计算前缀α′的后缀，以α为前缀的投影数据库中的后缀序列中第一次出现前缀α′的后缀即为α′的后缀。然后将Map节点产生的键值对传递给Reduce节点。Reduce节点对这些键值对不做任何的处理，生成最终的投影数据库。

[0097] 实施例，第一次执行投影数据库生成步的具体执行过程参见图3，Map节点对数据

库分片1产生键值对结果如下表10：

[0098] 表10

[0099]

输出结果
<c,g>

[0100] Map节点对数据库分片2产生键值对结果如下表11：

[0101] 表11

[0102]

输出结果
<a,a c>
<a,c>
<c,g>

[0103] Map节点对数据库分片3产生键值对结果如下表12：

[0104] 表12

[0105]

输出结果
<a,c g h>
<c,g h>
<g,h>
<g,a>

[0106] Reduce节点对这些键值对不做任何的处理，生成最终的投影数据库如下表13：

[0107] 表13

[0108]

前缀	后缀
c	<g>
a	<a c>
a	<c>
c	<g>
a	<c g h>
c	<g h>
g	<h>
g	<a>

[0109] 在迭代执行数据库切分步时，为了实现较好的负载平衡，将投影数据库切分成相同大小的数据库分片(分片数最好与集群中Map节点数相同)，使每个数据库分片中的包含的序列近乎相等。

[0110] 在第二次执行数据库切分步时，本实施例设定将表13的投影序列数据库划分为n＝3个数据库分片。

[0111] 划分得到的数据库分片1、2、3分别如下表14、15、16：

[0112] 表14

[0113]

前缀	后缀
c	<g>
a	<a c>
a	<c>

[0114] 表15

[0115]

前缀	后缀
c	<g>
a	<c g h>
c	<g h>

[0116] 表16

[0117]

前缀	后缀
g	<h>
g	<a>

[0118] 在迭代执行支持度计数步时,每一个Map节点首先调用flatMap函数从投影数据库片段$S_i|_\alpha$中读取每一行后缀序列,然后再调用另一个flatMap函数将后缀序列中的第一项切分出来,然后将这一项b加入到前缀α后产生<α+b,1>键值对。然后拥有相同键的键值对被合并传递给Reduce节点。最后每一个Reduce节点调用ReducebyKey()函数计算<α+b,1>键值对的支持度,输出支持度大于等于设定的最小支持度的键值对。

[0119] 本实施例第二次执行支持度计数步的具体执行过程参见图4,Map节点对数据库分片1产生键值对结果如下表17:

[0120] 表17

[0121]

输出结果
<c,g>
<a,a>
<a,c>

[0122] Map节点对数据库分片2产生键值对结果如下表18:

[0123] 表18

[0124]

输出结果
<c,g>
<a,c>
<c,g>

[0125] Map节点对数据库分片3产生键值对结果如下表19:

[0126] 表19

[0127]

输出结果

<g,h>
<g,a>

[0128] Reduce节点合并具有相同的键的键值对,输出支持度大于等于2的键值对的结果如下表20:

[0129] 表20

[0130]

序列模式	支持度
a c	2
c g	3

[0131] 在迭代执行投影数据库生成步骤时,利用一个MapReduce任务为每个在支持度计数步中产生的序列模式生成相应的投影数据库。需要说明的是,一个前缀(如前缀<a c>)的后缀是之前的前缀(如前缀<a>)的后缀的子序列。所以对前缀α′的构造无需扫描原始的序列数据库,只需扫描α的投影数据库即可,其中α′是以α为前缀的序列模式。在这个MapReduce任务中,每个Map节点调用flatMap()函数读取在之前的以α为前缀的投影数据库中的后缀序列。然后每个map函数计算前缀α′的后缀,以α为前缀的投影数据库中的后缀序列中第一次出现前缀α′的后缀即为α′的后缀。然后将Map节点产生的键值对传递给Reduce节点。Reduce节点对这些键值对不做任何的处理,生成最终的投影数据库。

[0132] 本实施例第二次执行投影数据库生成步的具体执行过程参见图5,Map节点对数据库分片1产生键值对结果如下表21:

[0133] 表21

[0134]

输出结果
<a c,g h>

[0135] Map节点对数据库分片2产生键值对结果如下表22:

[0136] 表22

[0137]

