基于深度优先的分步分治算法研究

郭双宙,徐济惠
(宁波城市学院,浙江 宁波 315100)

摘 要:Top-k 服务组合问题对于学术界和工业界来讲,都有实际的研究意义和应用场景。文中分析了理解 top-k 问题的关键所在解图,提出了基于深度优先的分步分治算法进行服务组合。该算法对用户请求的输出参数分别进行求解,该过程可并行处理,在求解结束后再进行合并。该方法可以支持分布式、并行处理框架,从而在应对大规模的服务集合时,能快速、高效地提供满足用户需求的组合服务;提出了通过构造解图的方法进行搜索求解,通过求解"关键路径"和"非关键路径"与合并"关键路径"和"非关键路径"得到解图。

关键词:服务组合;关键路径;服务筛选;top-k
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2014)06-0131-05
doi:10.3969/j.issn.1673-629X.2014.06.033

Research on Algorithm Based on Depth First Search and Task Partition

GUO Shuang-zhou, XU Ji-hui
(Ningbo City College, Ningbo 315100, China)

Abstract: Top-k service composition problem has the actual research significance and application scenarios for academia and industry. Analyze the key solution graphs of top-k problems in this paper, propose an algorithm based on depth first search and task partition. This algorithm first solves the user request separately which can be parallel processing, and then merge this solutions to provide a complete solution for the user request. This algorithm can support distribute and parallel framework, quickly and efficiently provide the service composition satisfies the user need when facing large scale service sets. Propose the method to construct solution graphs to search solutions, through solving critical path and no-critical path and merge critical and no-critical paths to gain solution graph.

Key words: service composition; critical path; service filter; top-k

0 引 言

近年来,Web 服务因其良好的跨平台性、可重用性得到了越来越广泛的应用。而通过组合一系列的 Web 服务来完成单个 Web 服务所不能完成的任务,也得到了学术界和工业界的广泛关注。文中算法关注于大规模 Web 服务情景下找出响应时间最优 top-k 解。

1 现有算法的分析与启示

1.1 基于图的服务组合算法

用图来做服务组合的方法中,国内文献的论述中有基于二分图、基于与或图、基于规约图,还有基于参数推导图,以及 ASC 图的,这些方法都用了特定的图模型去求解服务组合问题,特别是 ASC 图的方法对加快搜索效率有很好的效果[1]。

国外文献中,有论述服务本体关系图的基于接口匹配的 Web 服务自动组合方法,该方法要解决图的规模过大所造成的搜索空间膨胀问题[2]。使用 UML 图来表述以流程为中心的服务组合问题。文献[3-4]对图方法做了细致的论述,有很大的理论参考价值。将服务组合问题建模为 Petri 网络模型[5]为服务组合提供了新的思路。将服务组合问题转化为图搜索问题,再使用类似 A * 搜索算法[6]查找组合方案,也很值得借鉴。在服务组合框架方面,文献[7-8]也做了论述。

该方法在搜索的某一层次,启发式地选取最有可能的服务进行搜索。所以该方法的效率与服务的启发选取有关。已知的该方法进行实验的服务集合比较小,所以难以推断在大规模服务集合下的性能表现。但是,因为该方法对组件服务有多次转换过程,所以,面对大规模的服务集合,转换过程的花费将是很大的。

收稿日期:2013-09-02 **修回日期**:2013-12-08 **网络出版时间**:2014-02-24
基金项目:宁波市自然科学基金资助项目(2010A10125,2012A610063);宁波市科技局创新科学基金资助项目(2011B710034)
作者简介:郭双宙(1963-),男,上海人,硕士,副教授,研究方向为软件开发方法、软件体系结构。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140224.0922.056.html

1.2 回溯树算法

在文献[9]中,提出了一种基于树结构的算法进行服务组合。基于回溯树的服务组合算法 CBCT,将 Web 服务转化成一条条规则,其中,规则的输入与 Web 服务操作的输入一样,规则的输出是 Web 服务操作的某一原子输出,所以,如果一个 Web 服务操作有 4 个输出,则会相应地转化为 4 条规则。回溯树的主要构建过程可表述为:对用户的单个请求,构造为一个节点,找到能产生该节点元素的规则,规则作为有向边添加,边的输出端指向该节点,同时在边的输入端添加新的节点作为原节点的子节点而存在,子节点的元素由父节点和边所关联的元素共同决定,同时,边和子节点的添加需要满足不产生环的要求,否则放弃添加。重复这个过程直到无法扩展为止。如果有解,则必包含在构造的回溯树中。

