Journal of Computer Applications
计算机应用,2012,32(5):1432 - 1435
ISSN 1001-9081
CODEN JYIIDU
2012-05-01
http://www.joca.cn

文章编号:1001 - 9081(2012)05 - 1432 - 04
doi:10.3724/SP.J.1087.2012.01432

基于拓扑序列归约的 Web 服务组合 QoS 度量算法

李兴芳[1*],苑迎春[1,2],王克俭[1]
(1.河北农业大学 信息科学与技术学院,河北 保定 071001; 2.河北省农村信息化工程技术研究中心,河北 廊坊 065000)
(* 通信作者电子邮箱 boy_jnsh@163.com)

摘 要: 考虑有向无环图(DAG)描述的组合服务模型,提出了一种新的组合服务 QoS 度量方法——基于拓扑序列归约的 Web 服务 QoS 度量方法(QCMTSR)。其借鉴迭代归约度量方法中的基本结构及 QoS 计算公式,定义了 DAG 图中的两类基本结构,串归约结构和并归约结构,并给出了两种基本结构的 QoS 属性计算公式;通过逐步归约 DAG 图拓扑序列中的每个节点,直至最后一个节点的 QoS 属性值就是组合服务的各 QoS 属性的度量结果。从理论上证明了 QCMTSR 算法适用于所有 DAG 描述的组合服务,并实验证明 QCMTSR 算法对可靠性和可用性能够更准确的度量。

关键词: Web 服务组合; QoS 度量; 有向无环图; 拓扑序列归约; 面向服务架构
中图分类号: TP311.5; TP302.7 **文献标志码:** A

QoS computing method for Web services composition based on topological sequence reduction

LI Xing-fang[1*], YUAN Ying-chun[1,2], WANG Ke-jian[1]
(1. *College of Information Science and Technology, Agricultural University of Hebei, Baoding Hebei* 071001, *China*;
2. *Village Informatization Engineering Technology Reseach Center of Hebei Province, Langfang Hebei* 065000, *China*)

Abstract: In this paper, considering the Web service composition model described by DAG (Directed Acrylic Graph), a new Quality of Service (QoS) computing method for the composition service based on topological sequence reduction (QCMTSR) was proposed. Based on the basic structures and their QoS computing formulas of iterative reduction method two kinds of basic structures (i.e. serial reduction structure and parallel reduction structure) were defined in graph DAG, and their QoS calculation formulas were also given. During accessing each node step by step in the topology sequence for DAG. Repeating this process until the last node in this queue, then the QoS measure results of the last node were the computing results of the composition service. It has been proved that the algorithm can be applied to all the composition services described by DAG, and the experimental results show that the algorithm QCMTSR is more accurate in the measurement of reliability and availability.

Key words: Web service composition; Quality of Service (QoS) measurement; directed acrylic graph; topological sequence reduction; Service-Oriented Architecture (SOA)

0 引言

面向服务架构(Service-Oriented Architecture,SOA)的 Web 服务组合技术已成为目前构造分布式应用软件的一种主要方式[1],其基本思想就是通过 Internet 将分布在不同平台、公司中已存在的 Web 服务(基本服务),按照一定的业务逻辑动态地发现并组装成一个增值的、更大粒度的服务(组合服务),来满足不同用户的复杂应用需求[2]。随着 Web 服务数量的不断增长,不同服务供应者提供了多个具有相同功能、不同服务质量(Quality of Service,QoS)的服务,使得构建具有 QoS 保证的组合服务成为 Web 服务组合研究领域的一个关键问题。

QoS 用于描述 Web 服务的非功能属性,其参数设置涉及性能、可靠性、可提供性、正确性、完整性、费用、安全等各个方面,已提出的 QoS 度量参数有服务的执行费用(代价)、执行时间、可用性、可靠性和声誉等[3-11]。有 QoS 保证的组合服务就是通过 Web 服务组合技术构建的组合服务必须满足上述所指的各种 QoS 需求。为验证一个组合服务是否满足用户提出的 QoS 需求,必须已知组合服务的各 QoS 属性值。也就是说,组合服务的 QoS 属性值应该用某种度量方法给出。组合服务的 QoS 度量不仅是衡量组合服务是否满足用户全局 QoS 需求的依据,也是服务选择算法设计的前提和基础。

