第36卷 第20期 计 算 机 工 程 2010年10月
Vol.36 No.20 Computer Engineering October 2010
·软件技术与数据库· 文章编号：1000—3428(2010)20—0058—03 文献标识码：A 中图分类号：TP393

质量驱动的 Web 服务 Top-K 查询

金 侠，杨卫东
(复旦大学计算机科学技术学院，上海 200433)

摘 要：在 Web 服务发现领域中，引入 Top-K 查询且同时考虑服务质量的研究相对较少。针对这种情况，提出一种重点考虑质量的 Web 服务 Top-K 查询方案。在该方案中，规范化服务的质量数据，给出质量评价函数，根据质量定义服务之间的从属关系，并且改进传统 Top-K 查询的门槛算法，提出收敛速度更快的 StopLine 算法。实验表明，该算法能更快地得到前 *K* 个服务，效果良好。

关键词：Web 服务发现；服务质量；Top-K 查询

Quality-driven Top-K Query of Web Services

JIN Xia, YANG Wei-dong
(School of Computer Science, Fudan University, Shanghai 200433, China)

【Abstract】 In the current research area on Web services discovery, solutions with Top-K query and taking the quality of services into account meanwhile are not common. In the view of the situation, this paper presents a solution for Top-K Web services, focusing on the quality of services. The quality data of services are normalized, the quality evaluation function is given, and the dominance relationship between services is defined. Especially, the typical threshold algorithm is improved to propose StopLine algorithm to get Top-K Web services, which converges faster. The algorithm is proved to be more efficient in the experiment.

【Key words】 Web services discovery; Quality of Service(QoS); Top-K query

1 概述

当前 Web 服务发现领域主要有 2 类解决方法：(1)基于传统信息检索技术的方案，例如关键字查找。(2)利用形式化逻辑，多用于结构化查询。然而，用户对“合适”服务的定义，并非仅仅在于功能的合适，服务的质量同样也是用户关心的重要因素[1]，例如一个服务满足功能的需求，但可用性却非常低，那么这样的服务必然不会被用户接受。而且，用户通常也没有时间去查看所有服务的质量。

针对上述问题，本文提出了质量驱动的 Top-K 服务查询方案。Top-K 查询[2]是一个传统的研究课题，而 Web 服务发现通常只是根据文本相似度等信息检索技术给服务打分排序，并没有显式地做 Top-K 查询。考虑质量的文章[3]也很多，但结合 Top-K 查询的较少。本文将服务发现和 Top-K 查询结合起来，相比前述 2 类解决方法，重点考虑了服务的质量。在改进传统的 Top-K 算法基础上提出了收敛更快的 StopLine 算法，应用在服务发现上。算法的关键在于找到更早的“停止线”。算法对比实验结果表明，大部分情况下 StopLine 可以更快地返回质量较优的那些服务。

2 Web 服务质量

衡量 Web 服务质量的指标尚未有统一的定义。文献[1]中将评价质量的参数分为 4 大类：运行时相关，事务相关，配置管理，安全。

2.1 服务质量数据

采用以下常用指标：响应时间，指用户发送服务请求至得到返回结果的时间段；时延，指服务提供方开始处理服务请求至完成服务返回结果的时间段；吞吐量，通常是单位时间内完成的服务请求的数量；可靠性，指给定时间内，服务提供方成功完成的服务请求的数量占总请求数量的比；可用性，定义为给定时间内服务可用的时间占的比。为简便，下文分别用 *Res*、*Lat*、*Thr*、*Rel*、*Ava* 来表示它们。

2.2 质量数据的规范化

如前文所见，衡量服务质量的指标单位各异，并且指标高低所对应的质量优劣含义也不同。响应时间越低，服务的质量越高；但吞吐量和可靠性等却相反，服务质量与它们成正比。为了能够从各个独立的指标获得对服务质量的综合评价，首先需要对数据做规范化处理。

针对响应时间这种服务质量与其成反比的指标，找到表中最小的数值 $Res_{\min}$，将其余的数据如式(1)进行规范化。这样响应时间最低的 $Res_{\min}$ 规范化为 1，而其余较高的响应时间 Res_{origi} 都将规范化为小于等于 1 的数值 Res_{norm}。

$$Res_{norm} = Res_{\min} / Res_{origi} \quad (1)$$

$$Thr_{norm} = Thr_{origi} / Thr_{\min} \quad (2)$$

对服务质量与之成正比的数据，例如吞吐量，采用式(2)进行规范。其中，$Thr_{\min}$ 是表中最小的吞吐量数值。那些吞吐量较高的 Web 服务，则会被规范化为大于等于 1 的 Thr_{norm}。剩下的指标 *Lat*、*Rel*、*Ava* 类似，分别按照它们的性质规范化即可。