对回溯树算法的实际编程实验,验证了以下结论:首先,回溯树是按层次来进行构造的;其次,回溯树算法保证树的构造过程中不产生环。

但是在大规模 Web 服务情景下回溯树算法表现并不理想。经过分析,大致有如下原因:

(1)规则过多。回溯树算法将 Web 服务转化为一条条比较细致的规则,这导致规则比较庞大。

(2)时间效率不理想。算法执行过程中,对每个点的扩展都要遍历所有的规则。规则多了之后,时间效率不理想。

(3)空间效率不理想。算法按层次构造求解树,理论上当一棵完备的树构造完成,所有的解也包含在其中。但是,在大量的服务情形下,按规则匹配的服务很多,造成这棵完备树的构造出现内存消耗过大,容易抛出内存不足的异常。

1.3 基于二叉树的算法

在文献[10]中,提到了一个基于二叉树的服务组合算法。

在此二叉树模型中,节点的左子树放置该节点的前驱条件服务集合,右子树放置节点的子服务。左子树上的节点是'或'的关系,右子树上的节点是'与'的关系。为了加速执行效率,在所有的输出元素上建立了倒排索引,加速前驱条件的查找。同时,算法执行过程中,对求解成功和失败的节点也建立索引,对后续的树的建立节省了时间与空间[11]。

回溯树与二叉树的算法,都能保证构造一棵完备的求解树,而且都是单机的串行化算法。但不同的是,二叉树是二叉的,而回溯树则是多叉的;回溯树运用队列结构,属于按层次回溯构造,二叉树则运用栈结构,进行类似深度优先的构造;回溯树能防止构造的过程中产生环或回路,二叉树则不能检测环或回路,只能对建树的高度进行控制,避免有环的时候造成构造的树深度无限。

2 基于深度优先的分步分治算法

基于深度优先的分步分治算法的主要思想有:

(1)对用户请求的众多输出请求,每一个单独求解,求解结束再合并。

(2)关键路径和非关键路径上公共部分的合并。

(3)在解的结构确定的情况下,对备选的服务进行筛选,只求解包含 top-k 解的最小解集合,避免求全解。

(4)由具有共同前驱和后继的一系列服务形成的组合服务,对解路径进行化简。

2.1 关键路径和非关键路径

关键路径定义:关键路径是建立在服务层次这一概念上的,它基于一个基本的认识:对于一个层次为 n 的服务,其某些输入"必然"只能由层次为 $n-1$ 的服务提供。这样,不断去找这种"必然"的服务,一定能找出至少一条从层次为 n 的服务到用户输入起点的路径。而这条路径就叫关键路径。

关键路径上任意两个相邻的服务,其服务层次的差别为 1。

非关键路径定义:非关键路径是相对于关键路径而言的,则非关键路径可以简单的认为是这样的路径:路径上存在两个相邻的服务,其服务层次的差别大于 1。

在图 1 中同时出现了关键路径与非关键路径。服务层次为 4 的服务 W_4,由它出发,有 3 条路径。其中,关键路径一条,为 $\{W_4, W_3, W_2, W_1\}$,非关键路径有两条,分别为 $\{W_4, W_{2.1}, W_{1.1}\}$ 和 $\{W_4, W_{1.2}\}$。

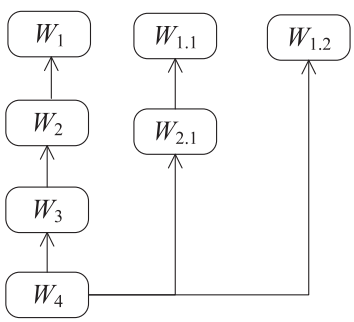

图 1 图路径

求解关键路径:按层次递减 1 的规律求解即可。以图 2 为例,以 W_4 为起点求解关键路径。首先,找出 W_4 的所有直接前驱,它们是 $W_3, W_{2.1}, W_{1.2}$。其中只有 W_3 是和 W_4 相差一个服务层次的,所以,将 W_3 加入到路径之中。然后,找出 W_3 的直接前驱,即 W_2,W_2 和 W_3 也是相差一个服务层次,于是将 W_2 加入到路径之中。最后,找到 W_2 的前驱 W_1,因为 W_1 的前驱是用户的输入,

