

基于缓存的复合数据服务更新优化方法

王家尧¹, 王桂玲², 张 鹏²

(1 山东科技大学 信息科学与工程学院, 山东 青岛 266590; 2 北方工业大学 云计算研究中心, 北京 100144)

摘 要: 复合数据服务是由具有层次结构的多个原子数据服务组合而成, 是实现跨域数据集成的一种手段. 当底层数据服务发生数据更新时, 数据更新如何自下而上经过多个中间复合数据服务传播到顶层, 并在这个传播的过程中保障顶层复合数据服务的更新效率就成为一个挑战问题. 针对这一问题, 提出一种基于缓存的复合数据服务更新优化方法. 该方法根据缓存点成本收益模型找出复合数据服务的最佳缓存点, 并且通过 B 树索引查找缓存点, 重用缓存数据, 避免重复的更新计算. 实验表明, 该方法提高了复合数据服务的更新效率.

关键词: 数据服务; 服务聚合; 服务优化; 缓存

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2013)03-0080-05

Cache-Based Optimization Method for the Composite Data Services

WANG Jia-yao¹, WANG Gui-ling², ZHANG Peng²

(1 College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; 2 Cloud Computing Research Center, North China University of Technology, Beijing 100040, China)

Abstract: A composite data service is composed of a number of basic data services with hierarchical structure. It is one of the important approaches for cross-domain data integration. Once a primitive data service is updated, it becomes a question of how to propagate the update bottom-up to the top level via several intermediate composite data services, and to ensure the update efficiency of the data view. This paper intends to present a cache-based optimization method for the composite data services, which is based on a cost-benefit model of cache points to find the best cache point of the composite data services. In addition, when the data services update, we can find the cache point quickly and efficiently to reuse the cached data through the B-tree index, that avoid repeat of the updated calculation. Experiments show that our approach can effectively improve the update efficiency of the composite data services.

Key words: data services; service mashup; service optimization; caching

1 引言

近年来出现了一些数据服务平台, 如 Yahoo Pipes^[1]、IBM Mashup Center^[2], 这类平台为用户提供了一种简便的方式来定义原子数据服务, 并通过对原子数据服务的组合生成复合数据服务, 数据服务^[3-4]托管运行在平台之上, 用户可以随时查看数据服务的执行结果. 这类平台有较好的易用性和灵活性, 但是, 它们在数据服务的更新优化方面存在挑

战. 当数据源发生更新时, 为了得到最新的数据集成结果, 平台往往需要重新执行相关数据服务, 使得数据更新从原子数据服务开始自下而上传播到顶层的复合数据服务. 事实上, 平台中托管着不同用户所建立的大量复合数据服务, 若两个复合数据服务(或其中间结果)是由相同的原子数据服务经过相同的操作组合而成的, 就可以互相重用彼此结果, 而无需重复执行. 如果平台不能够重用彼此结果就会造成过

收稿日期: 2012-06-15; **修回日期:** 2012-07-27

基金项目: 国家自然科学基金项目(60903048, 60970132, 61003294); 山东省自然科学基金项目(ZR2011FQ028)

多的重复操作,系统后台负载过大,复合数据服务更新延时,将严重影响更新效率,导致用户无法即时得到最新数据集成结果。

基于上述研究背景,本文拟提出一种基于缓存的复合数据服务更新优化方法。该方法首先对给定的一个复合数据服务找出更新代价最小的生成该复合数据服务的数据服务组合方案。然后,借鉴数据缓存成本效益的思想,选取成本效益最大的数据服务组合方案中间结果进行缓存,并且在数据服务更新时通过 B 树缓存索引^[5],快速有效的发现缓存点,重用缓存数据,避免数据服务重复执行,降低系统后台压力,提高复合数据服务更新效率。

