基于MapReduce的数据倾斜连接算法

梁俊杰 何利民
(湖北大学计算机与信息工程学院 武汉 430062)

摘 要 连接操作是大规模数据集在数据分析应用中最常用的操作,针对MapReduce自身不能有效地处理数据倾斜情况下的连接操作,提出了基于MapReduce的频次分类连接算法。根据数据在连接数据集中出现的频率将整个数据集分为3类,对倾斜数据利用分区算法和广播算法实现数据重分布,以消除数据倾斜的影响;对非倾斜数据采用Hash算法实现数据重分布。重分布后的数据在单节点内即可完成数据连接操作,避免了MapReduce框架下连接操作的跨节点传输代价;同时有效地均衡了MapReduce各节点的任务负载,从而提高了数据倾斜状态下连接操作的效率。通过与传统连接算法的对比,证明了所提算法的有效性和实用性。

关键词 数据倾斜,MapReduce,连接算法,负载均衡
中图法分类号 TP311.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.9.005

Join Algorithm in Skewed Datasets Based on MapReduce

LIANG Jun-jie HE Li-min
(School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China)

Abstract Join operation is the most common operation in data analysis applications with large-scale datasets, and MapReduce can not support join operation perfectly in handling data skew problem. MapReduce frequecncy classified join algorithm was proposed, and datasets were classified into three categories according to the appeared data frequency. Data redistribution applying partitioning algorithm and broadcast algorithms eliminate the impact of skewed data. And data redistribution is realized by using hash algorithm for the non-skew data. Join operation can be completed in a single node, avoiding the cost of communications across the nodes under the MapReduce for the redistributed data, and balancing the workload of each node effectively, thereby improves the efficiency of join operations in skewed data. The effectiveness and practicality of the algorithms are proved by the comparison with traditional algorithms.

Keywords Data skew, MapReduce, Join algorithm, Load balancing

1 引言

在MapReduce计算模型[1]下容易产生倾斜问题,文献[2]将MapReduce的倾斜分为map端的倾斜和reduce端的倾斜。map端的倾斜主要是因为原始数据分布不均匀,造成一个或某一些map任务拖慢了整个执行效率。reduce端倾斜主要是分区倾斜,分区倾斜是指map按照默认的哈希函数分区容易造成某个reducer任务的数据处理量要远多于其他reducer任务的数据处理量,使得整体运行时间取决于执行时间最长的reducer任务,导致运行效率低下。

最近很多学者提出了基于MapReduce的数据倾斜连接算法。文献[3-5]利用CPU和I/O建立连接算法的代价模型;文献[6]利用智能网络感知算法,在reducers间重新动态分配记录,极大优化了执行效率与网络传输;文献[7,8]提出的数据倾斜连接算法使得每个节点负载均衡且网络传输量最小;文献[9]借鉴传统的倾斜连接算法[10]利用域分区(Range Partition)方法优化两表连接。上述算法主要基于连接过程的CPU、I/O、网络传输代价对连接算法进行优化,域分区方法对数据的分布特征考虑不精准。如何对数据分布进行准确评估,建立适应的优化机制以消除连接过程中数据倾斜的影响来实现负载均衡是很重要的研究课题。

本文针对在MapReduce环境下可能出现的数据倾斜情况进行研究,提出了基于MapReduce的频次分类连接算法Frequecncy Classified Join(FC-Join),以解决数据倾斜情况下的等值连接问题。

2 相关工作

大数据处理中,数据分布和信息统计的评估工作十分重要。直方图可以很好对一个数据集内的数据分布进行评估。例如,在连接操作中,需要计算表之间连接属性的倾斜度,从而生成最优的连接方案。

在MapReduce计算框架中,mapper任务输出的中间结果是一系列(K,V)键值对的集合,这些(K,V)按照键K被shuffle到不同的组中,每一组中所有记录的键都相同。Ma-

到稿日期:2015-08-19 返修日期:2015-10-29 本文受湖北省自然科学基金重点项目(2015CFA067,2013CFA115),湖北省教育厅科研项目计划(D20151001),武汉市科技攻关计划项目(2013012401010851)资助。

梁俊杰(1974-),女,副教授,主要研究方向为大数据、数据库、云计算,E-mail:ljjhubu@163.com;何利民(1989-),男,硕士生,主要研究方向为大数据分析,E-mail:helmstudy@163.com。

pReduce 的默认分区函数只能保证每个 reducer 任务处理的分组数目相同，无法保证每个 reducer 任务处理的记录总数相当，由此导致 reducer 负载倾斜。

