第36卷第12期 东北大学学报(自然科学版) Vol. 36, No. 12
2015年12月 Journal of Northeastern University(Natural Science) Dec. 2015

doi: 10.3969/j.issn.1005-3026.2015.12.003

一种基于执行环境特征的组合服务混合执行机制

杨 雷[1], 代 钰[2], 张 斌[1]
(1. 东北大学 信息科学与工程学院, 辽宁 沈阳 110819; 2. 东北大学 软件学院, 辽宁 沈阳 110819)

摘 要: 研究了一种组合服务混合执行机制,建立了服务间的交互协议,给出了执行环境特征模型,并在该模型基础上给出了执行模式选择规则. 该机制结合了控制流、数据流全集中,控制流集中、数据流分散两种执行模式的特点,能够根据组合服务业务特性和网络环境特点动态选择执行模式,可以使得数据量较大的消息直接在服务间传递而无需通过中心节点转发. 实验结果表明:组合服务混合执行机制通过在组合服务执行过程中动态选择执行模式,能够有效保证组合服务的执行效率.

关 键 词: 组合服务;组合服务执行;数据流;控制流
中图分类号: TP 303.1 **文献标志码**: A **文章编号**: 1005-3026(2015)12-1682-05

A Hybrid Composite Service Execution Mechanism Based on the Characteristics of the Execution Environment

YANG Lei[1], *DAI Yu*[2], *ZHANG Bin*[1]
(1. School of Information Science & Engineering, Northeastern University, Shenyang 110819, China; 2. School of Software, Northeastern University, Shenyang 110819, China. Corresponding author: YANG Lei, E-mail: yanglei@mail.neu.edu.cn)

Abstract: A hybrid execution mechanism for composite service was researched. The interaction protocol between the services was established, and the characteristic model of the execution environment was presented. In addition, the rules for dynamically selecting the execution schema were proposed. The mechanism was a combination of execution schema of the centralized control-flow and data-flow as well as the execution schema of the centralized control-flow and decentralized data-flow, which could dynamically select the execution schema according to the feature of the business and the execution environment of the composite service. And then the message with large data amount could be transferred between the services without being redirected by the central node. The experimental results showed that the dynamic selection of the execution schemas could ensure the execution efficiency of the composite service.

Key words: composite service; execution of composite service; data-flow; control-flow

随着云计算技术的发展,特别是随着软件服务技术[1-2]的成熟,以服务构建应用系统[3]已经成为一种重要的软件使用和开发模式,服务组合及其应用是目前研究的一个热点.

组合服务是由服务构成的[4-5],组合服务执行效率受服务执行时间以及服务间消息通信时间的影响. 为保证组合服务执行效率,国内外研究者开展了大量的研究工作. 例如,文献[6-7]提出了QoS驱动的服务选取方法. 这些研究工作大都强调服务间的通信需要通过执行引擎进行中转(即控制流、数据流全集中模式,简记为CCCD模式),同时假设服务间消息通信量增大时对执行引擎的处理效率没有任何影响. 然而,在实际应用中,服务间可能需要传递大量数据,由于数据流消息需要通过执行引擎转发给其他服务,执行引擎的消息转发效率将影响组合服务执行性能. 另外,

收稿日期: 2015-04-17
基金项目: 国家科技支撑计划项目(2015BAH09F02, 2015BAH47F03); 中央高校基本科研业务费专项资金资助项目(N130417002, N130404011).
作者简介: 杨 雷(1974-),男, 辽宁营口人,东北大学副教授,博士; 张 斌(1964-),男, 辽宁本溪人,东北大学教授,博士生导师.

也有研究者提出通过服务间直接消息传递解决组合服务执行效率的方法. 例如,文献[8]中提出了一种控制流集中、数据流分散的组合服务执行模式(简记为 CCDD 模式). 该模式中,控制流消息需要根据服务间的控制流依赖关系依次发送给其所有的控制流后继服务. 当服务间数据流消息数据量较小且控制流依赖关系较多时,采用 CCDD 模式的组合服务执行效率可能要比采用 CCCD 模式的低.

