

分布式数据库数据副本最优安置问题的研究

朱泓丞,徐志广

(中国科学技术大学 计算机科学与技术系,合肥 230027)

(zhuhcheng@gmail.com)

摘要:对树形网络上的数据副本最优安置问题,在已有 K 子树中心优化模型的基础上提出了 K 节点中心的改进模型。改进模型相对于原有模型优化了分布式数据库更新操作的执行代价。给出了两个动态规划算法来求解树形网络 K 节点中心问题,一个是非常简单的复杂度较高的动态规划,另一个是使用分治的较复杂的高效动态规划,最后通过实验验证了模型的优化作用。

关键词:分布式数据库; K 中心问题;树形网络;动态规划算法

中图分类号: TP301.6 **文献标志码:** A

Research on optimal placement of data copies in distributed database

ZHU Hong-cheng, XU Zhi-guang

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei Anhui 230027, China)

Abstract: For optimally placing data copies in a tree network, an improved optimizing model, K -node core, was proposed, based on the existing model K -tree core. The improved model had better performance in updating database, compared to the original model. Two efficient dynamic programming algorithms for solving K -node core problem in a tree network were given. The optimizing model was tested by experiments.

Key words: distributed database; K -median problem; tree network; dynamic programming algorithms

0 引言

分布式数据库^[1]是数据库技术与网络技术相结合的产物。分布式数据库在多个网络服务器上保存数据副本,从而减少了来自网络其他主机的查询请求的执行时间,并且能够增强数据库的可靠性。但是保存多个数据副本增加了数据更新操作的开销,因为每次需要更新所有数据副本。分布式数据库的设计需要在优化查询和优化数据更新之间做出权衡,一方面要选择足够数量的服务器来存放数据副本,另一方面要避免过多的服务器带来的数据更新操作的额外开销,所以数据副本的最优安置问题被提出和研究。

在实际应用中,树形网络因其拓扑结构简单和费用较小而得到广泛使用,树形网络上的最优安置问题已有一些模型和算法。文献[2]研究了树形网络上数据副本的最优安置问题,提出了 K 子树中心(K -tree core)的概念,即在树形网络上选择若干个副本服务器,使得这些服务器形成一个有 K 个叶子节点的原网络的子树,优化的目标是所有网络主机离最近的副本服务器的距离的加权和最小,主机的权值是该主机发出查询操作的频率,而在这种模型下的最优安置对应的副本服务器集合被称为该树网络的 K 子树中心。文献[3]给出了一个在线性时间内求得给定树形网络的 K 子树中心的算法。文献[4]在 K 子树中心的基础上提出了 (L, K) 子树中心的概念,即在原 K 子树中心的条件下增加子树的直径不大于 L 的约束,目的是使得在一个相对较短的时间段内所有的数据副本都能够完成同一个更新操作。文献[4]也给出了一个计算 (L, K) 子树中心的高效算法。

上述的 K 子树中心和 (L, K) 子树中心的优化模型都有一个共同的缺点:虽然子树的叶子节点个数为已知的 K ,但是最优子树的节点数目可能远大于 K ,于是副本服务器可能过多,从而造成数据更新操作的额外开销很大。本文为解决这一问题,提出树形网络 K 节点中心(K -core)的优化模型和相应的计算 K 节点中心的算法,即在限制副本服务器的数目为 K 的条件下优化查询操作的平均执行时间,从而达到了查询和更新之间更好的平衡。

K 节点中心实际上是树形网络上带约束的 K -median问题,增加了要求选取的median节点数目正好为 K 的约束条件。 K -median问题由于其在聚类算法、物流运输和互联网等领域的大量应用而被广泛研究。文献[5-6]介绍了树形网络上的 K -median问题,并给出了相应的动态规划算法。文献[7]研究了 K -median问题在互联网中的应用。 K 节点中心问题也是被广泛研究和应用的设施选址问题的一个特殊形式,文献[8]详细介绍了设施选址问题的各种形式。

1 问题描述

K 节点中心可用图论语言描述如下。

给出一个树形网络 $T(V, E)$,其 V 和 E 分别是节点集合和边集合,对应网络中的主机和主机之间的链路。对任意节点 $v \in V$,节点权值 $w(v)$ 对应网络上该主机发送数据库查询请求的频率。对任意边 $e \in E$,边长度 $l(e)$ 对应该条网络链路的传输时间。对任意节点 v 和 V 的子集 S , v 到 S 的距离定义为:

$$D(v, S) = \min\{D(v, x) \mid x \in S\} \quad (1)$$

其中 $D(v, x)$ 为节点 v 和 x 在树 T 中的距离,即 v 和 x 之间的唯

收稿日期:2009-03-12;修回日期:2009-04-20。

作者简介:朱泓丞(1982-),男,四川巴中人,硕士研究生,主要研究方向:算法及算法应用;徐志广(1983-),男,河南开封人,硕士研究生,主要研究方向:并行机系统机构、多级互联网结构与分析。