本文假设关系 R 和 S 进行连接 $R \bowtie S$，关系 R 和 S 切分成数据块(blocks)存储在 Hadoop 的 HDFS 上。相关符号如表 1 所列。

表 1　相关符号描述

符号	具体含义
$\|T\|$	T 的数据块数
$\overline{T}$	T 中实际产生连接的记录集合
T_i^{map}	mapper 任务处理的数据子集
T_i^{red}	reducer 任务处理的数据子集
$\overline{T}_i^{map}$	mapper 任务处理的连接记录子集
$Hist(T_i^{map})$	T_i^{map} 的频次直方图
Hist(T(v))	键值 v 对应的频次 n_v
HistIndex($R \bowtie S$)	关系 R 和 S 中的连接频次索引值
$c_{r/w}$	HDFS 上数据页的读写成本
c_{comm}	数据页的传输成本
t_s^i	在节点 i 上的哈希表中的搜索成本
t_h^i	向节点 i 上的哈希表中插入记录的成本
mappers	mapper 任务数
reducers	reducer 任务数

3　基于 MapReduce 的数据倾斜连接算法

在现实世界，根据 Zipf 原则，80%的记录的键值占总键值的比例只有 20%。将每个 reducer 的负载表示为该 reducer 处理的(K,V)键值对数目。hadoop 默认的散列函数 $f(k)=hash(k)\%r$，其中 r 代表 reduce 任务的个数，该方法只能保证每个 reducer 的分组数目相同，但很难保证每个 reducer 分配到的(K,V)键值对数目均衡。文献[11]指出这是因为 Hadoop 本身无法感知 mapper 端输出数据的分布情况，导致 reducer 的负载不均衡，影响连接执行的效率。例如，文献[12,13]提出了两种根据采样结果来确定划分函数的方法，从而保证 reducer 负载均衡。如果能够对 mapper 任务输出的中间结果进行统计分析，确定数据分布情况，从而设计适应的分区函数和数据分发机制，就能确保每个 reducer 的负载均衡。

本文提出的 FC-Join 算法包括两对 map-reduce 作业：第一对 map-reduce 作业完成键值 key 的频次统计与分类，创建 key 的频次分类索引；第二对 map-reduce 作业，根据频次分类索引执行不同的数据分发策略，执行连接操作产生连接输出结果。

3.1　基于直方图的数据分类

假设 $R(X,Y) \bowtie S(X,Z)$，连接属性是 X。基于直方图的数据分类包括两个阶段：第一阶段，计算每个 mapper 任务处理的数据集中每个连接属性值出现的频次；第二阶段，计算关系 R 和 S 中每个连接属性值出现的总频次 $Hist(R)(K)$ 与 $Hist(S)(K)$，利用直方图与定义 1，将关系 R 和 S 按照连接属性值各分为 3 类，标记为 $R=R_{par} \cup R_{rep} \cup R_{hash}$ 和 $S=S_{par} \cup S_{rep} \cup S_{hash}$。

定义 1　关系 R 和 S 的数据分类

关系 R 中的记录：

R_{par} 类：$Hist(R)(K) \geqslant f_0 \& Hist(R)(K) \geqslant Hist(S)(K)$，$R_{par}$ 类的索引值为 2，记为：$F(K)=2$。

R_{rep} 类：$Hist(S)(K) \geqslant f_0 \& Hist(R)(K) \leqslant Hist(S)(K)$，$R_{rep}$ 类的索引值为 1，记为：$F(K)=1$。

R_{hash} 类：$Hist(R)(K) < f_0 \& Hist(S)(K) < f_0$，$R_{hash}$ 类的索引值为 0，记为：$F(K)=0$。

关系 S 中的记录：

S_{par} 类：$Hist(S)(K) \geqslant f_0 \& Hist(S)(K) \geqslant Hist(R)(K)$，$S_{par}$ 类的索引值为 2，记为：$F(K)=2$。

S_{rep} 类：$Hist(R)(K) \geqslant f_0 \& Hist(S)(K) \leqslant Hist(R)(K)$，$S_{rep}$ 类的索引值为 1，记为：$F(K)=1$。

S_{hash} 类：$Hist(S)(K) < f_0 \& Hist(R)(K) < f_0$，$S_{hash}$ 类的索引值为 0，记为：$F(K)=0$。