所以,停止寻找。至此,关键路径已经解出。

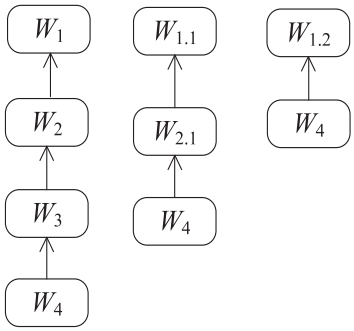

图 2　关键路径和非关键路径

求解非关键路径:在求解完关键路径后,有可能在关键路径上的某些服务其输入并不是完全被满足的,这个时候,对这些不能被满足的输入,需要再次求解,也就是求解非关键路径。仍以图 2 为例,在求解完关键路径后,W_4 被满足的输入是由 W_3 提供的,W_4 还有部分输入未被满足。输入没有被完全满足,意味着服务是不能被触发的,所以,必须求解非关键路径。求解非关键路径的基本方法和求解关键路径的方法只有一点差别,即不再严格按服务层次递减 1 来求解,非关键路径上的服务允许相邻的服务层次差别大于 1[12]。

路径上的跨层次依赖:在关键路径和非关键路径上都可能存在跨层次依赖,所谓跨层次依赖是指,在一条路径中,某个服务的某些输入没有被相邻的服务输出满足,但是,该路径上的其他服务能提供这样的输出。示例如图 3 所示。图中,跨层次的依赖以曲线表示。此外,跨层次的依赖还有可能存在于路径之间。示例如图 4 所示。

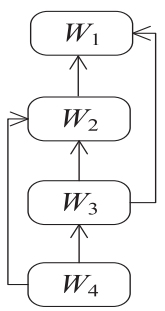

图 3　跨层次依赖

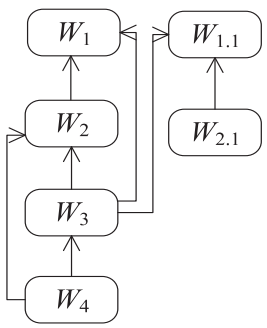

图 4　路径之间的跨层次依赖

在同一路径中解决跨层次的依赖相对好做一点,当某服务有尚未满足的输入时,只需查找该路径的其他服务,看它们是否能够提供一致的输出即可。而当跨层次依赖出现在路径之间时,解决办法就不那么显而易见了。因为,算法设计的出发点是基于深度优先的路径搜索。算法使用栈来存放中间搜索遇到的节点(服务),当搜索成功时,栈中存放的节点实际上是“一条”路径。求解关键路径的栈,将之称为“主栈”,求解非关键路径的栈称之为“次栈”。主栈只有 1 个,但次栈却可能有多个。由于主栈和次栈是分别求解的,所以,如果主栈和次栈之间存在路径上的跨层次依赖,若不做处理,虽然服务都找全了,但是丢失了依赖关系[13],这个问题将在下面描述。

2.2　解　图

解图:将所有关键路径和非关键路径合并,就得到了解图。解图完整地描述了从输入到输出必须经历的节点和这些节点之间的调用关系。

合并路径时是将次栈的内容合并到主栈中,即主栈中没有的内容要从次栈中补充,主次栈都有的内容不做处理,并遵循了这样的原则:具有相同服务层次的服务放置在栈的同一深度位置上。

从解图找寻解,比上一节中的从关键路径和非关键路径中找寻解有明显的优势。因为从关键路径和非关键路径中找解,要遵循这样的过程:

1)从关键路径中提取一个解,姑且称为主解。

2)以主解中含有的服务为前提,从非关键路径中找寻解,姑且称为次解,将主解和次解合并,得到最终解。

前面说过,非关键路径可能有多条,所以,主解和次解合并需要做很多工作—繁琐且容易出错。在解图中找解,不需要做上述的任何假设。

在解图中,节点中备选服务的数量决定了一个解图中可蕴含的解的个数。具体来说,一个解图包含的解个数等于每个节点备选服务个数的乘积。

2.2.1　解图中服务的选取

在一个解图中找出 top-k 最优解时,应努力避免求解图中的所有解。比较好的做法是:在不影响解精度的情况下,求解包含 top-k 解的最小的解集合。但是,并没有很好的算法去做到这点。

解决上述问题的算法描述如下:

1)将每个桶中的数字由小到大排序,每个桶中只留下一个最小的数字。此时得到一个解。

2)对每个桶而言,桶中其他的数字和最小的数字都存在一个大于等于 0 的差值,将这些数字收集起来,按差值由小到大的顺序放入优先级队列中。

3)每次从队列中取出一个具有最小差值的数字,并将其放入其原来的桶中。

4)计算当前状态下蕴含的组合个数,计算的方法与求解图中蕴含的解个数是一样的,即所有桶中含有数字个数的乘积。如果乘积小于 k,则转到第 3 步;如果乘积大于等于 k,则终止,所求的组合必从当前的桶含有的数字中组合产生[14]。