一路径上所有边的长度之和。式(1)表示任意网络主机发出的查询请求都由距离最近的副本服务器来处理。

K 节点中心定义为满足如下条件的 V 的一个子集 C : C 是 T 的一个子树且正好包含 K 个节点,并且 T 中所有节点到 C 的距离的加权和尽量小,即

$$WD(T, C) = \sum_{v \in T} w(v)D(v, C) \quad (2)$$

尽量小。式(2)的实际意义是,在选择集合 C 对应的网络主机作为数据副本服务器的情况下,使得来自网络主机的查询请求的平均执行时间尽量短。

K 节点中心约束了副本服务器的个数为 K ,从而保证了数据更新操作的额外开销不会太大,同时优化查询操作的平均执行时间尽量小,从而保证了在满足前一条条件的同时达到最优的查询执行时间。所以 K 节点中心较好的实现了优化查询和优化更新两者之间的平衡。

2 K 节点中心算法描述

本节给出两个计算 K 节点中心的算法,一个简单的动态规划算法和一个使用分治的相对较复杂的动态规划算法。前者的时间复杂度比后者高,但是编程实现比后者简单,不同的应用环境可以选择不同的算法。先给出将要使用的一些符号表示。

表1 一些符号及其代表的含义

符号	符号代表的含义
K	K 节点中心的节点个数
N	树 T 的节点个数
T_r	由树 T 转化而来的以 r 为根的有根树
$T_r(v)$	有根树 T_r 中以节点 v 为根的子树
$w(S)$	节点集合 S 权值和,即 $w(S) = \sum_{v \in S} w(v)$
$D(v, S)$	节点 v 到节点集合 S 的距离,见式(1)
$WD(T, C)$	T 中节点到节点集合 S 的加权距离和,见式(2)
$T_r(v, k)$	由节点 v 及其前 k 个儿子节点对应的子树的并构成的子树
$Euler(T_r)$	T_r 的欧拉序列
$ET(T_r, k)$	$Euler(T_r)$ 的前 k 项对应节点集合导出的 T_r 的子树
$T/T_c(v_i)$	从 T 中删掉 $T_c(v_i)$ 后剩余的子树

2.1 简单动态规划算法

首先,任意选择树 T 的一个节点 r 作为根节点,把树 T 转化为以 r 为根的有根树 T_r 。

对于节点 v 和正整数 n 定义状态函数 $f(v, n)$ ($1 \leq n \leq K$, $v \in V$),其意义为子树 $T_r(v)$ 上包含节点 v 的最优 n 节点中心对应的加权距离和,严格定义如下:

$$f(v, n) = \min\{WD(T_r(r), C) \mid C \subseteq T_r, |C| = n, v \in C\} \quad (3)$$

对于节点 v 和正整数 n ,假设 v 在 T_r 中的儿子节点为 v_1, v_2, \dots, v_m ,定义另一个状态函数 $g(v, n, k)$ ($1 \leq n \leq |T_r(v)|$, $0 \leq k \leq m$),其意义是由 v 和 $T_r(v_1), T_r(v_2), \dots, T_r(v_k)$ 构成的子树 $T_r(v, k)$ 上包含节点 v 的最优 n 节点中心对应的加权距离和,严格定义如下:

$$g(v, n, k) = \min\{WD(T_r(v, k), C) \mid C \subseteq T_r(v, k), |C| = n, v \in C\} \quad (4)$$

基于式(3)、(4)的状态表示,可以得到状态间递推关系

(5)和(6)。

$$f(v, n) = g(v, n, m) \quad (5)$$

其中 m 是节点 v 在 T_r 中的儿子节点的个数。

$$g(v, n, k+1) = \min\{g(v, n-i, k) + f(v_{k+1}, i) \mid 0 < i < n\} \quad (6)$$

其中 v_{k+1} 是 v 在 T_r 中的第 $k+1$ 个儿子节点。式(6)的意义是通过枚举在子树 $T_r(v_{k+1})$ 中选取的节点个数来找到最优值。

另外对于节点 v 有边界条件:

$$f(v, 1) = WD(T_r(v), v) \quad (7)$$

$$g(v, 0, n) = 0; 1 \leq n \leq K \quad (8)$$

由式(5)~(8)可以计算所有的状态函数, K 节点中心对应的最小加权距离和即为如下表达式:

$$\min\{f(v, K) + WD(T, v) - WD(T_r(v), v) \mid v \in T_r\} \quad (9)$$

分析动态规划可知,总共有 $O(NK)$ 个状态函数需要计算,对每个状态 $g(v, n, k)$ 用式(6)递推计算的时间复杂度是 $O(K)$,所以简单动态规划算法的时间复杂度是 $O(NK^2)$ 。