具体分类过程如图 1 所示。

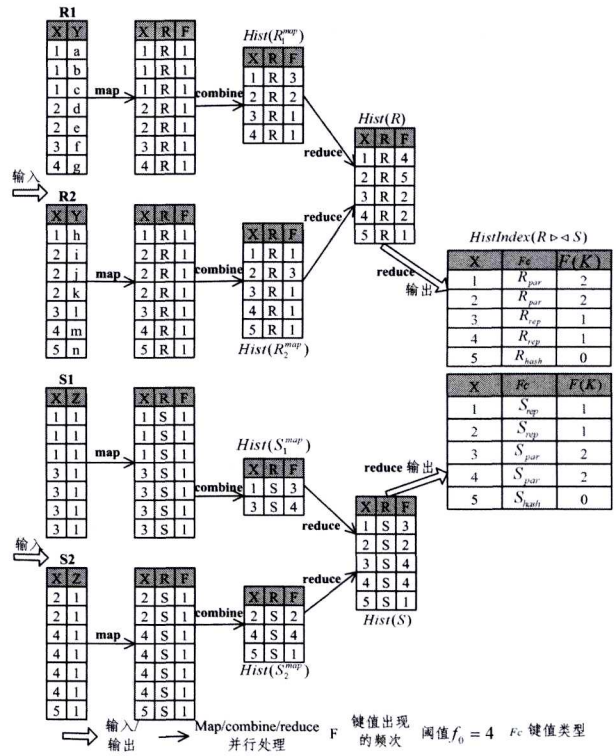

图 1　关系 R 和 S 的数据分类算法过程

算法 1　数据分类算法

1. Map 阶段：/ * 读取关系 T 的记录产生带标签的频次直方图 * /

1)每个 mapper 任务从 HDFS 上读取子关系 R_i^{map} 和 S_i^{map} 的记录。

2)从输入关系的记录中抽取连接属性值 join_key。

3)获取标识源输入关系的标签 tag。

4)发送频次为 1 带标签的键值对((join_key,tag),1)。

Combine 阶段：/ * 计算关系 R_i^{map} 和 S_i^{map} 中 join_key 的频次 * /

5)每个 combiner 计算带标签的键 K(join_key,tag)在 Map 阶段出现的总频次。

Partition 阶段：/ * 计算 mapper 任务输出的目的 reducer * /

6)对 Map 阶段输出的键 K(join_key,tag)，通过哈希 join_key 计算目的 reducer。

2. Reduce 阶段：/ * 合并 shuffle 过程的记录集并建立频次直方图索引 * /

1)计算同时存在于关系 R 和 S 中的 join_key 的总频次。

2)依据每个 join_key 的频次进行分类，产生 R 和 S 表的频次索引值 F(K)。

3)输出带频次索引的键值对(join_key,F(K))。

3.2　基于数据分类的连接算法

3.2.1　数据分发机制

根据频次直方图索引 $HistIndex(R \bowtie S)$，对 3 类数据

采用不同的分发机制。其中 R_{par} 和 S_{par} 类数据采用随机分区分发，R_{rep} 和 S_{rep} 类数据采用广播复制，R_{hash} 和 S_{hash} 类数据采用哈希分区，并将它们分散存储到不同 Slave 上的 HDFS 文件系统中。对于 R_{par} 和 S_{par} 类数据随机分发可以保证每个 reducer 任务的数据处理量均衡，提高 R_{par} 和 S_{par} 类数据连接操作的并行性。对于 R_{rep} 和 S_{rep} 类数据广播复制，可以增加连接的并行度。R_{hash} 和 S_{hash} 类数据对 reducer 任务的负载倾斜没影响，采用默认的哈希分区。数据分发存储如表 2 所列。

表 2　倾斜数据分类分发 Slave 存储表

Slave \ Relation	R			S		
	Rpar	Rrep	Rhash	Spar	Srep	Shash
Slave1	Rpar1	Rrep		Spar1	Srep	
Slave2	Rpar2	Rrep		Spar2	Srep	
Slave3	Rpar3	Rrep	Rhash	Spar3	Srep	Shash