考虑到组合服务执行模式对其执行效率的影响,本文研究一种组合服务混合执行机制,根据组合服务业务特性和网络环境特点,判断数据流是经过执行引擎转发还是直接在服务间进行传递,从而通过选择执行模式保证组合服务执行效率.

1　组合服务混合执行机制

1.1　机制描述

图 1 给出了组合服务混合执行机制的分层结构,包括实例调度执行层、执行模式选择层和集成交互层. 这里:实例调度执行层的主要功能是执行并调度组合服务运行实例;执行模式选择层主要负责根据运行实例的业务特点和网络环境状况,选择能够保证实例执行效率的执行模式;集成交互层的主要功能是接收/发送消息,完成与服务代理以及服务间的交互过程.

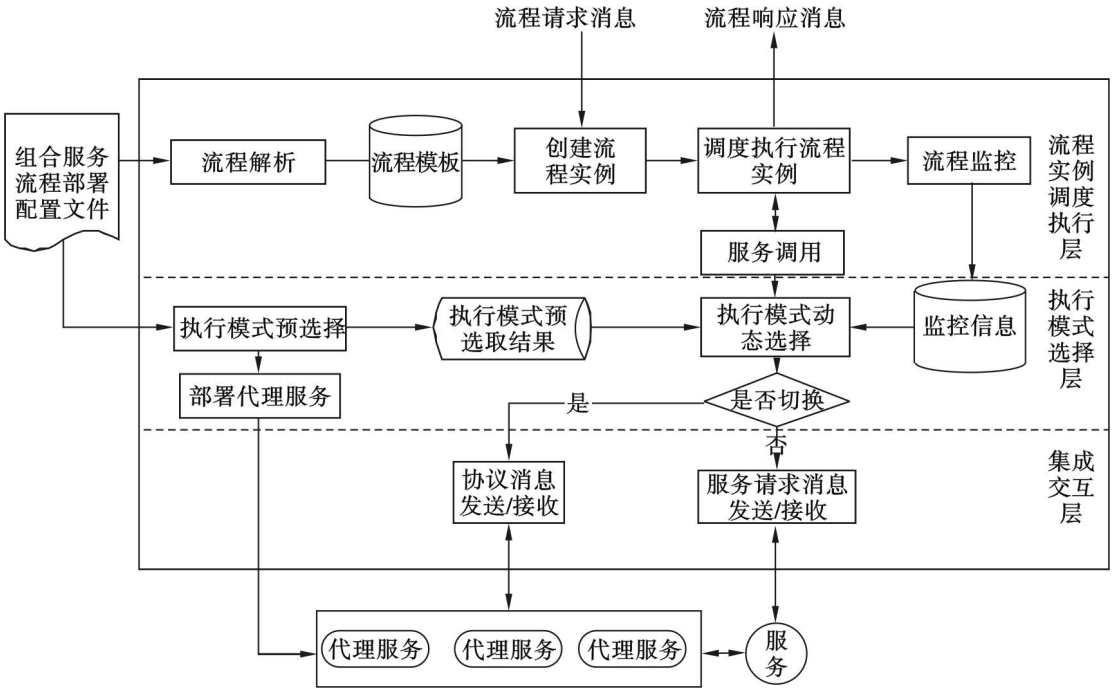

图 1　组合服务混合执行机制分层结构

Fig. 1　Architecture of hybrid execution mechanism for composite service

1.2　面向混合执行的交互协议

在组合服务混合执行机制中,执行引擎[9-10]是组合服务系统的控制中心;服务代理起到引擎和服务间的协调作用,负责接收引擎的控制消息,对服务进行调用. 交互协议用于在执行引擎和服务代理间传递控制流和数据流消息. 表 1 给出了本文所设计的交互协议.

1.3　执行环境特征模型及执行模式选择规则

本文建立了能够反映服务所处业务和网络环境特点的执行环境特征模型,并在该模型的基础上提出了执行模式选择规则.