输出结果
<c g,h>

[0138] Map节点对数据库分片3没有产生键值对。

[0139] Reduce节点对这些键值对不做任何的处理,生成最终的投影数据库如下表23:

[0140] 表23

[0141]

前缀	后缀
a c	<g h>
c g	<h>

[0142] 在第三次执行数据库切分步时,为了实现较好的负载平衡,将投影数据库切分成相同大小的数据库分片(分片数最好与集群中Map节点数相同),使每个数据库分片中的包含的序列近乎相等。在第三次执行数据库切分步时,本实施例设定将表23的投影序列数据库划分为n=2个数据库分片。

[0143] 划分得到的数据库分片1、2分别如下表24、25：

[0144] 表24

[0145]

前缀	后缀
a c	<g h>

[0146] 表25

[0147]

前缀	后缀
c g	<h>

[0148] 在第三次执行支持度计数步时，每一个Map节点首先调用flatMap函数从投影数据库片段$S_i|_\alpha$中读取每一行序列(后缀)，然后再调用另一个flatMap函数将序列中的每一项切分出来，然后将每一项b加入到前缀α后产生<α+b,1>键值对。然后拥有相同键的键值对被合并传递给Reduce节点。最后每一个Reduce节点调用ReducebyKey()函数计算<α+b,1>键值对的支持度，输出支持度大于等于设定的最小支持度的键值对。

[0149] 本实施例第三次执行支持度计数步的具体执行过程参见图6，Map节点对数据库分片1产生键值对结果如下表26：

[0150] 表26

[0151]

输出结果
<ac,g>

[0152] Map节点对数据库分片2产生键值对结果如下表27：

[0153] 表27

[0154]

输出结果
<c g,h>

[0155] Reduce节点合并具有相同的键的键值对，输出支持度大于等于2的键值对，发现所有的Reduce节点没有输出，因此程序终止。

[0156] 本文中所描述的具体实施例仅仅是对本发明精神作举例说明。本发明所属技术领域的技术人员可以对所描述的具体实施例做各种各样的修改或补充或采用类似的方式替代，但并不会偏离本发明的精神或者超越所附权利要求书所定义的范围。

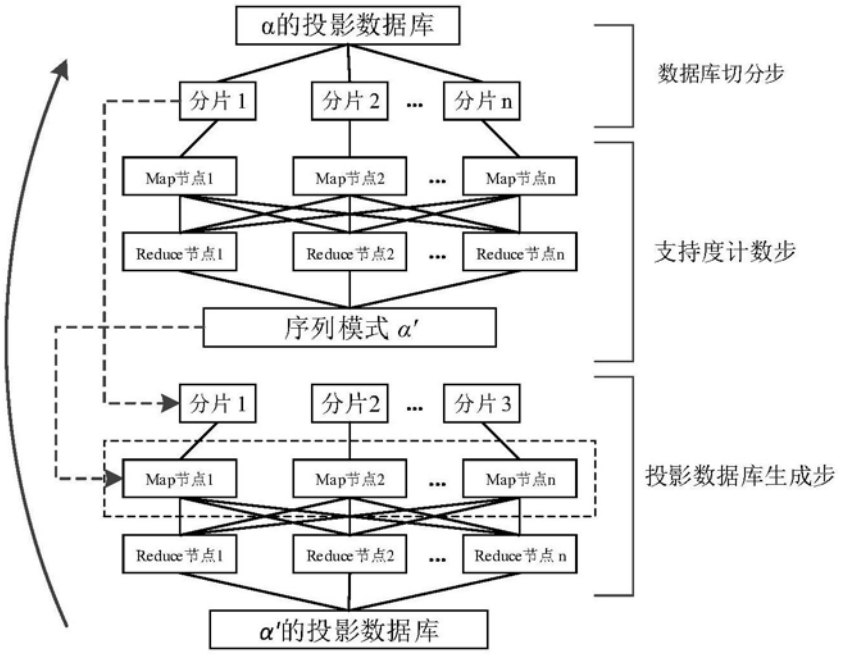

α的投影数据库
分片 1
分片 2
...
分片 n
数据库切分步
Map节点1
Map节点2
...
Map节点n
Reduce节点1
Reduce节点2
...
Reduce节点n
支持度计数步
序列模式 α′
分片 1
分片2
...
分片 3
Map节点1
Map节点2
...
Map节点n
Reduce节点1
Reduce节点2
...
Reduce节点 n
投影数据库生成步
α′的投影数据库