表 2 中 $R_{par}=R_{par1}\cup R_{par2}\cup R_{par3}$，$S_{par}=S_{par1}\cup S_{par2}\cup S_{par3}$，其中 R_{par1} 和 S_{par1} 存储在 $Slave_1$ 上，R_{par2} 和 S_{par2} 存储在 $Slave_2$ 上，R_{par3} 和 S_{par3} 存储在 $Slave_3$ 上，R_{rep} 和 S_{rep} 同时存储在 $Slave_1$、$Slave_2$、$Slave_3$ 上，R_{hash} 和 S_{hash} 存储在 $Slave_3$ 上。

3.2.2　基于 MapReduce 的连接算法

对关系 R 和 S 使用 3.1 节的数据分类算法进行分类后，关系 R 和 S 被分成 3 个不相交的子关系：$R=R_{par}\cup R_{rep}\cup R_{hash}$ 和 $S=S_{par}\cup S_{rep}\cup S_{hash}$。$R$ 与 S 的连接操作可以转换为 R_{par} 与 S_{rep} 的连接、R_{rep} 与 S_{par} 的连接、R_{hash} 与 S_{hash} 的连接，3 个连接操作的合集即为 R 与 S 的连接结果。

定理 1　$R\bowtie S=(R_{par}\bowtie S_{rep})\cup(R_{rep}\bowtie S_{par})\cup(R_{hash}\bowtie S_{hash})$

证明：根据分类定义 1 可知，$R_{par}\bowtie S_{par}=\varnothing$，$R_{par}\bowtie S_{hash}=\varnothing$，$R_{rep}\bowtie S_{rep}=\varnothing$，$R_{rep}\bowtie S_{hash}=\varnothing$，$R_{hash}\bowtie S_{par}=\varnothing$，$R_{hash}\bowtie S_{rep}=\varnothing$，所以 $R\bowtie S=(R_{par}\cup R_{rep}\cup R_{hash})\bowtie(S_{par}\cup S_{rep}\cup S_{hash})=(R_{par}\bowtie S_{par})\cup(R_{par}\bowtie S_{rep})\cup(R_{par}\bowtie S_{hash})\cup(R_{rep}\bowtie S_{par})\cup(R_{rep}\bowtie S_{rep})\cup(R_{rep}\bowtie S_{hash})\cup(R_{hash}\bowtie S_{par})\cup(R_{hash}\bowtie S_{rep})\cup(R_{hash}\bowtie S_{hash})=(R_{par}\bowtie S_{rep})\cup(R_{rep}\bowtie S_{par})\cup(R_{hash}\bowtie S_{hash})$。

具体的连接过程如图 2 所示。

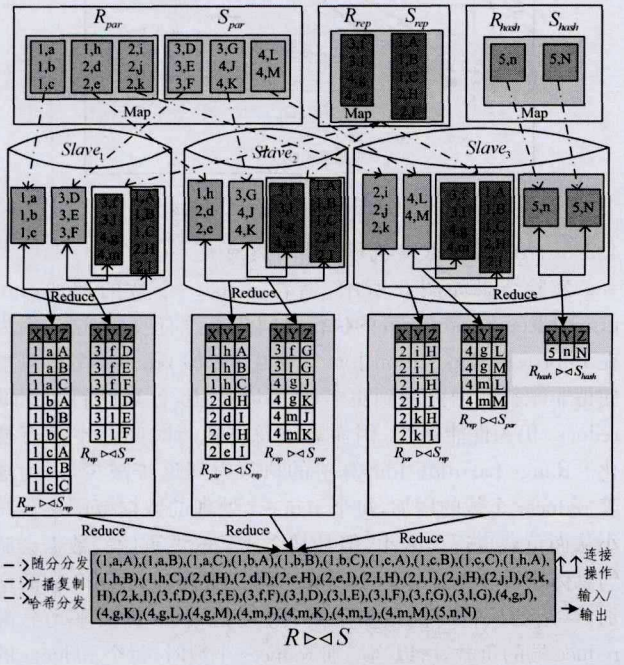

图 2　关系 R 和 S 的连接算法过程

算法 2　连接算法

1. Map 阶段：/ * 根据倾斜数据分类分发 Slave 存储表和数据 * /

1）根据倾斜数据分类分发 Slave 存储表，将所有记录分发到 Slave 上；

2. Reduce 阶段：/ * 读取 Map 的输出记录并产生连接结果 * /

1）Slave 上读取数据记录，同时执行 $R_{par}\bowtie S_{rep}$，$R_{rep}\bowtie S_{par}$，$R_{hash}\bowtie S_{hash}$ 3 种连接操作；