组合服务是由多个成员服务按一定的逻辑关系组合而成,其 QoS 度量必然不同于基本 Web 服务的度量,组合服务的 QoS 度量既与组成它的成员服务有关,也与这些成员服务间的逻辑关系有关[6-13]。目前,已有许多学者对组合服务的 QoS 度量方法进行了研究。研究方法大多依据成员服务的 QoS 属性和它们之间的逻辑关系,较为代表性的评估方法有两种:迭代归约度量法和关键路径度量法。迭代归约度量法主要用于工作流描述的组合服务模型,借鉴工作流的性能评价方法,运用求和、级乘、概率、最大、最小等策略给出顺序、并行、选择、循环等基本模型结构的 QoS 计算方法,然后对基本结构逐步归约得到组合服务 QoS 的计算结果[7-11]。这种度量方法有较强的数学理论基础支持,使用比较广泛,但只能用于能够迭代归约计算的服务组合模型。关键路径度量法常用于有向图描述的服务组合模型,首先找到组合服务的时间关键路径,然后利用关键路径给出组合服务的执行费用、执行时间、可靠性和可用性的度量结果。如 Zeng 等[13]采用状态图描述服务组合模型,通过将循环、分支结构展开和分解,转化

收稿日期:2011-10-27;修回日期:2011-12-22。 基金项目:国家自然科学基金资助项目(60873236);河北省自然科学基金资助项目(F2009000653);河北省教育厅科学研究项目(2010251)。

作者简介:李兴芳(1987 -),男,山东济南人,硕士研究生,CCF 会员,主要研究方向:计算机网络、数据库; 苑迎春(1970 -),女,河北保定人,副教授,博士,CCF 会员,主要研究方向:计算机网络、数据库; 王克俭(1971 -),女,河北保定人,教授,CCF 会员,主要研究方向:计算机人工智能。

状态图为多个有向无环图(Directed Acyclic Graph,DAG) 表示的服务执行路径; 再依据 DAG 的时间关键路径给出组合服务的执行时间、可靠性和可用性的度量结果,执行费用则采用所有节点求和的方法。金海等[14] 研究 DAG 描述的服务组合模型,依据时间关键路径给出执行时间的度量方法。这种度量方法的优点是计算过程比较简单且效率较高。但关键路径度量法仅是相对执行时间而言的"关键",对可靠性和可用性计算时仅考虑时间关键的节点,具有一定的片面性。

本文针对关键路径度量方法存在片面性的不足,在借鉴迭代归约度量方法思想的基础上,提出了一种新的组合服务 QoS 度量方法——基于拓扑序列归约的服务 QoS 度量算法 QCMTSR(QoS Computing Method based on Topological Sequence Reduction) 。算法首先利用拓扑排序将 DAG 中所有节点组成一个有序队列,然后逐步扫描队列中的节点,每一步中,依据当前访问节点的类型组成串归约结构或并归约结构(在 2.1 节中定义) ,利用两种基本结构 QoS 属性计算公式(在 2.2 节中定义) ,归约得到基本结构的 QoS 属性计算结果。重复这种过程,直到队列的最后一个节点。此时序列中最后一个节点的 QoS 属性值就是组合服务的各 QoS 属性的度量结果。

1　问题描述

本文中的服务组合采用 DAG 描述,记作 $G=\{V,E\}$,其中节点集 $V=\{1,2,3,\cdots,n\}$ ($|V|=n$) 表示组合服务应用中用到的所有成员服务的集合(这里的节点编号表示能完成该节点任务的抽象服务,并不指定具体 Web 服务实例) ; E 是有向边集合,表示成员服务间的控制或数据依赖关系,即对 $(i,j)\in E$, i 执行完后 j 才可执行(i,j 是节点编号,规定 $i<j$) 。图中没有前驱的节点称为入口节点,没有后继的节点称为出口节点。为算法实现方便,规定图中只有单一入口和出口节点 (可添加虚节点使其满足要求,虚节点不完成具体功能,QoS 属性不参与度量) 。图 1 是一个简单的组合服务 QoS 应用实例,其中的节点 1 和 n 分别是添加的唯一入口和出口节点。