2 问题描述

我们通过一个实例来分析问题,社区突发中毒和传染病,政府部门要合并这两种数据统计应急程度以及分配医疗资源。如图 1 所示,左图为判断事件严重程度的复合数据服务 DS_1 ,其输入是中毒和传染病数据,输出是应急事件的严重程度。该服务把中毒病例数据和传染病数据分别封装为数据服务进行组合,然后经过统计人数操作,以此判断事件的严重程度;右图为分配医疗资源的复合数据服务 DS_2 ,其输入是中毒和传染病数据,输出是分配医疗资源。该服务同样把中毒病例数据和传染病数据分别封装为数据服务进行组合,然后经过统计患病人在地操作,有效的分配医疗资源。随着事态发展,中毒和传染病两种数据源不断动态变化,数据视图应及时体现底层数据变化从而为政府决策人员提供正确的信息,所以数据服务需即时更新。图 1 中两种数据服务 DS_1 和 DS_2 虚框中底层操作都是一样,即两者等价。当数据更新时, DS_1 和 DS_2 虚框中重复执行相同的操作,严重影响数据服务的更新效率。我们可以设定一种方案把 DS_1 的虚框子树进行缓存,当 DS_2 虚框子树数据更新时,就不必重新执行虚框中操作,可以直接调用对 DS_1 的虚框子树的缓存。当数据服务组合

平台上存在很多和 DS_1 的虚框子树等价的子树时,这些数据服务都可以直接重用对 DS_1 的虚框子树的缓存,这样就避免了数据服务重复执行,从而降低系统后台压力,提高了复合数据服务更新效率。

3 生成数据服务优化组合方案

数据服务是以提供数据资源的访问为目的的软件服务,它提供了对数据资源的抽象,以确切的数据模式表示数据资源,对数据的质量、服务的质量以及其他非功能属性进行描述,它接收一组关于对该数据资源进行访问、查询、转换等“只读”请求的参数,并返回一个结果数据集。复合数据服务由原子数据服务(即直接包装数据资源得到的服务)和其他复合数据服务组合而成,是实现多源、异构数据集成的一种手段,具有层次结构。我们的前期工作定义了生成复合数据服务的 Create、Rename、Delete、Filter、Sort、Nest、Unnest、Group、Ungroup 以及并、交、差、连接的操作,从复合数据服务操作的角度来看,复合数据服务是由一系列操作组成。我们把复合数据服务操作的整体过程结构化为一棵树,称为“复合数据服务树”。每一次操作的中间结果都作为树的节点,树的根节点就是复合数据服务的最终结果,也就是以嵌套表格的形式反馈给用户所呈现的结果,例如:Yahoo! Pipes 中 Pipe 的建立流程,Pipe 的每一步操作作为树的节点,最后的整体流程即为 Yahoo! Pipes 树。

为了更好的寻找缓存点,重用复合数据服务的中间结果,我们首先对数据服务组合平台中的数据服务生成优化组合方案。通过启发式规则和数据服务等价变换法则^[6]生成若干输出结果相同的候选复合数据服务。在启发式规则中 Filter 操作尽可能先做,这是优化策略中最重要、最基本的一条。Filter 操作一般使计算的中间结果大大变小,它常常可使执行时间节约几个数量级。所以在生成输出结果相同的候选复合数据服务时,根据启发式规则,对于每个 Filter 操作,利用涉及 Filter 的规则,尽可能将 Filter 操作下推移向复合数据服务树的树叶方向。及早执行 Filter 操作可以减少对中间结果进行 sort 操作的代价。根据 Filter 操作的性质使 Filter 操作结合或交换其他操作,使多个 Filter 操作能同时执行。数据服务等价变换法则即数据服务交换率、结合率和分配率以及其他操作的性质等,在文献[6]已经介绍。根据上述的启发式规则和这些等价变换法则把我们正操作的复合数据服务生成输出结果相同的