2）产生连接结果，写入 HDFS。

3.3　FC-Join 算法分析

通常采用时间复杂度和空间复杂度两种指标衡量算法的效率，即算法执行需要的运算和存储。连接查询是数据密集型计算的一种，该类型计算有以下特点：1）当数据量超过一个限度时，存储系统难以满足海量数据处理的读写需要，数据传输带宽成为计算的一个瓶颈[14]；2）在数据连接过程中，对数据集的操作是确定的，不论采用什么实现算法，其计算代价都近似相等[5]。综上，MapReduce 连接算法是一种数据密集型计算，二元连接的复杂度并不高，算法代价以磁盘 I/O 和网络传输为主。因此，本节对 FC-Join 算法各个阶段的成本以磁盘 I/O 和网络传输为主进行详细分析，以便能够对算法的代价进行量化估计。

（1）Map 阶段为输入关系的记录产生带标签的频次直方图。

每个 mapper 任务从 HDFS 中读取输入关系 R 和 S 的数据块并为 R_i^{map} 与 S_i^{map} 的每条记录输出键值对$(\langle K,tag\rangle,1)$，其中 K 表示连接属性 join_key，tag 表示输入关系标签。本步骤的成本是：

$$Cost(1.1.1)=O(\sum_{i=1}^{mappers}c_{r/w}*(|R_i^{map}|+|S_i^{map}|)+\sum_{i=1}^{mappers}(\|R_i^{map}\|+\|S_i^{map}\|))$$

对 map 输出的键值对$(\langle K,tag\rangle,1)$进行合并和分区。直方图 $Hist(R_i^{map})$和直方图 $Hist(S_i^{map})$需要排序和网络传输。本步骤的成本是：

$$Cost(1.1.2)=O(\max_{i=1}^{mappers}(\|Hist(R_i^{map})\|*\log\|Hist(R_i^{map})\|+\|Hist(S_i^{map})\|*\log\|Hist(S_i^{map})\|+c_{comm}*(|Hist(R_i^{map})|+|Hist(S_i^{map})|))$$

Map 阶段的成本是：

$$Cost_{step_{1.1}}=Time(1.1.1)+Time(1.1.2)$$

（2）Reduce 阶段计算连接结果频次直方图索引

reducer 任务计算同时出现在关系 R 和 S 中的 join_key 的总频次，对于每个连接属性 K，输出$(K,F(K))$的条目。对于给定的连接属性 $K\in HistIndex_i(R\bowtie S)$，使用 $HistIndex$ 信息，每个 reducer 任务就会知道输入关系的相关记录在下一个 map 阶段如何分发。

Reduce 阶段的总成本是：$Cost_{step_{1.2}}=O(\max_{i=1}^{reducers}(\|Hist(R_i^{red})\|+\|Hist_i^{red}(S_i^{red})\|))$。

1）Map 阶段创建哈希表并分发数据记录。

mapper 读取连接结果的直方图索引 $HistIndex$ 来创建本地哈希表。

$$Cost(2.1.1)=O(\max_{i=1}^{mappers}t_h*\|HistIndex(R\bowtie S)\|)。$$

创建本地哈希表后，从 HDFS 上读取记录分发到目的 reducer。本步骤的成本是：

$$Cost(2.1.2)=O(\max_{i=1}^{mappers}(c_{r/w}*(|R_i^{map}|+|S_i^{map}|)+t_s^i*(\|R_i^{map}\|+\|S_i^{map}\|)+\|\bar{R}_i^{map}\|*\log\|\bar{R}_i^{map}\|+\|\bar{S}_i^{map}\|*\log\|\bar{S}_i^{map}\|+c_{comm}*(|\bar{R}_i^{map}|+|\bar{S}_i^{map}|)))$$

其中，$c_{r/w}*(|R_i^{map}|+|S_i^{map}|)$是 mapper 任务从 HDFS 上读取记录的时间，$t_s^i*(\|R_i^{map}\|+\|S_i^{map}\|)$是在哈希表中对每条记录进行扫描的时间，$\|\bar{R}_i^{map}\|*\log\|\bar{R}_i^{map}\|+\|\bar{S}_i^{map}\|*\log\|\bar{S}_i^{map}\|$是对 mapper 任务上的相关记录进行排序的时间。$c_{comm}*(|\bar{R}_i^{map}|+|\bar{S}_i^{map}|)$是 mapper 任务向 reducer 任务传输相关记录的时间。Map 阶段的总成本是：

$$Cost_{step_{2.1}}=Cost(2.1.1)+Cost(2.1.2)$$

2)Reducce 阶段计算连接结果。

步骤 1)完成后，reducer 任务获得$\bar{R}_i^{red}$和$\bar{S}_i^{red}$并对其执行连接操作。Reducce 阶段的成本是：

$$Cost(2.2)=O(\max_{i=1}^{reducers}(\|\bar{R}_i^{red}\|+\|\bar{S}_i^{red}\|+c_{r/w}*|\bar{S}_i^{red}\bowtie\bar{S}_i^{red}|))$$