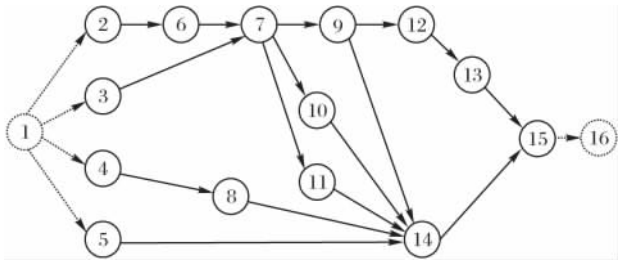

图 1　组合服务应用实例(DAG 表示)

对图中任一个节点 $i\in V$,都能找到一个 Web 服务实例完成相应的功能,假设能够完成该节点功能的 Web 服务实例名称为 S_i 。一个 Web 服务实例通常分为功能描述和非功能描述两部分,服务 QoS 用于描述 Web 服务的非功能属性[3] 。本文考虑服务的执行费用(PR) 、执行时间(DU) 、可用性(RE) 、可靠性(AV) 4 个参数。则对组合服务中的任一节点 $i\in V$,其对应的 Web 服务实例 S_i 的 QoS 属性可用一个 4 元组表示 $Q^i=\{PR_j,DU_j,RE_j,AV_j\}$ 。

在给出 QoS 属性描述之前,首先给出如下假设。1) QoS 相互独立性: 即对参与组合操作的各个成员服务而言,其 QoS 信息是相互独立的,一个服务的执行不会影响另外一个服务的 QoS 信息。2) 度量单位统一性: 即对参与组合操作的各成员服务而言,其 QoS 信息的度量单位是一致的,如各成员服务的执行时间都是以秒或毫秒等作为度量单位。表 1 给出了图 1 中各节点对应的 Web 服务实例名称及相应的 QoS 属性数据,其中执行费用的单位假定是元(人民币) ,执行时间单位是 s,可靠性、可用性是区间 [0,1] 上分布的一个小数。组合服务的 QoS 度量就是根据成员服务的 QoS 属性信息和 DAG 中的逻辑结构,通过某种计算方法得到组合服务的执行费用,执行时间,可靠性和可用性信息,从而完成对组合服务 QoS 的评估。

表 1　图 1 中各成员服务的 QoS 属性数据

节点	服务	费用/元	时间/s	可靠性	可用性
1	S_1	0	0	1.00	1.00
2	S_2	5	6	0.80	0.79
3	S_3	9	4	0.30	0.20
4	S_4	7	6	0.71	0.65
5	S_5	10	2	0.60	0.60
6	S_6	8	1	0.50	0.50
7	S_7	9	10	0.81	0.80
8	S_8	6	3	0.70	0.71
9	S_9	4	5	0.60	0.60
10	S_{10}	17	15	0.99	0.95
11	S_{11}	9	6	0.84	0.83
12	S_{12}	6	9	0.82	0.81
13	S_{13}	18	20	0.98	0.97
14	S_{14}	12	15	0.70	0.64
15	S_{15}	4	10	0.82	0.83
16	S_{16}	0	0	1.00	1.00

2　基于拓扑序列归约的服务 QoS 度量算法

基于拓扑序列归约的 Web 服务 QoS 度量算法(QCMTSR) 设计的基本思想: 利用拓扑排序将图中所有节点组成一个有序队列; 通过逐步扫描队列中的节点,将图中的节点划分为两种结构: 串归约结构或并归约结构,再依据两种基本结构的 QoS 计算公式,得到当前访问节点的 QoS 度量结果,重复这种过程,直到队列的最后一个节点。此时序列中最后一个节点的 QoS 属性值就是组合服务的各个 QoS 属性的度量结果。

下面首先给出该算法的相关定义,然后介绍两种基本结构的各 QoS 属性计算公式,最后给出算法的具体描述。

2.1　相关定义

定义 1　在 DAG $G=\{V,E\}$ 中,对 $\forall i\in V$,若其前驱节点为空,则该节点称为 QoS 可归约节点。

例如图 1 中的节点 1 是 QoS 可归约节点。

定义 2　在 DAG $G=(V,E)$ 中,对 $\forall i\in V$,若其所有的直接前驱节点都是 QoS 可归约节点,则节点 i 记为 QoS 可度量节点。