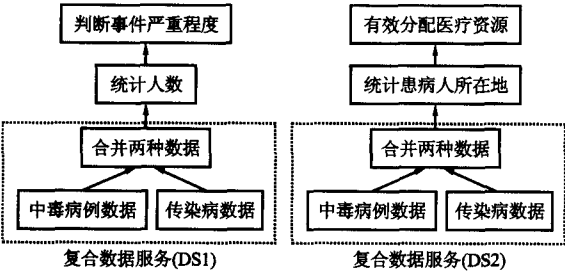


图 1 数据服务实例

候选复合数据服务,然后根据文献[6]把数据服务组合方案的更新优化问题建模为 0-1 规划问题,通过遗传算法得到更新代价最小的数据服务组合方案,把最优数据服务组合方案代替原来的数据服务组合方案,便于本文下面介绍的数据服务缓存点的选取和复用。

4 选取缓存点

我们把缓存点表示为复合数据服务树中出现的子树的计算结果,这个子树本身有一个或者多个调用数据服务以及数据服务操作的叶子节点.本节中缓存点的选择是权衡数据服务更新优化问题的关键,我们可以用公式具体描述缓存点的选择。

复合数据服务树的每一个操作节点都有潜力成为缓存点,所以建立一个可行缓存点集 $FcSet = \{Fc_0, Fc_1, Fc_2, \dots, Fc_N\}$, $FcSet$ (Feasible Cache-point Set) 的成员被称为可行缓存点,可行缓存点与数据服务组合平台的所有复合数据服务树的节点对应.考虑到复合数据服务执行时,当操作节点含有多个子节点时,其子节点所在的子树是并发执行的,所以其成本为本身成本和子树最大值成本之和,我们定义 $C(Fc_K)$ 为 Fc_K 本身的成本,即执行 Fc_K 时所用的时间,所以 Fc_K 以及其后代节点的累计总成本 TC (Total Cost) 为:

$$TC(Fc_K) = C(Fc_K) + \text{Max}(\sum C(Fc_H))$$

其中 $Fc_H \in (\text{Descendent}(Fc_K))$. Max 代表当 Fc_K 有多个子树时,取其中最大值成本的子树。

4.1 成本效益分析

对于 $FcSet$ 中的任一成员可行缓存节点 Fc_K ,其缓存数据可以被数据服务组合平台中的复合数据服务复用,避免了该节点以及其后代节点的重复执行.如果 Fc_K 没有被缓存,那么单位时间内 Fc_K 的请求总成本为 $RR(Fc_K) \times TC(Fc_K)$,其中 $RR(Fc_K)$ 表示 Fc_K 的请求频率,即在数据服务组合平台上用户对所有复合数据服务的单位时间内执行 Fc_K 次数。

可行缓存点 Fc_K 的成本还与外部数据源的更新频率和更新数据量密切相关,我们称其为一致性成本.一致性成本用于外部数据源数据更新时维护 Fc_K 输出数据的一致性. Fc_K 的缓存数据随着外部数据源的更新而改变,这就需要对 Fc_K 及其后代节点进行重新计算,因此,单位时间内 Fc_K 的一致性成本为 $UR(Fc_K) \times Sv(Fc_K) \times TC(Fc_K)$,其中 $UR(Fc_K)$ 代表 Fc_K 所有外部数据源数据更新频率的总和,

$Sv(Fc_K)$ 为外部数据源更新的数据量。

通过以上分析,我们用公式 $(RR(Fc_K) \times TC(Fc_K) - UR(Fc_K) \times TC(Fc_K)) / Sv(Fc_K)$ 权衡 Fc_K 的成本效益 ($WCB(Fc_K)$),从 $WCB(Fc_K)$ 公式中可以看出,可行缓存点请求总成本越大一致性成本越小,他的成本效益就越高,所以 $WCB(Fc_K)$ 在本文中被用来量化 Fc_K 作为缓存点所节约的成本。

4.2 缓存点选取算法

我们统计出所有已经被收集的可行缓存点的请求频率和更新频率,并计算出相应的累计成本,对于数据服务组合平台上的复合数据服务选取缓存节点的算法如下:

算法开始于复合数据服务起始操作节点,假设算法开始于可行缓存节点 Fc_K ,如果该节点正在被搜索称为当前的搜索点 (CSP),计算 $WCB(CSP) = (RR(CSP) \times TC(CSP) - UR(CSP) \times TC(CSP)) / Sv(CSP)$. 然后对比当前节点的 $WCB(CSP)$ 与其祖先节点的 WCB 以及后代节点的 WCB (如果 Fc_K 有多个后代节点,这里是指所有后代节点 WCB 的最大值). 如果祖先节点 WCB 值高于 Fc_K 节点的,那么把祖先节点初始化为新的 CSP、以这个可行缓存点向上继续搜索算法. 搜索的终止条件为:达到一个或多个节点各自的后代和祖先节点的 WCB 值低于他们本身自己的 WCB 值,这些可行缓存点在搜索终止时被选择作为缓存点. 如果复合数据服务起始操作节点的 WCB 大于其祖先节点的 WCB ,起始操作节点被选择作为缓存点,另一方面,如果复合数据服务最后的操作节点的 WCB 大于其后代节点的 WCB ,最后的操作节点被选择作为缓存点. 以同样的方式搜索每个复合数据服务来发现所有的缓存点,用这些缓存点组成缓存点集 $BpSet = \{Bp_1, Bp_2, \dots, Bp_N\}$.

5 缓存点复用原理

在数据服务组合平台中判断数据复用的缓存点并不是一个简单的任务. 某个缓存点 Bp_K 的结果可以被新的复合数据服务 Ds_i 所复用,当且仅当 Bp_K 的子树和 Ds_i 的子树精确匹配. 所谓精确匹配是指输入的复合数据服务的子树 Ds_i 和 Bp_K 中的子树具有完全相同的结构,并且两个子树的算子以及它们的参数也相同。

由于数据服务组合平台托管大量用户构造的复合数据服务,因此我们需要一种可扩展的机制来判断是否一个新的复合数据服务中存在一个或者多个

子树匹配现有的缓存点,为此我们设计一种缓存点索引模型来解决这个问题. 如果一个新的复合数据服务 DS_i 匹配现有的缓存点,那么 DS_i 做如下修改:对每个匹配现有缓存点的子树,通过引用对应这个缓存点的缓存数据替换这个子树,例如,如果 DS_i 中一个任意子树 $Subt$ 匹配一个现有的缓存点 B_{PK} ,那么 $Subt$ 被一个引用 B_{PK} 缓存点的缓存数据所替代,修改后的复合数据服务再被发送给数据服务组合平台去运行. 图 2 说明了一个新的复合数据服务是如何通过替换来复用缓存数据的过程. 我们使用 B 树结构建立索引缓存点.