FC-Join 的总成本由以上 4 步构成：

$$Cost_{FC\text{-}Join}=Cost_{step_{1.1}}+Cost_{step_{1.2}}+Cost_{step_{2.1}}+Cost_{step_{2.2}}$$

通过对 FC-Join 算法的成本分析，能够了解本算法各步骤的时间成本，对算法的总体时间成本进行量化估计，从而大致估计算法的运行时间。接下来将对本算法的高效性进行验证。

4 实验验证与结果分析

实验对提出的 FC-Join 算法进行验证和分析。采用了对比测试方法，将提出的 FC-Join 算法与 Improved Repartition Join[12]和 Range partition Join[9]算法的执行效率进行对比。Improved Repartition Join 和 Range partition Join 算法与本文类似，都是通过改变默认的分区方法来优化 reduce 端的负载均衡，以提高连接算法的效率。通过对实验结果的分析验证了本算法的高效性。

4.1 实验数据

实验由 4 台 HP 刀片服务器搭建了一个内部网络，组成 4 个节点的 Hadoop 集群，其中 1 个节点作为 Master，其余 3 个节点作为 Slave，网络内的各个节点通过 100M 网卡相互访问。Master 节点服务器 CPU 为 Inter(R) Xeon(R) E5620 2.4GHz 4 * 4 核，Memory 为 16GB，Disk 为 300G * 8。Salve 节点服务器 CPU 为 Inter(R) Xeon(TM) 3.00GHz 4 核，Memory 为 8GB，Disk 为 146.8G * 2。每台服务器上安装 OS：64bit CentOS6.2，Hadoop 版本 1.0.3 和 Eclipse 版本 4.3.1。Hadoop 默认参数配置 block 为 64M，备份数为 3。

为了验证 FC-Join 算法的有效性，以等值连接为例，执行关系表 R 与 S 的连接操作，数据记录数分别为 $\|R\|$ 和 $\|S\|$，r 代表reduce 个数，α 代表关系 R 和 S 中连接属性值的倾斜度 $\alpha=(\|R_{par}\|+\|S_{par}\|)/(\|R\|+\|S\|)$。测试数据集：$\|R\|=0.5$ 千万，$\|S\|=1$ 千万，数据量为 15G；$\|R\|=1$ 千万，$\|S\|=2$ 千万，数据量为 30；$\|R\|=1.5$ 千万，$\|S\|=3$ 千万，数据量为 45G；$\|R\|=2$ 千万，$\|S\|=4$ 千万，数据量为 60G；$\|R\|=2.5$ 千万，$\|S\|=5$ 千万，数据量为 75G。

4.2 实验分析

通过 3 组实验分析在不同的倾斜度(α)、数据规模、reduce 个数时的算法效率，在每组实验中只改变一个参数，其他参数不变，以保证实验结果的有效性。

4.2.1 倾斜度对执行时间的影响

在不同倾斜度下的实验结果如图 3 所示。从图 3 可以看出，随着倾斜度的增大，FC-Join 算法的执行时间基本没有变化，而 Improved Repartition Join 算法和 Range partition Join 算法的执行时间有所增加。因为随着倾斜度的增大，必然导致 Improved Repartition Join 和 Range partition Join 算法中某个或某些 reducer 的负载不均衡，从而使执行时间增加。FC-Join 算法能够负载均衡所有 reduce 端的数据处理量，因为随着倾斜度增大，根据分发规则，shuffle 过程广播到 reduce 端的 R_{rep} 与 S_{rep} 类数据会减少，可以减少网络传输代价和 reduce 端处理的数据量，从而使算法的执行效率有所提高。