例如图 1 中节点 2、3、4、5 只有前驱节点 1,且节点 1 是 QoS 可归约节点,因此它们都是 QoS 可度量节点。

定义 3　在 DAG $G=(V,E)$ 中,若节点 $j\in V$ 有且只有一个直接前驱节点 i ,且节点 j 是 QoS 可度量节点,节点 i 是 QoS 可归约节点,则节点 i 和 j 组成串归约结构。

例如图 1 中的节点 1 与 2,1 与 3,1 与 4,1 与 5 组成串归约结构。

定义 4　在 DAG $G=(V,E)$ 中,若节点 $j\in V$ 有 $m(m\geqslant 2)$ 个直接前驱节点 $i_1,i_2,\cdots,i_m$,且节点 j 是 QoS 可度量节点, m 个节点 $i_1,i_2,\cdots,i_m$ 都是 QoS 可归约节点,则节点 j 和 $i_1,i_2,\cdots,i_m$ 组成并归约结构。

需要说明的是,算法在运行过程中,节点的类型是不断变化的,这是由于已经归约过的节点要从图中删除。正是节点类型的不断变化,使得算法在运行过程中,逐步将当前的 QoS 可归约节点和 QoS 可度量节点组织为串归约结构或并归约结构。再利用 2.2 节给出的这两种结构的 QoS 属性计算公

式，就可以归约得到组合服务的 QoS 属性计算结果。

2.2 基本结构的 QoS 属性计算公式

由第 1 章中的问题描述，一个服务的 QoS 属性可用一个 4 元组表示，即 $Q^j = \{PR_j, DU_j, RE_j, AV_j\}$。借鉴迭代归约度量方法中基本结构的 QoS 度量公式，本节给出串归约结构和并归约结构的 QoS 属性计算公式。

2.2.1 串归约结构的 QoS 计算公式

由迭代度量法可知，执行费用、执行时间在串归约结构上的计算结果分别是两个节点的执行费用、执行时间之和，可靠性、可用性在串结构上的计算结果分别是两个节点的可靠性、可用性之积。将上述计算方法应用于本文提出的串归约结构。计算方法描述如下：

令节点 i 和 j 是组成串归约结构的两个节点，且 i 是 j 的直接前驱。串归约结构的 QoS 属性计算结果是：将节点 i 的 QoS 属性值按照式(1)归约到节点 j 的 QoS 属性值上。

$$Q^j = \begin{cases} PR_j = pr_j + PR_i \\ DU_j = du_j + DU_i \\ RE_j = re_j + RE_i \\ AV_j = av_j + AV_i \end{cases} \tag{1}$$

需要说明的是，式(1)中，节点 j 的执行费用计算完成后，还需将节点 i 的执行费用置为零。这是因为，若节点 i 存在多个后继节点，并与后继节点多次组成串归约结构的情况下，节点 i 的执行费用会被多次累加到它的后继节点上，影响了执行费用的度量准确性。为避免这种情况发生，当节点 i 的执行费用被累加到第一个组成串归约结构中的节点 j 后，节点 i 的执行费用就被设置为零。

2.2.2 并归约结构的 QoS 计算公式

设节点 $j \in V$ 与其 $m(m \geqslant 2)$ 个直接前驱节点 $i_1, i_2, \cdots, i_m$ 组成并归约结构。其度量方法可分为两步：首先由 m 个节点组成并结构，根据迭代归约度量法中的并结构公式，得到并结构的 QoS 属性计算结果；再将并结构的 QoS 属性计算结果与节点 j 组成串归约结构，按式(1)的计算方法，即并结构的 QoS 计算结果归约到节点 j 的 QoS 属性值上。

由迭代归约度量法知，并结构在执行费用上的计算结果是 m 个节点的执行费用求和；在执行时间上的计算结果是 m 个节点的执行时间取最大值；在可靠性上的计算结果是 m 个节点的可靠性取最小值，在可用性上的计算结果是 m 个节点的可用性取最小值。