定义 1　组合流程. 组合流程包括一组活动以及这组活动间的时序控制关系,它可以被形式化地定义为 $C = <\mathrm{ID}, A, E_c, E_d>$,其中:ID 为组合流程的唯一标识;$A$ 表示组合流程中的活动集;E_c 表示组合服务流程中的控制流;E_d 表示组合服务流程中的数据流.

定义 2　组合服务和组合服务执行实例. 当组合流程 C 中的活动 a_i 要利用候选服务 s_{ij} 完成其功能时,称二元组 $<a_i, s_{ij}>$ 为活动 a_i 的一个指派. 如果对于 C 中的每个活动 a_i,指派集合 P 中有且仅有一个元素为 a_i 的指派,则称 P 为该组合流程的一个组合服务. 对于一个组合服务,其可以被多个使用者所调用,从而可以存在多个组合服务执行实例. 对于组合服务 P_i,其第 j 个组合服务执行实例可以记为 p_{ij}.

表 1 交互协议中的消息
Table 1 Messages in the interaction protocol

协议消息名称	含义	发送者	接收者	类型
服务参数输入消息	调用服务所需的输入变量	执行引擎或服务代理	服务代理	数据流相关协议消息
服务参数完成消息	通知执行引擎服务的完成状况	服务代理	执行引擎	控制流相关协议消息
服务参数传递通知消息	通知所需交互的下一个服务代理的地址	执行引擎	服务代理	控制流相关协议消息
服务参数提取消息	通知服务代理将执行结果中与执行控制相关的参数传递给执行引擎	执行引擎	服务代理	控制流相关协议消息
服务参数回传消息	根据服务参数提取消息,发送相关服务参数给执行引擎	服务代理	执行引擎	控制流相关协议消息

定义 3 组合服务执行实例的执行环境特征模型. 对于组合服务执行实例 p_{ij},在 t 时刻,其执行环境特征模型可以定义为 $F = <\boldsymbol{B},\boldsymbol{N}>$,其中,$\boldsymbol{B}$ 为组合流程的业务特征向量,$\boldsymbol{B} = [b_1, b_2, \cdots, b_m]$,这里,如果组合服务执行实例 p_{ij} 中服务 s_{ijk} 的输出消息类型为布尔、字符以及整型等简单数据类型,则 s_{ijk} 面向的是控制处理逻辑,b_k 为 1;如果 s_{ijk} 的输出消息类型为复杂数据类型,则无法比较数据流相关协议消息与控制流相关协议消息数据量之和的大小,该情况下称 s_{ijk} 面向的业务处理逻辑不确定,b_k 为 0. $\boldsymbol{N}$ 为组合服务执行实例在 t 时刻的网络环境特征矩阵. 记组合服务执行实例 p_{ij} 中所有服务和执行引擎共同构成的集合为 O. 在时刻 t,对于 O 中的任意两个元素 o_l 和 o_k 且 o_l 和 o_k 不为组合服务执行引擎,如果 o_l 和 o_k 间存在数据流,则 n_{lk} 的取值为 t 时刻它们之间的网络带宽值;如果 o_l 和 o_k 间不存在数据流,则 n_{lk} 取值为 -1;对于 O 中的任意两个元素 o_l 和 o_k 且 o_l 为组合服务执行引擎,则 n_{lk} 的取值为它们之间的网络带宽.

本文先给出在两种执行模式下组合服务执行时间的计算方法,并在此基础上给出组合服务的执行模式选择规则. 以下,在组合服务执行实例 p_{ij} 中,服务 s_e 采用 CCCD 执行模式的执行时间记为 $W(p_{ij}, s_e)$,简记为 W;采用 CCDD 执行模式的执行时间记为 $H(p_{ij}, s_e)$,简记为 H.

1) 单一数据流依赖. 对于组合流程 CP 中的活动 a_e,如果 a_e 有且仅有一个数据依赖后继活动 a_f,则称活动 a_e 为单一数据流依赖活动.