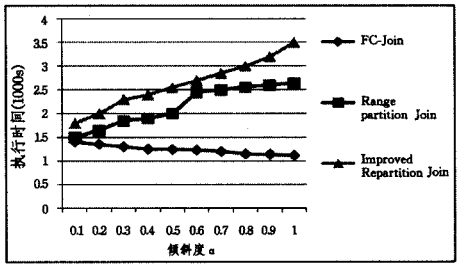

图 3 倾斜度对执行时间的影响

4.2.2 数据规模对执行时间的影响

不同数据规模下的实验结果如图 4 所示，随着数据集规模的增大，3 个算法的执行时间都有所增加。但是 FC-Join 算法的效率优于 Improved Reparttion Join 和 Range partition Join 算法。随着数据集规模的增大，Improved Reparttion Join 和 Range partition Join 算法的处理时间急剧增加，因为数据集规模的增大导致倾斜的数据量增加，reducer 负载不均衡对执行效率的影响加剧。FC-Join 算法能够保证 reduce 端的负载均衡，因此时间不会急速增加。

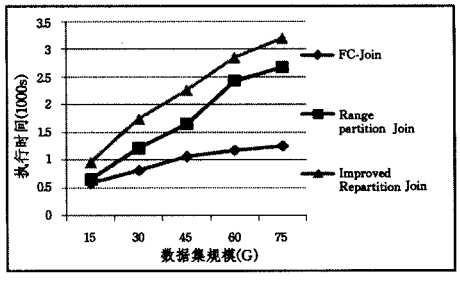

图 4 数据集规模对执行时间的影响

4.2.3 reduce 个数对执行时间的影响

实验结果如图 5 所示，随着 reducer 个数的增加，Improved Repartition Join 的执行时间几乎没有改变，这是因为在 Improved Repartition Join 算法中，增加 reducer 个数只能使得非倾斜数据分配到更多的 reducer 上，处理倾斜数据的 reducer 仍为性能瓶颈，因而算法总体执行时间基本没有变化。Range partition Join 算法的执行时间逐步减少，因为随着 reducer 个数的增加，每个 reducer 处理的数据范围逐步减小从而负载也逐步减小，但是某个或某些严重倾斜数据会制约总体执行效率。对于 FC-Join 算法，随着 reducer 个数的增加，算法的执行时间逐步下降，因为 FC-Join 算法平均分配 reduce 端的负载，所以当增加 reducer 个数时，每个 reducer 的负载会逐步减少，同时也增加了 reduce 端连接操作的并行

度，从而降低了总体执行时间。

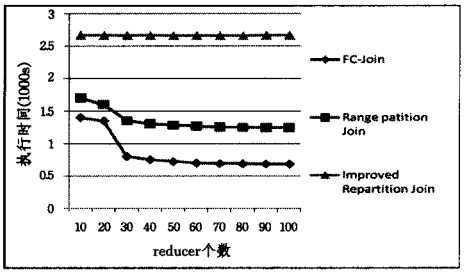

图 5　reducer 个数对执行时间的影响

结束语　本文研究了 MapReduce 环境下，引起数据倾斜的各种原因。针对数据倾斜现有连接算法 Hash Join、Broads Join、Range Join 等的不足，提出了基于 MapReduce 的连接算法 FC-Join。FC-Join 算法通过基于 MapReduce 的分布式直方图技术对连接数据的数据分布情况进行统计，根据连接属性值的统计结果对数据进行分类。对不同类型的数据采用适应的分发机制以保证 reducer 任务的负载均衡，从而保证连接操作的高效执行。通过对 FC-Join 算法的分析可以量化估计算法的执行时间。最后本文通过实验验证了 FC-Join 算法的高效性和实用性。本文主要研究了两表等值连接的情况，接下来将对多表连接（如星型连接）进行进一步的研究。希望可以从 MapReduce 连接操作的网络传输代价、磁盘 I/O 代价和响应时间等多方面去评估数据倾斜对表连接操作的影响。

参 考 文 献

[1] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[2] YongChul K, Magdalena B, Bill H, et al. A Study of Skew in MapReduce Applications[C]// Open Cirrus Summit. 2011

[3] Viswanath P, Yannis E I. Estimation of Query-Eesult Distribution and its Application in Parallel-Join Load Balancing[C]// Proceedings of the 22nd VLDB Conference (PVLDB). UMumbai(Bombay), India, 1996: 448-459

[4] Chen Yong-xu, Chen Meng-jie, Liu Xue-bin, et al. MapReduce Based Aggreate-Join Query Algorithms[J]. Journal of Computer Research and Development, 2013, 50(Suppl): 306-311(in Chinese)
陈勇旭，陈梦杰，刘雪冰，等. 基于 MapReduce 的连接聚集查询算法研究[J]. 计算机研究与发展，2013，50(Suppl)：306-311