上述贪心算法能够很好的工作是因为在串行的结构中,计算方法是单纯的加法,即 $W_1+W_2+W_3+W_4+W_5$,这样具有小差值的数字必然优于具有大差值的数字。

虽然,上述的服务选取的算法只适用于串行情况,无法适用于既有串行又有并行结构的解图,但是,还是具有很好的启示:可以将该算法用于解图中的路径上,因为路径是串行的,而后,以路径为基础求解解图中的解,具体的论述请见第 2 节。

零差值的意义:差值为零意味着替换服务对 Qos 丝毫不影响,而任何差值不为零的服务,在替换后都可能影响整体的 Qos 值。具体的说,串行情况下,一定会影响整体 Qos 值,并行情况下,可能影响整体 Qos 值,所以,差值非零的替换需要重新计算整体的 Qos 值。而零差值对串行和并行结构的 Qos 计算结果不会有任何的影响。

2.2.2 解图中解的产生和验证

解图,从本质上讲,就是解的框架。这个框架中最重要的信息有两部分:由节点构成的路径和节点之间的调用关系。其中,路径是建立在节点调用关系基础上的。由于两个信息对所有的解是等同的,所以可以让它们作为模板而存在--所有的解都必须完全符合这个模板,解中缺失的信息也可以从模板中去补充。这样的模板对于解的产生和验证是至关重要的。

解图的解:在解图中,从每个节点的备选服务中选择一个服务,这些服务和它们间的调用关系就构成了一个解。

集合路径:解图中,由于每个节点都是由若干备选服务组成的集合,所以将由节点组成的路径称为集合路径。

元路径:集合路径中的每个节点都选择一个备选服务,就得到了由服务组成的路径,成为元路径。

解图的路径集合:在一个解图中,可以枚举其所包含的所有路径,得到路径集合。

解的 Qos 值计算:对某一个特定的解,将它带入路径集合中去进行 Qos 计算,这些路径集合中计算所得最大的 Qos 值就是这个解的 Qos 值。

综上所述,解的产生实际上有两个步骤:产生一个服务集合;计算这个服务集合的 Qos 值。

有两种方法去求解和验证:

1)key path 选取方法。

在一个解中,在众多从初始输入到最终输出的路径中,Qos 值最大的那个路径决定了解的 Qos 值,将这样的路径称为 key path。基于这一点认识,该方法首先选出一条路径,且认为其就是 key path。然后,对照解模板,找到 key path 缺失的服务,由 key path 中的服务和其缺失的服务,就构成了解的服务集合。最后,将这个服务集合放入到解图的路径集合中去计算 Qos,如果计算得出的 Qos 值和 key path 的 Qos 值一样,则得到一个解;如果不一样,说明 key path 选择错误,放弃这个解,并进入下一轮 key path 选取的过程中。

2)枚举每个解,直接验证。

依次列举解图中各种可能的服务组合,将服务组合带入解图的路径集合中去计算其 Qos,再通过模板补充服务间的调用关系,至此,就得到了一个解。

上面的两种方法,各有千秋。前者需要先完善服务集合,再进行计算;后者则直接进行计算。前者的 Qos 目标性很明确,所以其计算得到的前 k 个解也一定是 Qos 最优的 k 个解;后者无 Qos 目标,要求 k 个最优解,则需要全部计算完毕之后,从结果中再选出 k 个最优解。

2.3 组合服务

组合服务:在解图中,具有共同的前驱和后继的两个或多个服务,可组合成一个服务,即组合服务。组合服务保持了单个服务的前驱和后继关系,而其 Qos 值则是构成这个组合服务的多个服务中 Qos 值最大的那一个。化简后的解图路径数只有 6 条,相比于未化简之前的上百条,毫无疑问是极大的简化,如图 5 所示。

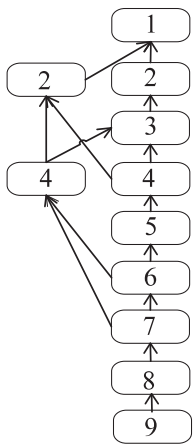

图 5 简化后的九层解图

再来看看化简后的数据。在未化简之前,第 7 层的三个节点中的备选服务和 Qos 见表 1。

表 1 可组合的备选服务

备选服务	Qos
serv1624499521	420
serv239131911, serv931815697	50, 220
serv1001247930, serv308564144	30, 200
serv1763363987, serv1693931754	380, 430

形成的组合服务和 Qos 见表 2。

表 2　组合服务

组合服务	Qos
{serv1624499521, serv239131911, serv1001247930}	420
{serv1624499521, serv239131911, serv308564144}	420
{serv1624499521, serv239131911, serv1763363987}	420
{serv1624499521, serv931815697, serv1763363987}	420
{serv1624499521, serv931815697, serv1001247930}	420
{serv1624499521, serv931815697, serv308564144}	420
{serv1624499521, serv239131911, serv1693931754}	430
{serv1624499521, serv931815697, serv1693931754}	430