更复杂的规范化可以考虑给每个指标增加相应的权重(满足用户更重视某些质量指标的要求)或者改变分子分母的

基金项目：上海市基础研究基金资助重点项目(08JC1402500)
作者简介：金 侠(1984－)，男，硕士研究生，主研方向：XML 数据管理，Web 服务；杨卫东，副教授
收稿日期：2010-05-30 **E-mail：**072021129@fudan.edu.cn

计算公式来增加差异，这些可以作为下一步的工作。

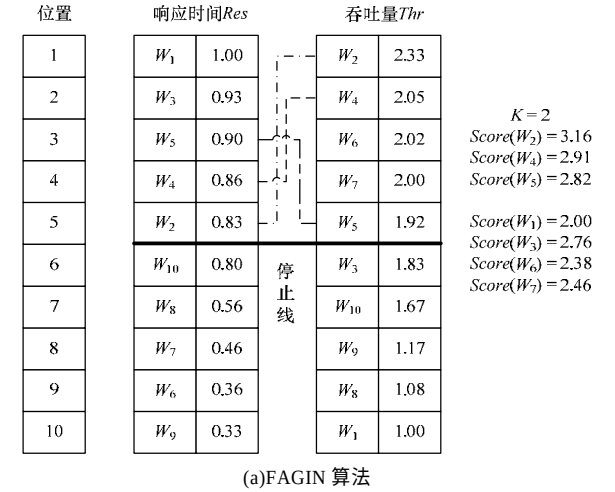
规范化处理之后，就可以给出综合评价。评价的函数可以采用各个指标分量的加和。如一个服务 W_i 的质量好坏由式(3)决定。

$$Score(W_i) = Score_{Res}(W_i) + Score_{Thr}(W_i) \quad (3)$$

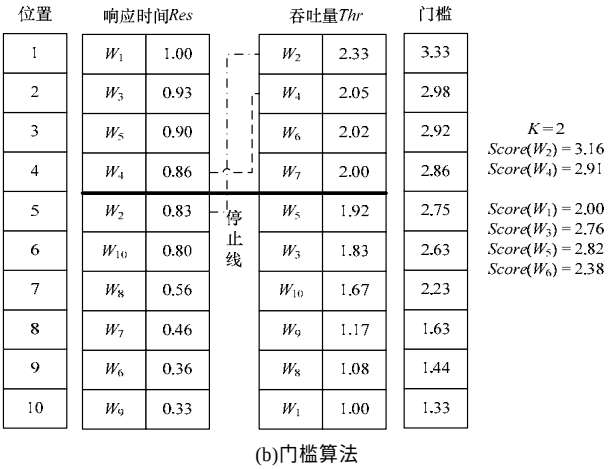
质量的优劣关系决定服务的从属关系。质量综合评价分数高，则表示该服务更优。

3 Top-K 查询

假设成立如下前提：在关键字查找得到的服务集中，所有服务都已经附有质量数据。将各个独立质量指标存储为按照规范后分数由高到低的顺序表，同时构建索引记录服务在各表的位置，如图 1 所示。这就建立了数据模型。



(a)FAGIN 算法



(b)门槛算法

图 1　Fagin 算法和门槛算法

3.1　传统算法

蛮力的 Top-K 查询是访问所有 Web 服务，逐一计算它们的质量评价分数，再取最高的前 K 个结果即可。这种方法时间复杂度较高。事实上由穷举剪枝的思路，只要减少可能解的搜索空间就可以提高算法的效率。

由于各个质量指标的表已经排好序，因此可以找到某个位置 L，使得在 L 以上的表搜索空间即可找到 K 个解，而 L 以下的服务，质量评价分不可能超过 L 以上的服务，不必再访问。Fagin 算法和门槛算法[4]就是基于上述思想。前者通过顺序访问确定该位置，而后者则计算各表中该位置的分数并加和作为门槛值来确定停止条件。门槛值的计算提高了 L 的位置，缩小了 Top-K 解搜索空间，算法会更早收敛。

图 1 是一个只考虑 2 个质量参数的例子：从 10 个 Web 服务中找出前 2 个(K=2)质量较好的。Fagin 算法总共访问了 14 次，而门槛算法则共访问了 12 次。算法的详细介绍可参阅文献[4]。

仔细观察 Fagin 和门槛算法，它们确定的位置其实都是在一个水平线上。而处在这条线上的表项并不是都已经访问过。如果称这条线为“停止线”，那么 StopLine 算法的动机在于是否能将其向上推进，并制造新的门槛值，加快算法收敛。