2.2 高效动态规划算法

本节的算法基于树分解和有根树的欧拉序列两个概念。树分解基于以下引理。

引理1 树中心引理。对任意树 $T(V, E)$,存在节点 $c \in V$,使得去掉该节点后形成的任意子树包含的节点数目都不超过 T 中节点个数的一半,满足以上条件的节点被称为树 T 的中心。

证明 任选 T 的一个节点 r 作为根节点,把 T 转为以 r 为根的有根树 T_r 。如果以 r 的儿子节点为根的任何子树含有的节点个数都不大于 $|V|/2$,那么 r 就是 T 的一个中心。否则,设 r 的儿子节点 v 对应的子树含有超过 $|V|/2$ 个节点(这样的儿子节点最多只能有一个),那么将 T 转为以 v 为根的有根树 T_v ,并重复上述步骤。容易看出上面的过程一定会终止,即某个中心被找到,因为含有最多节点的以树根节点的儿子节点为根的子树的节点个数在严格递减。

从引理1证明过程中容易看出,可以在线性时间内找到 T 的一个中心 c 。

树分解过程就是找到 T 的一个中心 c ,然后从 T 中删掉 c 从而得到若干子树 T_1, T_2, T_3, \dots ,再递归的对这些子树做同样的分解,知道所有子树都成为孤立节点。由树分解的定义容易得到如下引理2。

引理2 树分解引理。树分解的递归深度最多为 $O(\log |V|)$ 层。

证明 因为分解深度每增加一层,新生成的子树的节点个数至多为原树的一半。

有根树的欧拉序列是图论中一个重要概念,其定义如下。

定义1 欧拉序列。对于有根树 T_r ,这里节点 r 是树根,假设 r 的儿子节点为 v_1, v_2, \dots, v_m ,那么 T_r 的欧拉序列 $Euler(T_r)$ 递归定义为节点序列 $(r, Euler(T_r(v_1)), r, Euler(T_r(v_2)), r, \dots, Euler(T_r(v_m)), r)$ 。对于单个节点的树 r , $Euler(r)$ 定义为 (r) 。

从欧拉序列的定义可以看出, $Euler(T_r)$ 的任意前缀包含的节点导出 T_r 的一个子树。

本节的动态规划算法的框架是,找到树 T 的一个中心 c ,利用欧拉序列 $Euler(T_c)$ 上的动态规划算法计算出 T 上含有 c 的最优 K 节点中心。然后从 T 中删掉 c ,形成若干子树 $T_1, T_2,$

T_3, \dots , 适当修改这些子树使得这些子树的 K 节点中心正好对应 T 上不包含 c 的最优 K 节点中心, 从而得到 T 的 K 节点中心。

首先描述欧拉序列上的动态规划算法。

算法1 欧拉序列动态规划算法: $EulerDP(T_r, K)$

输入: 以节点 r 为根的有根树 T_r , 参数 K 。

输出: T_r 上包含 r 的一个最优 K 节点中心。

算法伪码如下。

1) 在线性时间内得到 $Euler(T_r) = (v_1, v_2, v_3, \dots)$ 。

2) 定义状态函数 $f(k, n)$ 表示从 $Euler(T_r)$ 的前 k 项导出的子树 $ET(T_r, k)$ 上选择包含节点 r 和 v_k 的最优 n 节点中心对应的加权距离之和, 严格定义如下:

$$f(k, n) = \min \{ WD(T_r, C) \mid C \subseteq ET(T_r, k), r \in C, v_k \in C, |C| = n \} \quad (10)$$

3) 按如下边界条件和递推关系递推计算所有的状态函数:

$$f(1, n) = WD(T_r, r); 1 \leq n \leq K \quad (11)$$

$$f(v, 1) = \infty; v \neq r \quad (11)$$

如果 v_{k+1} 是 v_k 的父亲节点, 则:

$$f(k+1, n) = \min \{ f(k, n), f(k+1, n) \} \quad (13)$$

如果 v_k 是 v_{k+1} 的父亲节点, 则

$$f(k+1, n+1) = f(k, n) - WD(T_r(v_k), v_k) + WD(T_r(v_{k+1}), v_{k+1}) \quad (14)$$

由以上算法描述容易看出, 算法1的动态规划需要计算 $O(NK)$ 个状态函数, 由状态递推关系可知每个状态函数可以在常数时间求得, 所以算法1的时间复杂度是 $O(NK)$ 。