从表 2 中,可以看出,最小的 Qos 为 420,而 Qos 为 420 的组合服务有 6 个,换言之,这 6 个组合服务可以相互替换而不会影响某个解的总体 Qos 值,因为它们之间的 Qos 差值为 0。

3　结束语

基于深度优先的分步分治算法最重要目标是要得到解图。解图中的节点和调用关系就如同地图中的站点和道路一样,得到解图就掌握了解的骨骼。得到解图,就可以抽取公共信息,构建解的模板,以合理的方式取得解,验证解。总而言之,解图是理解,也是求解 top-k 问题的关键。

参考文献:

[1] Rao J. Semantic Web service composition via logic-based program synthesis[D]. Norwegian: Norwegian University of Science and Technology, 2012.

[2] Thonel S, Depke R, Engels G. Process-oriented, flexible composition of web services with UML[J]. LNCS, 2013, 2784: 390-401.

[3] Hamadi R, Benatallah B. A Petri net-based model for web service composition[C]//Proceedings of the 14th Australasian database conference. Darlinghurst, Australia: Australian Computer Society, Inc, 2003: 191-200.

[4] Oh Seog-Chan, On Byung-Won, Larson E J, et al. Web services discovery and composition as graph search problem [C]//Proc of the 2005 IEEE international conference on e-technology, e-commerce and e-service. Hongkong: IEEE Computer Society, 2005: 784-786.

[5] Pathak J, Basu S, Lutz R, et al. Parallel Web service composition in MoSCoE: a choreography-based approach[C]//Proc of 4th IEEE European conference on web services. [s. l.]: IEEE CS Press, 2011: 1123-1127.

[6] 刘家茂,顾　宁,施伯乐. 基于 Mediator 的 Web Services 无回溯反向链动态合成[J]. 计算机研究与发展,2005,42(7):1153-1158.

[7] 刘永和,冯锦明,郭维栋,等. Delaunay 三角网通用合并算子及分治算法的简化[J]. 中国图象图形学报,2012,17(10):1283-1291.

[8] 谢增广. 平面点集 Delaunay 三角剖分的分治算法[J]. 计算机工程与设计,2012,33(7):2652-2658.

[9] 邓水光,吴　健,李　莹,等. 基于回溯树的 Web 服务自动组合[J]. 软件学报,2007,18(8):1896-1910.

[10] Zhou A, Huang S, Wang X. BITS: a binary tree based web service composition system[J]. Int J Web Serveice Res, 2007, 4(1): 40-58.

[11] Cormen T H, Leiserson C E, Rivest R L, et al. 算法导论[M]. 潘金贵,顾铁成,李成法,等,译. 第 2 版. 北京:机械工业出版社,2011.

[12] 刘晓峰,业　宁,李国宝. 基于分治策略的无线传感器网络屏障覆盖算法[J]. 计算机与数字工程,2012,40(10):22-25.

[13] 郭双宙,梁金兰. 构件库用户反馈子系统的客观反馈的设计[J]. 计算机技术与发展,2007,17(5):129-132.

[14] 黄创光,印　鉴,汪　静,等. 不确定近邻的协同过滤推荐算法[J]. 计算机学报,2010,33(8):1369-1377.

(上接第 130 页)

Fisherfaces: recognition using class specific linear projection [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 1997, 19(7): 711-720.

[12] 杨　健,杨静宇,叶　晖. Fisher 线性鉴别分析的理论研究及其应用[J]. 自动化学报,2003,29(4):481-493.

[13] Jing Xiaoyuan, Zhang D, Jin Zhong. UODV: improved algorithm and generalized theory[J]. Pattern Recognition, 2003, 36(11): 2593-2602.

[14] 杨　健,杨静宇,刘宁钟. 统计不相关最优鉴别分析的理论与算法[J]. 南京理工大学学报,2002,26(2):179-182.

[15] Martinez A M, Benavente R. The AR face database[R]. [s. l.]: CVC, 1998.

[16] Zhang D, Guo Z, Lu G, et al. An online system of multi-spectral palmprint verification[J]. IEEE Trans on Instrumentation and Measurement, 2010, 59(2): 480-490.

基于深度优先的分步分治算法研究

作者： 郭双宙， 徐济惠， GUO Shuang-zhou， XU Ji-hui
作者单位： 宁波城市学院,浙江 宁波,315100
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(6)