3.2　StopLine 算法

如前文所述，StopLine 算法的关键就在于：需要找到新的停止线，并且计算更快使算法收敛的门槛值。从直觉上说，这个门槛值应该越低越好，这样就能很快地找到 K 个比它高的结果。

具体的，可以在各表中记录这样的位置 s：在 s 前面的表项都已经访问过。这个位置 s 称为“最大访问位”。因为有随机访问，s 是跳跃式的增长，各表中该位置上数值之和变小的幅度也就会较大，以此作为新的门槛值，算法就会更快收敛。收敛时各表中当前访问到的位置称为该表的“停止位”。停止位不会超过最大访问位。将各表的停止位连在一起，就构成了算法的停止线，如图 2 所示。

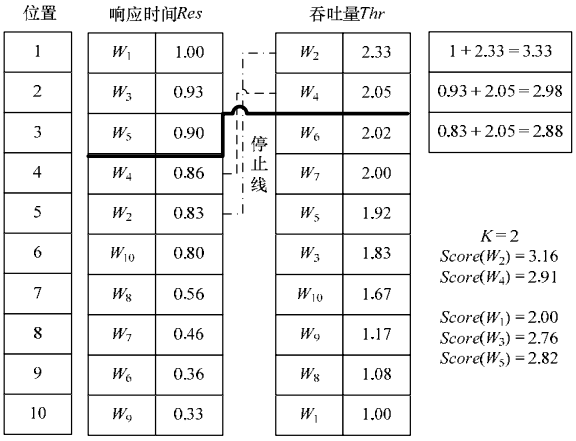


图 2　StopLine 算法

对比图 1 与图 2 可以看到，StopLine 算法的停止线位置更高，*Res* 表和 *Thr* 表的最大访问位分别为 5 和 2，新门槛值为 2.88。访问次数为 10 次。

以下是 StopLine 算法伪代码(为了简化，这里只考虑 2 个质量指标，实验会使用前述全部指标)：

输入　响应时间表 *resList*，吞吐量表 *thrList*

输出　质量最好的前 K 个 Web 服务

(1)若服务个数 n<=K，则返回全部服务；

(2)resList.MaxAccess = 0；

(3)thrList.MaxAccess = 0；

(4)int resCursor = 1；

(5)int thrCursor = 1；

(6)访问 resList[resCursor]，对应服务为 W_i，记录分值 $Score_{res}(W_i)$；

(7)根据索引访问 W_i 在 thrList 表中的项并记录分值 $Score_{thr}(W_i)$；

(8)记录服务 W_i 的总分，$Score(W_i)=Score_{res}(W_i)+Score_{thr}(W_i)$；

(9)查找结果集 R，如果结果数小于 K，直接插入，若存在分值小于 $Score(W_i)$的结果，则替换它，否则丢弃 W_i；

(10)更新 resList.MaxAccess 和 thrList.MaxAccess；

(11)更新 StopScore=resList.MaxAccess.Score+thrList.MaxAccess.score，若 R 中存在 K 个结果且分数都大于或等于 StopScore，算法结束，返回 R，否则接着执行下一步；

(12)resCursor=resList.MaxAccess+1；

(13)thrCursor=thrList.MaxAccess+1；

(14)访问 thrList[thrCursor]，对应服务为 W_j，接着做与第(6)步~第(13)步类似的操作，对象改为 W_j 即可；

(15)跳到第(6)步。

针对图 1 和图 2 中同样的问题，图 3 给出了 StopLine 算法的执行过程。图中 *MA* 为最大访问位 *MaxAccess* 的简写。伪代码第(6)步~第(15)步是轮流访问 2 个表。更新最大访问位 *MA* 可以通过在每个表项增加一个位(bit)的标志位来实现，0 表示未访问过，1 表示已访问。在表中一直向下查找，直至遇到访问标志位为 0 的位置即为当前最大访问位。至于结果集 *R* 可以简单地用栈或排序单表实现。算法的正确性证明很简单，这里省略。