在 CCCD 执行模式下,活动 a_e 与 a_f 间的交互时间为发送结果消息(m_r)给引擎的网络传输时间、引擎处理时间(t_g)及引擎将输入消息(m_i)发送给服务 s_f 的网络传输时间之和.

在 CCDD 执行模式下,活动 a_e 与 a_f 间的交互时间为服务参数完成协议消息(m_p)的网络传输时间、对于服务参数完成协议消息的执行引擎处理时间(t_d)、服务参数传递通知协议消息(m_y)的网络传输时间、对于服务参数传递通知协议消息的服务代理处理时间(t_f)和服务参数输入协议消息(m_n)的网络传输时间之和.

本文假设相比较于网络传输时间,引擎和服务代理的协议消息处理时间很小,可以忽略不计. 本文分别将活动平均输出数据量 $\overline{D}(P, a_e)$ 和活动平均数据依赖数据量 $\overline{R}(P, a_e, a_f)$ 作为 m_r 和 m_p 的历史经验值. 从而,在 CCCD 和 CCDD 执行模式下,服务 s_e 与 s_f 的交互时间可以分别计算为

$$W = \frac{\overline{D}(P, a_e)}{\overline{n}(s_e, E)} + \frac{\overline{R}(P, a_e, a_f)}{\overline{n}(s_f, E)}, \quad (1)$$

$$H = \frac{m_p}{\overline{n}(s_e, E)} + \frac{m_y}{\overline{n}(s_e, E)} + \frac{\overline{R}(P, a_e, a_f)}{\overline{n}(s_e, s_f)}. \quad (2)$$

根据协议消息内容的定义,m_p 和 m_y 的内容和格式是固定的,其数据量大小可以被事先确定. $\overline{n}(o_l, o_k)$ 为在过去一段时间 t_u 到 t_v 内,o_l 和 o_k 间网络环境特征的平均值. E 表示执行引擎.

在单一数据流依赖的情况下,本文将根据规则 1 和规则 2 进行执行模式的选择.

规则 1. 对于组合服务执行实例 p_{ij} 中的服务 s_e,s_e 绑定的活动 a_e 为单一数据流依赖活动且 s_e 与 s_f 的交互时间远小于 a_e 与 a_f 的执行间隔时间. 如果服务 s_e 在执行模式 CCDD 的执行时间

T_H 远小于在 CCCD 的执行时间 T_W，则采用 CCDD 执行模式；否则，采用 CCCD 执行模式，以降低切换代价.

规则 2. 对于组合服务执行实例 p_{ij} 中的服务 s_e，s_e 被绑定的活动 a_e 为单一数据流依赖活动且 s_e 与 s_f 的交互时间和 a_e 与 a_f 的执行间隔时间差别不大. 如果 T_W 较小，则采用 CCCD 执行模式；如果 T_H 较小，则采用 CCDD 执行模式；如果 T_W 和 T_H 相同，则采用 CCCD 执行模式.

2）多数据流依赖. 对于组合流程 CP 中的活动 a_e，如果 a_e 的数据流依赖后继活动的个数大于 1，则称活动 a_e 为多数据流依赖活动. 此时，将根据规则 3 确定服务 s_e 的执行模式.

规则 3. 对于组合服务执行实例 p_{ij} 中的服务 s_e，s_e 被绑定的活动 a_e 为多数据流依赖活动. 在 a_e 的所有数据依赖后继活动中，如果存在活动 a_l 且 a_l 满足如下的条件①，②和③，则服务 s_e 的执行模式为活动 a_l 的执行模式；如果所有的数据依赖后继活动都不满足条件②，且存在活动 a_l，a_l 为 a_e 的数据依赖后继活动且 a_l 的执行模式为 CCDD，则服务 s_e 的执行模式为 CCDD；否则服务 s_e 的执行模式为 CCCD：

①活动 a_l 为活动 a_e 的一个数据依赖后继活动；

②a_e 与 a_l 的交互时间同 a_e 与 a_l 的执行间隔时间相近；

③活动 a_e 与 a_l 在不同执行模式下的交互时间差是 a_e 与其他所有数据依赖后继活动的最大值.