基于树分解的 K 节点中心算法详细描述如下。

算法2 K 节点中心算法: $Kcenter(T, K)$

输入: 树 T , 参数 K 。

输出: T 的一个 K 节点中心

算法伪码如下。

1) 在线性时间内找到 T 的一个中心 c , 并将 T 转化为以 c 为根的有根树, 设 c 的儿子节点为 v_1, v_2, \dots, v_m 。

2) 利用算法1计算得到 T 上的包含 c 的最优 K 节点中心 $C(T, c)$ 。

3) 修改 v_i 的权值 $w(v_i) := w(v_i) + w(T) - w(T_i)$ 。

4) 然后对每个儿子节点 v_i 递归调用 $Kcenter(T_c(v_i))$, 得到 $T_c(v_i)$ 上的最优 K 节点中心 $C(T_c(v_i))$ 。

5) 返回 $C(T, c)$ 和 $C(T_c(v_1)), C(T_c(v_2)), \dots, C(T_c(v_m))$ 中最优的那一个作为 T 的 K 节点中心。

算法2的步骤3)中修改 v_i 的权值相当于把 $T/T_c(v_i)$ 中的节点的权值全部压缩到 v_i 上, 使得计算出的 $C(T_i)$ 包含了 $T/T_c(v_i)$ 中节点的加权距离和。算法2实际上考虑了 T 的两类 K 节点中心, 一类包含 T 的中心 c , 另一类不包含 c 。前一类通过欧拉序列上的动态规划算法求得, 后一类通过递归求解。

由引理2可知算法2的递归深度最多为 $O(\log N)$, 时间复杂度最高的步骤2可以在 $O(NK)$ 时间内完成, 所以算法2的时间复杂度为 $O(NK \log N)$, 相对于简单动态规划算法有了数量级的改进。

3 实验结果

本章利用实验对比了 K 子树中心, (K, L) 子树中心和 K

节点中心三个分布式数据库数据副本最优安置问题优化模型的效果。实验随机生成 100 到 1000 个节点范围内的树网络, K 为网络节点个数 10% 到 20% 之间的随机数, 这些参数范围能够反映一般规模的分布式数据库。实验对比结果如表 2。

表2 对比实验结果

(网络节点 数, K)	K 子树中心		(K, L) 子树中心		K 节点中心	
	查询代 价 / %	更新代 价 / %	查询代 价 / %	更新代 价 / %	查询代 价 / %	更新代 价 / %
(100, 10)	100	100	118	58	101	44
(200, 38)	100	100	128	83	114	82
(300, 57)	100	100	114	83	122	82
(400, 48)	100	100	122	81	102	49
(500, 51)	100	100	108	66	103	84
(600, 118)	100	100	120	75	103	66
(700, 84)	100	100	126	55	103	53
(800, 111)	100	100	126	84	114	77
(900, 118)	100	100	111	58	103	65
(1000, 165)	100	100	120	65	100	56

表2中的查询与更新代价是相对于 K 子树中心算法的对代代价的百分比。从表中可以看出, (K, L) 子树中心和 K 节点中心算法相对于 K 子树中心算法都用较小的额外查询代价换来了较大的更新代价的减少量, 充分体现了两个算法对更新操作的优化。还可以从表中看出, K 节点中心算法在整体上比 (K, L) 子树中心性能更好。

4 结语

本文对分布式数据库的数据副本最优安置问题提出了 K 节点中心优化模型和求解 K 节点中心的动态规划算法, 并通过对实验验证了优化模型的优化效果。

参考文献:

- [1] 肖凌, 刘继红. 分布式数据库系统的研究与应用[J]. 计算机工程, 2001, 27(1): 86-89.
- [2] STEPHENS A B, YESHA Y, HUMENIK K. Optimal allocation for partially replicated database systems on tree-based networks[C]// Proceedings of the 11th International Phoenix Conference on Computing and Communication. Philadelphia, PA, USA: [s. n.], 1992, 4(3): 125-129.
- [3] SHIOURA A, UNO T. A linear time algorithm for finding a k-tree core [J]. Journal of Algorithms, 1997, 23(1): 281-290.
- [4] WANG B F, PENG S, YU H Y, et al. Efficient algorithms for a constrained k-tree core problem in a tree network [J]. Journal of Algorithms 2006, 59(2): 107-124.
- [5] TAMIR A. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs [J]. Operations Research Letters, 1996, 19(1): 59-64.
- [6] VIGNERON A, GOLIN M, ITALIANO G. An algorithm for finding a k-median in a directed tree [J]. Information Processing Letters, 2000, 74(1): 81-88.
- [7] LI B, GOLIN M. On the optimal placement of Web proxies in the Internet [C]// Proceedings of the 18th Conference on Computer Communications. New York, USA: [s. n.], 1999, 3(2): 1282-1290.
- [8] 王菲, 徐渝, 李毅学. 离散设施选址问题研究综述[J]. 运筹与管理, 2006, 15(5): 64-69.