图1

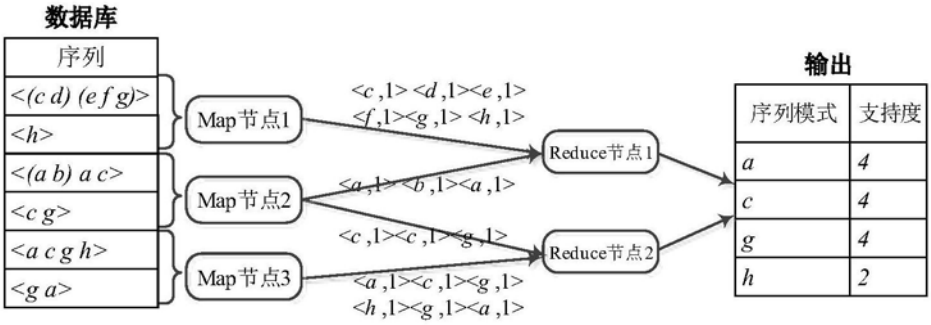

数据库
序列
<(c d) (e f g)>
<h>
<(a b) a c>
<c g>
<a c g h>
<g a>
Map节点1
Map节点2
Map节点3
<c ,1> <d ,1><e ,1>
<f ,1><g ,1> <h ,1>
<a ,1> <b ,1><a ,1>
<c ,1><c ,1><g ,1>
<a ,1><c ,1><g ,1>
<h ,1><g ,1><a ,1>
Reduce节点1
Reduce节点2
输出
序列模式
支持度
a
4
c
4
g
4
h
2

图2

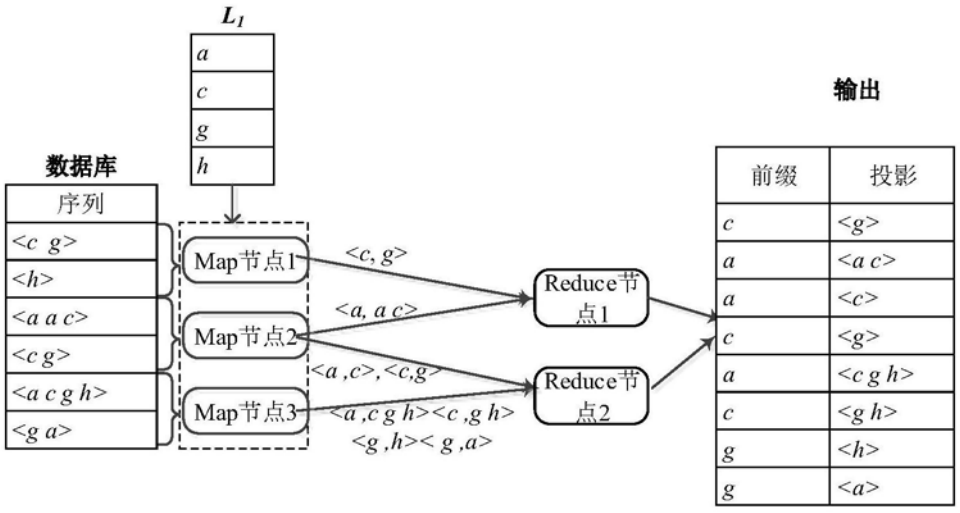

L_1
a
c
g
h
数据库
序列
<c g>
<h>
<a a c>
<c g>
<a c g h>
<g a>
Map节点1
Map节点2
Map节点3
<c, g>
<a, a c>
<a ,c>,<c,g>
<a ,c g h><c ,g h>
<g ,h>< g ,a>
Reduce节点1
Reduce节点2
输出
前缀
投影
c
<g>
a
<a c>
a
<c>
c
<g>
a
<c g h>
c
<g h>
g
<h>
g
<a>

