最小非平凡函数依赖发现算法原理说明

搜索策略

由于平凡函数依赖在任何情况下都成立，这里只需要搜索非平凡函数依赖。给定关系 r 的属性集合 X，在进行候选函数依赖搜索时，对所有形式如 $X\backslash\{A\} \rightarrow X$ 进行搜索，这保证了搜索的所有候选函数依赖为非平凡的函数依赖

候选函数依赖的搜索本文采用逐层搜索的方法，如图 1 所示，在具体搜索过程中，本文采用候选函数依赖的 LHS 部分包含属性个数由少到多、自底向上进行搜索，这种搜索的一个好处是便于进行候选函数依赖的剪枝。

假定关系 r 包含 n 个属性，则在逐层搜索时，首先从第 n 层开始，搜索 LHS 部分包含 1 个属性的候选函数依赖，即 $1attr$ 候选函数依赖，然后进入第（n-1）层，搜索 $2attr$ 候选函数依赖，以此类推，直到搜索完第 1 层，得到所有 $(n-1)attr$ 的候选函数依赖为止

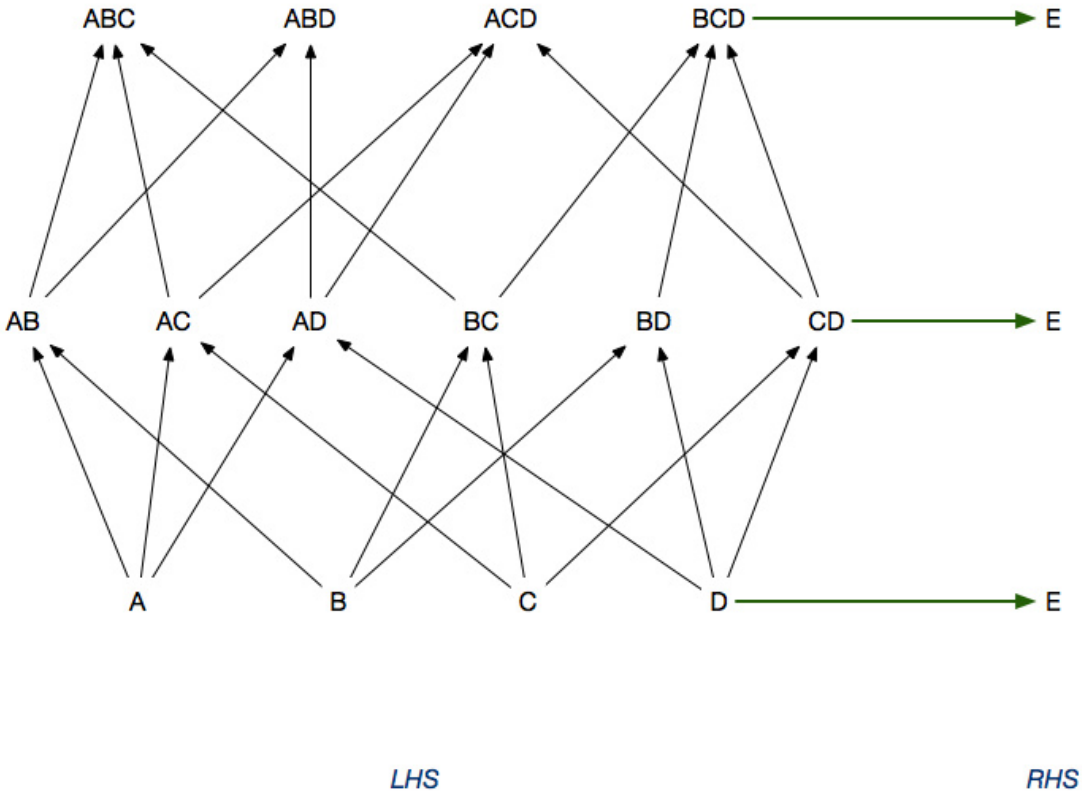

图 1 LHS 属性集为 $\{A, B, C, D\}$，RHS 属性为 $\{E\}$ 的候选函数依赖组合

剪枝策略

为提高函数依赖发现的效率，考虑对候选函数依赖集进行剪枝。

上文中的候选函数依赖的搜索策略，从 LHS 部分包含最少属性的候选函数依赖开始，逐层向上进行搜索。在搜索过程中，如果出现候选函数依赖成立的情况，则改函数依赖符合最小非平凡性质，可以对与之相关的 LHS 部分为该 LHS 属性超集的的候选函数依赖进行剪枝，因为其不符合最小性质

注：　如果候选函数依赖 $X \rightarrow A$ 为最小非平凡函数依赖，则必然有 $Y \rightarrow A$ 不符合最小非平凡函数依赖，其中 $X \subset Y$.