2　实验对比分析

实验测试环境是在实验室局域网内搭建的，每个服务分别部署在不同的主机上，并且每个主机上为相应的服务部署一个服务代理，执行引擎部署在另一台主机. 限于实验室网络环境，将服务之间、执行引擎与服务之间的网络带宽降低为 100 kB/s. 实验中将测试四种情况的组合服务执行实例平均响应时间. 同时，实验中依次将整个系统并发组合服务执行实例数量由 1 增到 50. 图 2 给出了实验结果.

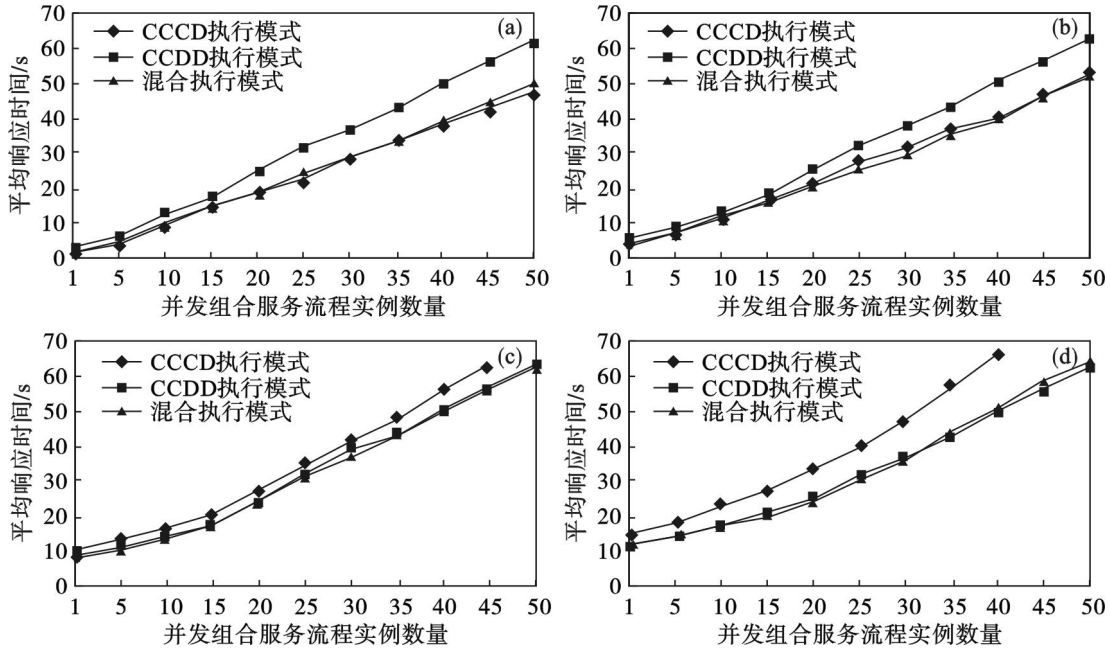

图 2　不同执行模式下的实例平均响应时间

Fig. 2　Response time of the execution instance under different schemas

(a)—每个服务的最大输出数据通信量都是 100 B；(b)—3 个服务的最大输出数据通信量为 50 kB，其他服务的最大输出数据通信量为 100 B；(c)—7 个服务的最大输出数据通信量为 50 kB，其他服务的最大输出数据通信量为 100 B；(d)—每个服务的最大输出数据通信量都是 50 kB.

图 2a 中，与 CCDD 模式相比，混合执行模式和 CCCD 执行模式具有较短的响应时间，能够保证组合服务执行效率. 这是因为在该情况下，由于服务间最大输出数据通信量都只有 100 B，为小数据量服务，最大输出数据通信量小于控制流相关协议消息数据量，CCDD 模式执行过程中增加了控制流相关协议消息在引擎与服务代理间的传递，从而使得其平均响应时间差于其他两种模式.