图3

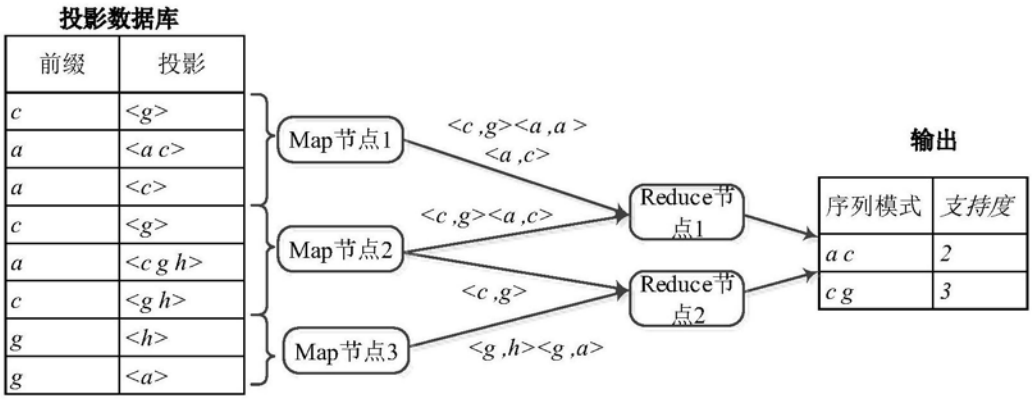

投影数据库
前缀
投影
c
<g>
a
<a c>
a
<c>
c
<g>
a
<c g h>
c
<g h>
g
<h>
g
<a>
Map节点1
Map节点2
Map节点3
<c ,g><a ,a ><a ,c>
<c ,g><a ,c>
<c ,g>
<g ,h><g ,a>
Reduce节点1
Reduce节点2
输出
序列模式
支持度
a c
2
c g
3

图4

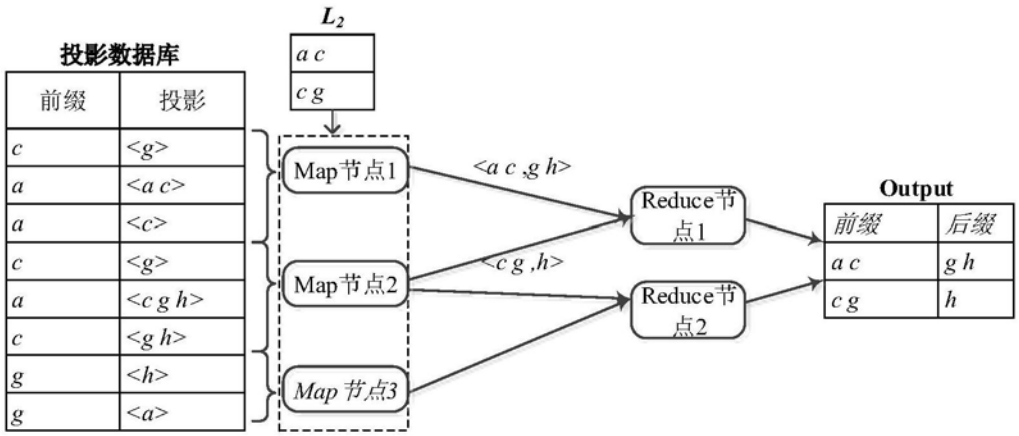

L_2
a c
c g
投影数据库
前缀
投影
c
<g>
a
<a c>
a
<c>
c
<g>
a
<c g h>
c
<g h>
g
<h>
g
<a>
Map节点1
Map节点2
Map 节点3
<a c ,g h>
<c g ,h>
Reduce节点1
Reduce节点2
Output
前缀
后缀
a c
g h
c g
h

图5

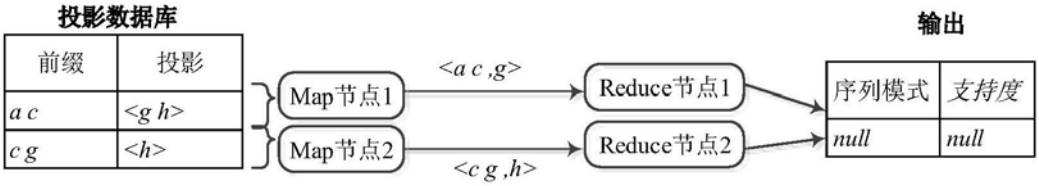

投影数据库
前缀
投影
a c
<g h>
c g
<h>
Map节点1
Map节点2
<a c ,g>
<c g ,h>
Reduce节点1
Reduce节点2
输出
序列模式
支持度
null
null

图6