综合上述过程，并归约结构的 QoS 计算公式为：

$$Q^j = \begin{cases} PR_j = pr_j + \sum_{i=1}^{m} PR_i \\ DU_j = du_j + \max\limits_{1 \leqslant i \leqslant m} \{DU_i\} \\ RE_j = re_j \times \min\limits_{1 \leqslant i \leqslant m} \{RE_i\} \\ AV_j = av_j \times \min\limits_{1 \leqslant i \leqslant m} \{AV_i\} \end{cases} \tag{2}$$

与串归约结构同样需要说明，式(2)中，节点 j 的执行费用计算完成后，m 个节点的执行费用都要置为零，原因与串归约结构相同。

2.3 QCMTSR 算法描述

QCMTSR 算法中采用前驱邻接表存储 DAG 中节点间的偏序关系和节点类型，同时还存储服务实例名称及 QoS 属性信息。为方便编程，令 T_j 表示节点 j 的节点类型，并假设节点类型对应如下：

$$T_j = \begin{cases} -1, & \text{初始状态} \\ 0, & \text{QoS 可归约节点} \\ 1, & \text{QoS 可度量节点} \end{cases}$$

在上述规则下，QCMTSR 算法具体描述如下：

1) 初始化 DAG：将图 G 中的偏序关系、服务实例的 QoS 属性信息存入前驱邻接表 A_G，并置所有节点的节点类型为初始状态 -1，即 $T_j = -1$；

2) 对图 G 拓扑排序生成拓扑序列 B_G；

3) 从序列 B_G 的头节点开始：$i = 1$；

4) 获取 $B_G(i)$；

5) 找到其前驱个数 N_i；

6) 若 $N_i = 0$，则修改 i 的节点类型为 QoS 可归约节点，即 $T_j = 0$，转 12)；

7) 修改 i 的节点类型为 QoS 可度量节点，即 $T_j = 1$；

8) 若 $N_i > 1$，转 10)，否则继续；

9) 根据串归约结构的 QoS 属性计算式(1)计算 i 的各 QoS 属性值；

10) 根据并归约结构的 QoS 属性计算式(2)计算 i 的各 QoS 属性值；

11) 修改 i 的节点类型为 QoS 可归约节点 $T_j = 0$；

12) i++；

13) 若 $i \leqslant n$，转 4)；

14) 输出节点 n 的 QoS 值；

15) 结束。

QCMTSR 算法中，1) 创建前驱邻接表的时间复杂度为 $O(n \times e)$，2) 拓扑排序的时间复杂度为 $O(n + e)$，3) ~13) 是扫描拓扑序列，且每个节点可能还要进行串归约和并归约的 QoS 属性计算，一次属性计算最多需要 l 次(l 是图中节点的最大入度，即最大前驱个数)，所以它们的时间复杂度最大为 $O(n \times l)$。因此，整个算法的时间复杂度为 $O(n \times e)$。

可以证明，对任意一个有向无环图表示的组合服务，该度量算法都是可行的。这是因为，任何一个有向无环图的全部顶点都可以通过拓扑排序生成一个拓扑序列。按照拓扑序列的节点顺序访问到的节点要么是 QoS 可归约节点，要么是 QoS 可度量节点。这是由拓扑序列的性质决定的。且若当前访问的节点是 QoS 可度量节点，其直接前驱节点必然是 QoS 可归约节点。而且 QoS 可度量节点经过归约后转化为 QoS 可归约节点。重复这种过程，能归约到最后一个节点。最后一个节点的 QoS 属性值就是组合服务的 QoS 度量结果。

3 实例说明及分析

为说明 QCMTSR 算法的执行过程，下面用以图 1 给出的组合服务为例来说明算法的计算过程，能够完成各节点功能的 Web 服务实例的 QoS 属性值如表 1 中数据所示。首先初始化 DAG，得到图 G 的前驱邻接表如图 2 所示(图中上方字母表示各列存储的信息)。然后对 DAG 拓扑排序，得到一个拓扑序列为 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16。按照 QCMTSR 算法步骤依次访问该序列，最终得到的 QoS 属性值如表 2 所示。

表 2 算法 QoS 属性度量结果

算法名称	时间/s	费用/元	可靠性	可用性
QCMTSR 算法	61	124	0.083 7	0.051 0
关键路径算法	61	124	0.128 1	0.123 6