如图 2 所示，对最底层候选函数依赖进行判断，得到非平凡最小函数依赖 $B \to E$, $D \to E$，向上一层搜索候选依赖时，可对 $AB \to E$, $AD \to E$, $BC \to E$, $BD \to E$, $CD \to E$ 进行剪枝

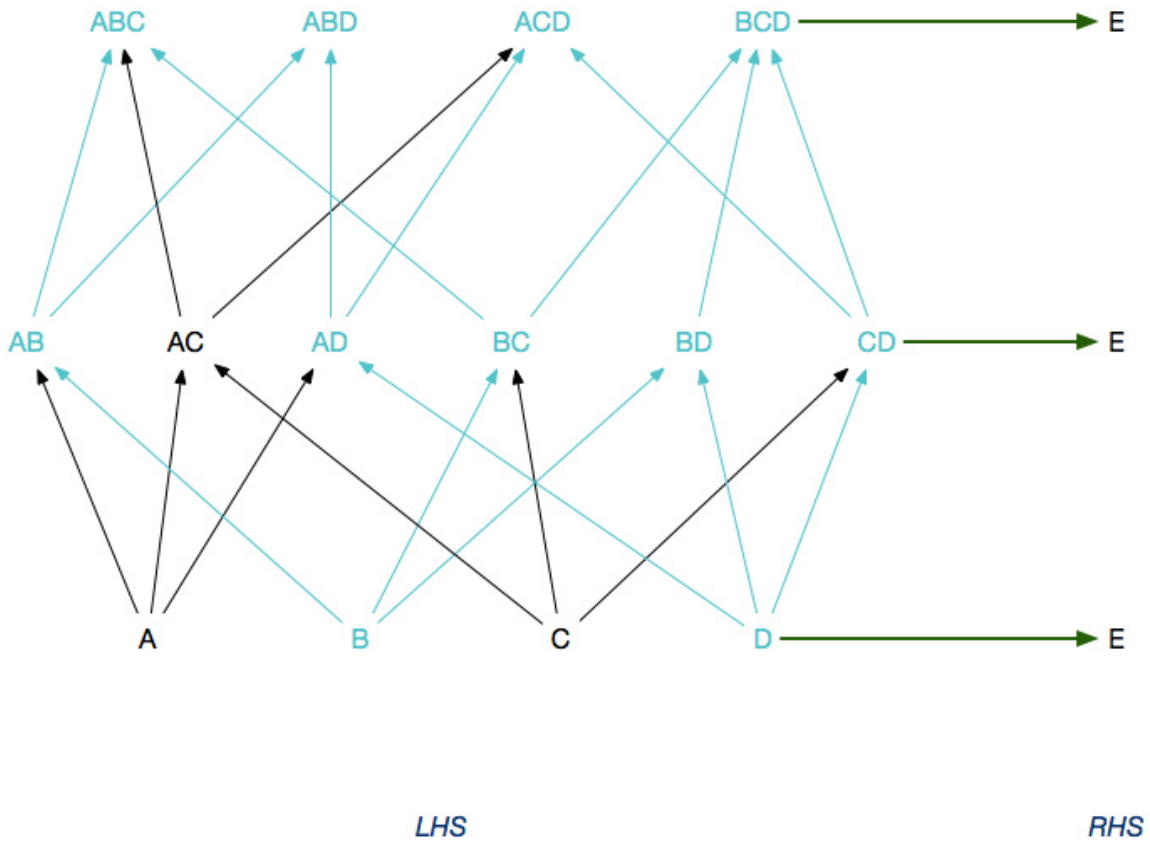

图 2 候选函数依赖搜索空间剪枝

函数依赖发现策略

函数依赖定义

假设$R = \{A_1, ..., A_m\}$为数据关系模式，其中包含了 m 个属性。R 上每一个属 性称为 A，每一个属性子集称为$X/subseteqR$。R 中的每个元组为 t，这里用 $t[A]$表示元组 t 中属性 A 的值，用 $t[X]$ 表示元组 t 在属性子集 X 上的值。

定义 假设存在两个属性子集 X 和 Y，函数依赖 X→Y 成立，当且仅当 R 中任 意两个元组 t 1 ，t 2 满足以下条件：如果 $t_1[X] = t_2[X]$，则 $t_1[Y] = t_2[Y]$。X 称为函数依赖 的左部，Y 称为函数依赖的右部。