[5] Song jie, Li Tian-tian, Zhu Zhi-liang, et al. Research on I/O Cost of MapReduce Join[J]. Journal of Software, 2015, 26(6): 1438-1456(in Chinese)
宋杰，李甜甜，朱志良，等. MapReduce 连接查询的 I/O 代价研究[J]. 软件学报，2015，26(6)：1438-1456

[6] Slagter K, Hsu C H, Chung Y C, et al. Smart Join: a network-aware multiway join for MapReduce[J]. Cluster Computing, 2014, 17(3): 629-641

[7] Hassan M A H, Bamha M. Towards Scalability and Data Skew Handling in GroupBy-Joins using MapReduce Model[J]. Procedia Computer Science, 2015, 51(1): 70-79

[8] Yu X, Pekka K, et al. Handling Data Skew in Parallel Joins in Shared-Nothing Systems[C]// SIGMOD 08. Vancouver, BC, Canada, 2008: 1043-1052

[9] Fariha A, Stratis D V, Salman N. SAND Join —A skew handling join algorithm for Google's MapReduce framework[C]// IEEE 14th International Multitopic Conference(INMIC). Karachi, Pakistan, 2011: 498-509

[10] David J D, Jeffrey F N, Donovan A S, et al. Practical Skew Handling in Parallel Joins[C]// Proceedings of the 18th VLDB Conference(VLDB). Vancouver, British Columbia, Canada, 1992: 27-40

[11] Zhou Jia-shuai, Wang Qi, Gao Jun. An Approach for Load Balancing in MapRedcue via Dynamic Partitioning[J]. Journal of Computer Research and Development, 2013, 50(Suppl): 369-377 (in Chinese)
周家帅，王琦，高军. 一种基于动态划分的 MapReduce 负载均衡方法[J]. 计算机研究与发展，2013，50(Suppl)：369-377

[12] Gufler B, Augslen N, Reiser A, et al. Load balancing in MapReduce based on scalable cardinality estimates[C]// Proc of ICDE. Piscataway, NJ, IEEE, 2012: 522-533

[13] Gufler B, Augsten N, Reiser A, et al. Handling data skew in MapReduce[C]// Proc of CLOSER. Portugal: SciTePress, 2011: 574-583

[14] http://www.enet.com.cn/article/2012/0401/A20120401990472.shtml

(上接第 17 页)

[37] Zhang Deng-hui, Yu Le. Support Vector Machine Based Classification for Hyperspectral Remote Sensing Images after Minimum Noise Fraction Rotation Transformation[C]// 2011 International Conference on Internet Computing and Information Services. HongKong, China: IEEE, September, 2011: 132-135

[38] Xu Han, Sun Yong-hua, Li Xiao-juan. Unmixing of Remote Sensing Images Based on Weighted Posterior Probability Support Vector Machines[J]. Journal of the Earth Information Science, 2013, 15(2): 249-254(in Chinese)
许菡，孙永华，李小娟. 遥感影像混合像元的分解—基于加权后验概率的支持向量机分类算法[J]. 地球信息科学学报，2013，15(2)：249-254

[39] Zhang Hua, Shi Wen-zhong, Liu K. Fuzzy-Topology-Integrated Support Vector Machine for Remotely Sensed Image Classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2012, 50(3): 850-862

[40] Han Ling, Wu Jing, Zhang Ruo-lan. The Classification Research of Support Vector Machine Based on Spot for Hyperspectral Remote Sensing Application[C]// 2010 International Conference on Computational and Information Sciences. Chengdu, China: IEEE, 2010: 1009-1012

[41] Li Cheng-fan, Yin Jing-yuan. Variational Bayesian independent component analysis-support vector machine for remote sensing classification[J]. Computers and Electrical Engineering, 2013, 39(3): 717-726

[42] Tan Kun, Du Pei-jun, Wang Xiao-mei. Multi-Class Support Vector Machine Classifier Based on Separability Measure for Hyperspectral Remote Sensing Image Classification[J]. Journal of Wuhan University(Information Science), 2011, 36(2): 171-175(in Chinese)
谭琨，杜培军，王小美. 利用分离性测度多类支持向量机进行高光谱遥感影像分类[J]. 武汉大学学报(信息科学版)，2011，36(2)：171-175