表 2 还同时列出了文献[10]采用时间关键路径算法得到的各个 QoS 度量结果。可以看出，两种度量算法在执行时间、执行费用得到了相同的计算结果；在可靠性、可用性上，QCMTSR 算法得到的度量结果要小于关键路径度量方法。这是因为 QCMTSR 算法中，执行时间是取组合服务中路径的最长执行时间，因此会和时间关键路径一致；在费用上两者实质

上都是对所有节点的执行费用求和，故两个算法的执行费用度量结果也相同。而在可靠性、可用性度量上，QCMTSR 算法考虑了图中所有节点对度量结果的贡献，弥补了时间关键路径只考虑关键路径节点的不足，因此 QCMTSR 算法比关键路径算法更能准确地实现对所有 QoS 参数的度量。

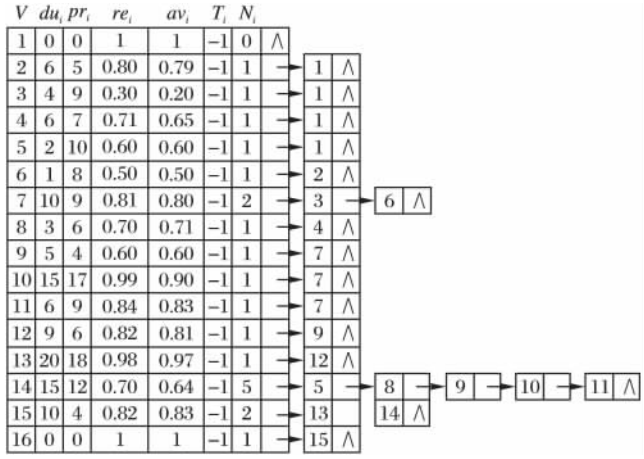

图 2　图 1 中实例的前驱存储邻接表

图 3 是 100 组随机生成的 DAG 数据[15] 可靠性参数的计算对比结果，说明在可靠性上，QCMTSR 算法得到的度量结果不大于关键路径度量方法(其中重合部分为 QCMTS 可靠性运算路径和关键路径相同所致)。

这说明，本文提出的 QCMTSR 算法，综合考虑图中所有节点对度量结果的贡献，而且在 QoS 属性计算方法中借鉴了迭代度量方法中基本结构的计算公式，有很强的数学理论基础。因此，QCMTSR 算法能够更客观、合理地度量组合服务的执行费用、执行时间、可靠性、可用性。

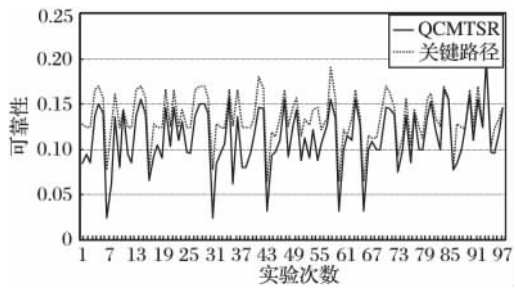

图 3　100 组数据比较图

4　结语

组合服务的QoS度量在Web服务组合研究中占据着极其重要的地位，如何对组合服务QoS给出一种客观、合理、快速有效的度量方法成为一个重要问题。本文考虑 DAG 描述的组合服务流程，提出了一个基于拓扑序列归约的组合服务 QoS 度量方法(QCMTSR)。从理论上证明了 QCMTSR 算法适用于所有 DAG 描述的组合服务，实验结果表明 QCMTSR 算法弥补了关键路径度量算法的不足，对可靠性和可用性能够得到更合理的度量结果。

参考文献:

[1] 韩燕波，王桂玲，刘晨，等. 网格、云、SaaS、SOA、WebX.0——本质是互联网计算[J]. 中国计算机学会通讯，2009，5(11)：55－58.

[2] 岳昆，王晓玲，周傲英. Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15(3)：428－442.

[3] 苑迎春，王克俭，韩宪忠，等. 基于工作流的 Web 服务组合多视图模型研究[J]. 计算机集成制造系统，2010，16(1)：30－36.