函数依赖判断

假设一个候选函数依赖为 $\{A, B, C\} \to E$，逐行遍历整个数据集，拼接 A, B, C 属性字段对应的数据值组合成 LHS，E 对应的数据值为 RHS，以 LHS 为 key， RHS 为 value 组成键值，两两比较 key 相同的数据的 value，若存在 value 不一致的情况，则该候选不符合函数依赖性质，反之，则符合

分布式下并行计算策略

利用 spark RDD 实现函数依赖判断

RDD 是弹性分布式数据集，本质上是一个只读的分区记录集合，每个RDD可以分成多个分区，每个分区就是一个数据集片段，不同分区可以被保存到集群中不同的节点上，从而可以在集群中的不同节点上进行并行计算，和减少通信开销

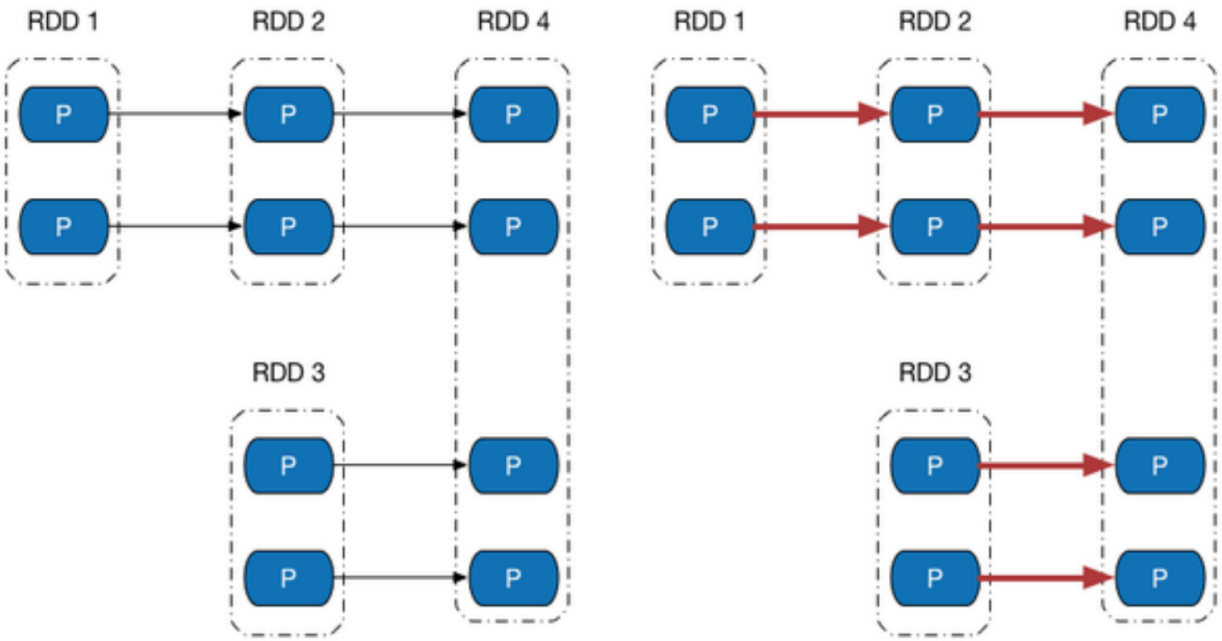

图3 RDD 分区示例

把数据集转换为 RDD 后即可对其实现分布式并行处理计算。上文中说明了通过 HashMap 来完成函数依赖判断的思想，而在 spark 平台上，可以用 reduceByKey 接口来实现，如图 4 所示，分布在不同节点、不同分区上的数据集片段是如何计算的