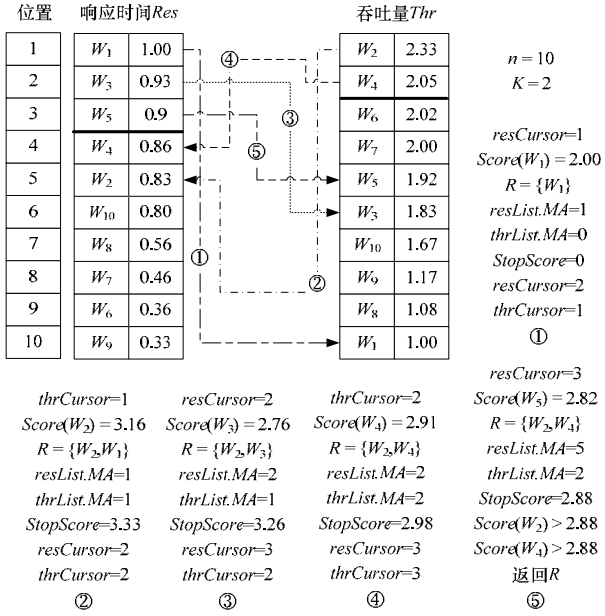


图 3　StopLine 算法示例

4　实验结果

算法采用 C++语言实现，CPU 为 AMD Athlon 64 X2 TK-57 双核，2 GB 内存。实验的数据集从实际数据集 QWS_Dataset[5]中选取。另外，不关注基于传统信息检索技术的关键字查找效果(省略了查全率和查准率)，而重点考察服务质量的评估效率。图 4 显示的是当 Web 服务个数为 2 505 时，选取 *K* 个质量最好的服务所用的时间。可以看到，*K* 值在 17 之前，StopLine 算法性能最优秀。但是在后面几个点相对落后。其中的原因是数据集较小而 *K* 值较大时，StopLine 在更新结果集时相对需要花费更多的比较时间。鉴于服务总量不多，图 5 显示的是 *K* 值为 2 时，不同 Web 服务数量情况下算法的性能，可以看到 StopLine 算法是最优的，且 Web 服务越多，其优势更加明显。

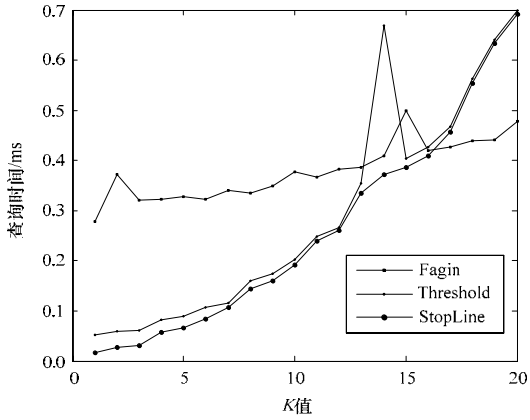


图 4　不同 *K* 值的算法运行时间

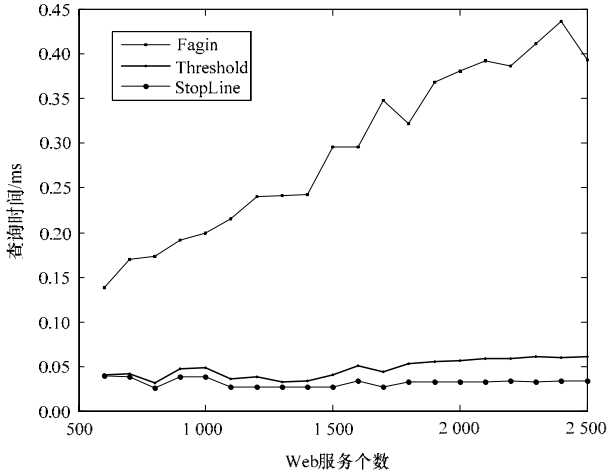


图 5　不同 Web 服务总量下算法运行时间

5　结束语

本文针对 Web 服务发现领域的问题，提出了重点考虑服务质量的 Top-K 查询方案。查询的 StopLine 算法充分利用类似于剪枝的思想，找到更早的停止线，缩小解搜索空间，达到收敛速度更快的效果。

参考文献

[1] Ran S. A Model for Web Services Discovery with QoS[J]. ACM SIGecom Exchanges, 2003, 4(1): 1-10.

[2] Ilyas I, Beskales G. Soliman M. A Survey of Top-K Query Processing Techniques in Relational Database Systems[J]. ACM Computing Surveys, 2008, 40(4): 1-58.

[3] 龚小勇, 朱庆生. 支持 QoS 的 Web 服务选择模型的研究与实现[J]. 计算机工程, 2008, 34(24): 55-57.

[4] Fagin R, Lotem A, Naor M. Optimal Aggregation Algorithms for Middleware[J]. Journal of Computer and System Sciences, 2003, 66(4): 614-656.

[5] Al-Masri E, Mahmoud Q H. Discovering the Best Web Service[C]//Proc. of the 16th International Conference on World Wide Web. Banff, Alberta, Canada: [s. n.], 2007: 1257-1258.

编辑　顾逸斐