而混合执行模式是通过执行模式动态选择采用了 CCCD 执行模式,其响应时间基本上与 CCCD 模式相同. 图 2 b 中,混合执行模式具有较短的响应时间. 这是因为在该情况下,有 3 个服务的最大输出数据通信量为 50 kB,为较大数据通信量,混合执行模式通过为该 3 个服务选择 CCCD 模式而为其他服务选择 CCDD 模式,能够避免其中数据量较大的 3 个服务通过引擎流转消息,进而能够保证组合服务执行效率. 图 2c 中,混合执行模式具有较短的响应时间. 这是因为在该情况下,有 7 个服务的最大输出数据通信量为 50 kB,为较大数据通信量,混合执行模式通过执行模式的选择,能够保证数据通信效率,进而能够保证组合服务执行效率. 同时,还注意到,随着并发实例数的增加,引擎负载随之增加,平均响应时间增速加快. 当并发实例数达到 45 时,由于引擎处理能力限制,其开始出现内存溢出现象,部分实例执行失败. 而混合执行模式和 CCDD 执行模式通过协议消息,使得数据流分散在服务间传递,能够降低引擎负载,因此执行效率高于 CCCD 模式. 图 2 d 中,CCCD 模式具有较长的响应时间. 这是因为,随着并发实例数的增加,引擎负载增大,在并发实例数超过 40 时,出现内存溢出现象,部分实例执行失败. 而混合执行模式和 CCDD 模式由于将数据流分散在服务代理间,具有较短的响应时间.

3 结 语

本文提出了一种组合服务混合执行机制,给出了该机制的分层结构. 针对于组合服务执行模式灵活切换和选择问题,建立了交互协议,给出了执行环境特征模型,并在该模型的基础上给出了执行模式选择规则. 实验结果验证了该机制的有效性.

参考文献:

[1] Stefan T R, Malte R, Bjorn M, et al. Mixed-tenancy in the wild-applicability of mixed-tenancy for real-world enterprise SaaS-applications[C]// International Conference on Cloud Computing. Anchorage, 2014: 865 – 872.

[2] Lee W, Choi M. A multi-tenant web application framework for SaaS [C] // International Conference on Cloud Computing. Honolulu, 2012: 970 – 971.

[3] Paktinat S, Salajeghe A, Seyyedi M A, et al. Service-based application adaptation strategies: a survey [J]. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 2014, 8(8): 1350 – 1354.

[4] Feng Z W, He K Q, Peng R, et al. Taxonomy for evolution of service-based system [C]// World Congress on Services. Washington DC, 2011: 331 – 338.

[5] Mirosolav Z, Hans B. Revenue optimization of service compositions using conditional request retries [C]// International Conference on Web Services. Santa Clara, 2013: 1 – 9.

[6] Xia Y M, Yang Y B. Web service composition integrating QoS optimization and redundancy removal [C]// International Conference on Web Services. Santa Clara, 2013: 203 – 210.

[7] Saleem M S, Ding C, Liu X, et al. Personalized decision making for QoS-based service selection [C] // International Conference on Web Service. Anchorage, 2014: 17 – 24.

[8] 翟岩龙,宿红毅, 肖玮,等. 支持数据流分发的组合服务协调框架[J]. 北京理工大学学报, 2009, 29 (12): 1091 – 1095.
(Zhai Yan-long, Su Hong-yi, Xiao Wei, et al. A coordination framework of web service composition to support data-flow distribution [J]. *Transactions of Beijing Institute of Technology*, 2009, 29(12): 1091 – 1095.)

[9] James P. How BPEL and SOA are changing web services development[J]. *IEEE Internet Computing*, 2005, 9(3): 60 – 67.

[10] Girish C, Sunil C, Vijay M, et al. Orchestrating composite web services under data flow constraints[C]// International Conference on Web Service. Orlando, 2005: 211 – 218.