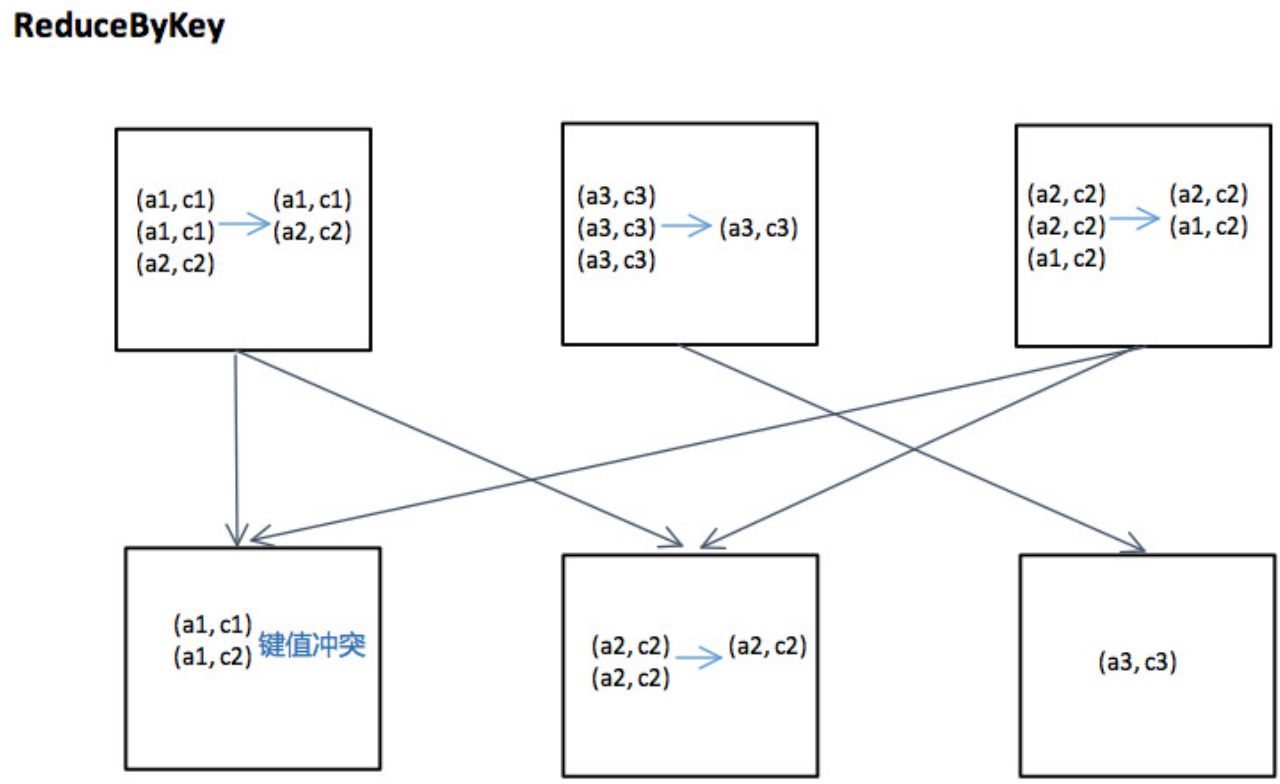

图 4 reduceByKey 流程

并行作业

使用多线程并发提交 spark 作业，充分利用集群性能

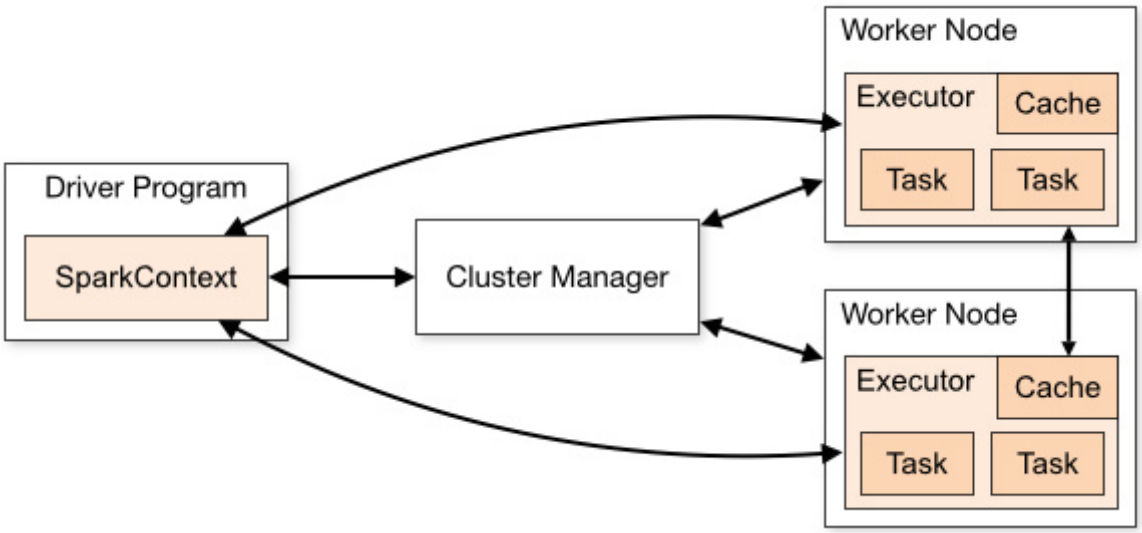

图 5 spark 集群工作示例