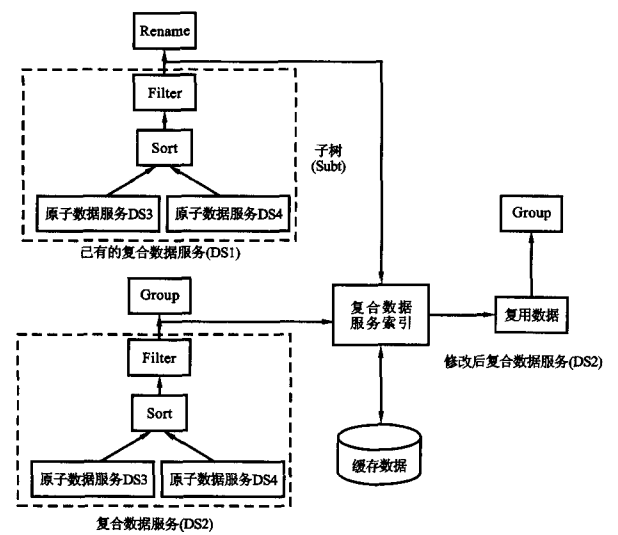


图 2 数据服务组合平台中缓存点复用原理

6 实验设计与结果

本文实验基于我们前期开发的数据服务建模和数据服务组合的工具^[7],并集成了本文提出的基于缓存的复合数据服务在线更新方法. 实验模拟了一个应用于复合数据服务环境下的数据服务组合平台,从 syndic8 (Barr, 2008)^[8] 数据仓库中提取 80 个数据封装成 500 个数据服务组成实验数据集,数据服务组合平台类似于 Yahoo! pipes,在数据服务组合平台上可以对数据服务进行 Create、Rename、Delete、Filter、Sort、Nest、Unnest、Group、Ungroup 以及并、交、差、连接等操作,用户建立的复合数据服务在数据服务组合平台上启动执行,并把执行后的最终结果以嵌套表格的形式呈现给用户. 数据服务执行时,与数据源建立一个连接提取数据,记录下提取数据的通信时间,对提取的数据进行一系列操作,并记录操作的执行时间.

我们的实验主要目的是评估动态缓存点的选取对数据服务组合平台的影响,我们分别分析了在请求频率、更新频率、数据量等参数变化情况下动态缓存点的选取对数据服务总成本的影响,即对数据服务执行时间的影响. 另外我们还实验了请求频率的变化对索引访问时间的影响.

6.1 评估动态缓存点的选取方案

在实验中,我们测试动态缓存点选取方案的成本,同时对比其他两种动态缓存点选取方案:一种是最终结果缓存即只缓存组合数据服务的最终结果,另一种是无缓存即组合数据服务不存在任何形式的缓存.

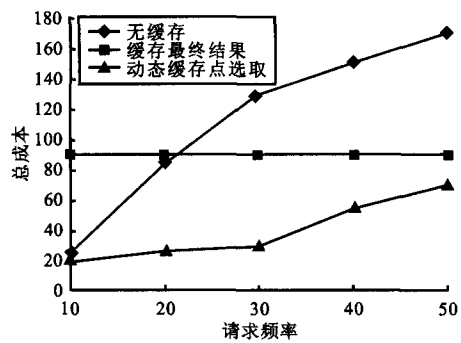
图 3(a)是实验改变请求频率时对三种方案数据服务总成本的影响,实验结果表明:当请求频率在相同范围时,动态缓存点选取算法的成本比其他两种方案的成本低,最终结果缓存方案的成本是一个常数,其成本主要是输入数据源发生改变时对最终结果的重计算,不需要额外计算缓存数据. 当请求频率很低时无缓存方案的成本类似与动态缓存点选取算法方案,然而,无缓存方案成本随着请求频率的增加迅速增加,而动态缓存点选取算法方案虽然也会随着请求频率的增加而增加,但是他不会无限制增加,当成本接近与最终结果缓存方案成本时他的增加曲线会变得平坦.

图 3(b)为实验数据源的更新频率的变化对三种方案数据服务总成本的影响,实验结果显示:动态缓存点选取算法方案优于其他两种方案,无缓存成本方案输入数据源变化时无缓存数据需要重计算,所以其成本为常数.

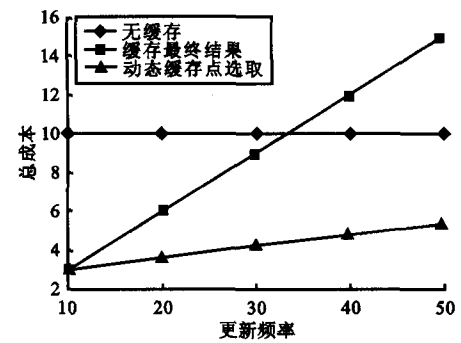
图 3(c)是实验数据源更新数据量的变化对三种方案数据服务总成本的影响,实验结果显示:动态缓存点选取算法方案优于其他两种方案,其他两种方案的成本高于动态缓存点选取算法方案.