[4] 胡建强，李娟子，廖桂平. 一种基于多维服务质量的局部最优服务选择模型[J]. 计算机学报，2010，33(3)：526－535.

[5] ZHANG XUYUN，DOU WANCHUN. Preference－aware QoS evaluation for cloud Web service composition based on artificial neural networks[C]// Proceedings of the 2010 International Conference on Web Information Systems and Mining. Berlin：Springer-Verlag，2010：410－417.

[6] 杨放春，苏森，李祯. 混合 QoS 模型感知的语义 Web 服务组合策略[J]. 中国科学：E 辑(信息科学)，2008，38(10)：1697－1716.

[7] CARDOSO J，AMIT P S，JOHN A. *et al*. Modeling quality of service for workflows and Web service processes[J]. Journal of Web Semantics，2004，1(3)：281－308.

[8] 蒋哲远，韩江洪，王钊. 动态的 QoS 感知 Web 服务选择和组合优化模型[J]. 计算机学报，2009，32(5)：1014－102.

[9] 张佩云，黄波，孙亚民. 面向服务组合的服务语义匹配机制[J]. 电子科技大学学报，2008，37(6)：917－921.

[10] 梁泉，王元卓. 面向服务 QoS 模型中一种需求映射方法[J]. 计算机科学，2010，37(5)：95－98.

[11] 王勇，代桂平，侯亚荣，等. 基于遗传算法实现服务组合中信任感知的成员服务选择[J]. 北京工业大学学报，2010，36(1)：112－116.

[12] 刘书雷，刘云翔，张帆，等. 一种服务聚合中 QoS 全局最优服务动态选择算法[J]. 软件学报，2007，18(3)：646－656.

[13] ZENG L Z，BENATALLAH B，NGU A H H，*et al*. QoS-aware middleware for Web services composition[J]. IEEE Transactions on Software Engineering，2004，30(5)：311－327.

[14] 金海，陈汉华，吕志鹏，等. CGSP 作业管理其合成服务的 QoS 优化模型及求解[J]. 计算机学报，2005，28(4)：578－586.

[15] 郭涛，么炜，苑迎春. 利用遗传算法改进 DAG 绘制的方法[J]. 计算机工程与应用，2011，47(8)：172－174.

(上接第 1431 页)

[2] 左正兴，蔡坪. 有限元后处理程序的研究[J]. 计算机辅助设计及图形学报，1992，4(1)：51－56.

[3] SOMAN K P，DIWAKAR S，AJAY V. Insight into data mining theory and practice[M]. 北京：机械工业出版社，2009.

[4] RICHTE J. Programming applications for Microsoft Windows[M]. Washington，DC：Microsoft Press，2000.

[5] PROSISE J. Programming Windows with MFC[M]. Washington，DC：Microsoft Press，1999.

[6] 侯俊杰. 深入浅出 MFC[M]. 武汉：华中科技大学出版社，2001.

[7] RICHTE J. Windows 核心编程[M]. 4 版. 黄陇，李虎，译. 北京：机械工业出版社，2008.

[8] 李增刚. Nastran 快速入门与实例[M]. 北京：国防工业出版社，2007.

[9] 孙文庆，刘秉权. 基于内存映射文件的数据共享技术研究与应用[J]. 微计算机应用，2005，26(1)：192－195.

[10] 沙海蜂，高飞. 基于内存映射文件技术的海量 VCT2.0 数据库文件管理方法[J]. 测绘通报，2009(10)：47－49.

[11] 王华. 基于内存映射文件的特大机械图纸快速处理技术[J]. 浙江科技学院学报，2005，17(3)：179－183.

[12] RICHTE J. Windows 高级编程指南[M]. 3 版. 王书洪，刘光明，译. 北京：清华大学出版社，1999.

[13] 张泽清. 浅析 Windows 内存映射文件[J]. 福建师范大学福清分校学报，2006(2)：20－25.

[14] SOLOMON D A，RUSSINOVICH M E. Inside Microsoft Windows 2000[M]. Washington，DC：Microsoft Press，2001：304－340.

[15] KRUGLINSKI D J. Visual C++ 技术内幕[M]. 4 版. 潘爱民，王国印，译. 北京：机械工业出版社，1998.