6.2 复合数据服务索引分析

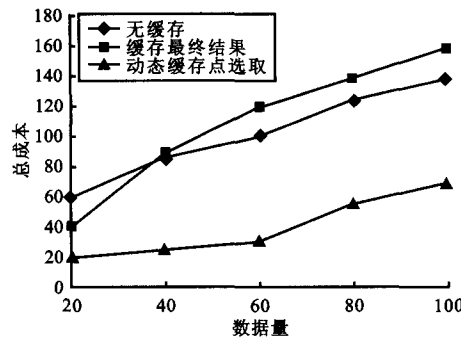
实验中,我们分析请求频率对索引访问时间的影响. 图 4 表明:索引访问时间随着请求频率的增加而减少. 这主要是因为当请求频率增加时,缓存索引会趋向于选择靠近复合数据服务树根节点近的缓存点,当离根节点近时,树的宽度会相对应缩小,缓存点的数量也会减少,请求频率越高,缓存点越接近于根节点,系统越快的寻找到匹配缓存点. 这个结果表明请求频率越高,响应就越快,从而缓存索引大大的提高了复合数据服务的更新效率.



(a) 改变请求频率对数据服务总成本的影响



(b) 改变更新频率对数据服务总成本的影响



(c) 改变更新数据量对数据服务总成本的影响

图 3 三种参数变化对数据服务总成本的影响

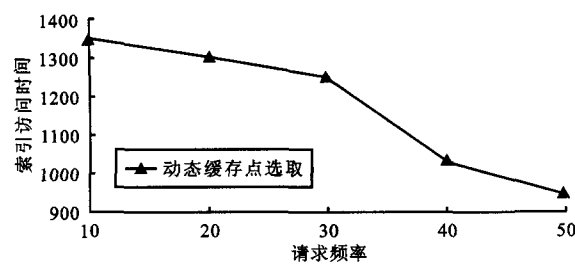


图 4 请求频率对索引访问时间的影响

7 结束语

基于缓存的复合数据服务更新优化问题是一个

亟待解决的问题,现有的传统工具还没有一个令人满意的解决方案.本文提出了基于缓存的复合数据服务更新优化方法.该方法能够利用启发式规则和数据服务等价变换法则快速把复合数据服务变换成候选的复合数据服务方案,通过遗传算法找出最优组合方案来替换被优化方案.同时利用缓存机制,缓存点成本收益模型和缓存索引快捷有效的解决复合数据服务的更新优化,提高了数据服务更新效率.

参考文献:

[1] Yahoo Inc. Yahoo pipes [EB/OL]. (2007-11-20). <http://pipes.yahoo.com/>.

[2] IBM Inc. IBM Mashup Center [EB/OL]. (2008-11-19). <http://www.ibm.com/software/cn/info/mashup-center/>.

[3] Di Lorenzo G, Hacid H, Paik H, et al. B. Data integration in mashups[J]. ACM Sigmod Record, 2009, 38(1), 59-66.

[4] Michael J Carey. Declarative data services: this is your data on SOA[C]// IEEE International Conference on Service-Oriented Computing and Applications. California, USA, 2007.

[5] Hassan O A, Ramaswarny L, Miller J A. The MACE approach for caching mashups[J]. International Journal of Web Services Research, 2010, 7(4): 64-88.

[6] 张鹏,王桂玲,季光,等.基于数据服务的数据组合视图的优化更新[J]. 计算机学报, 2011, 34(12): 2344-2354.

[7] Wang G, Yang S, Han Y. Mashroom: end-user mashup programming using nested tables[C]// Proceedings of the 18th international conference on World Wide Web. Spain;Madrid, 2009: 861-870.

[8] Barr J, Kearney B. syndic8 feeds repository[EB/OL]. (2001-03-15). <http://www.syndic8.com>.

作者简介:

王家尧 男,(1986-),硕士研究生.研究方向为服务组合、数据集成等.

王桂玲 女,(1978-),博士,副研究员.研究方向为工作流、数据集成、Mashup 技术等.

张 鹏 男,(1984-),博士研究生.研究方向为服务组合、工作流、数据集成等.