ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.8, August 2007, pp.1943−1957
DOI: 10.1360/jos181943

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

虚拟计算环境中服务行为与质量的一致性*

胡 昊[1,2+], 殷 琴[1,2], 吕 建[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机软件研究所,江苏 南京 210093)

Service Behavior and Quality Consistency in Virtualized Computing Environment

HU Hao[1,2+], YIN Qin[1,2], LÜ Jian[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Institute of Computer Software, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-83686690, Fax: +86-25-83593283, E-mail: myou@nju.edu.cn, http://moon.nju.edu.cn

Hu H, Yin Q, Lü J. Service behavior and quality consistency in virtualized computing environment. *Journal of Software*, 2007,18(8):1943−1957. http://www.jos.org.cn/1000-9825/18/1943.htm

Abstract: A unified framework based on Petri-net for the modeling of service behavior and QoS is proposed. Based on this framework a set of service consistency rules are derived to improve the accuracy of service discovery and to ensure the system consistency in service substitution. A middleware named SOBECA (service-oriented, behavior and capability support architecture) compatible with current Web services standards is developed to make a proof of concept of the framework. In SOBECA the service description is extended with behavior and quality attributes which can be automatically checked against current requirements.

Key words: service-oriented component model; service behavior model; behavior inheritance; QoS; Petri net

摘 要: 提出了一种用 Petri net 对服务行为和服务质量进行统一建模的框架.基于该框架所提出的统一的服务行为和服务质量的一致性规则可以在服务的查找和替换中综合判断服务行为和服务质量的一致性.自行开发的SOBECA(service-oriented, behavior and capability support architecture)中间件系统在现有的 Web 服务标准上增加了服务行为和服务质量的描述,可以更好地自动化支持服务行为和服务质量的一致性检查,同时也验证了上述框架的合理性.

关键词: 服务构件模型;服务行为模型;行为继承;服务质量;Petri 网

中图法分类号: TP393 **文献标识码**: A

网格计算作为一种新型的面向Internet的分布式计算模式,越来越受到学术界和产业界的重视[1].网格计算

* Supported by the National Natural Science Foundation of China under Grant Nos.60403014, 60603034 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z159, 2006AA01Z177, 2007AA01Z140, 2007AA01Z178 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2002CB312002 (国家重点基础研究发展计划(973)); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2006712 (江苏省自然科学基金)

Received 2007-03-07; Accepted 2007-05-31

所处的Internet环境具有无统一控制的“真”分布性,节点的高度自治性,节点链接的开放性和动态性,人、设备和软件的多重异构性,实体行为的不可预测性,运行环境的潜在不安全性,使用方式的个性化和灵活性,网络连接环境的多样性等[2].如何在开放、动态、难控的Internet环境下实现网格系统对各类资源的“虚拟化”共享和集成,已成为计算机软件技术面临的重要挑战之一.

Web服务技术[3]的出现改变了应用对Internet上网格资源的访问方式,Web服务为网格资源定义了标准的接口和协议,通过封装,用户对资源的特定访问就转化成了对Web服务接口的通用访问.最新的网格计算环境下的软件架构的研究正在向标准的SOA(services-oriented architecture)靠近.Ian Forster在文献[4]中表达了应该将SOA作为网格计算环境的标准软件架构的概念,并在此基础上将网格计算的研究描绘为面向服务的科学(services-oriented science).尤其是开放网格服务体系结构(open grid service architecture,简称OGSA)[5]和Web服务资源框架(Web service resource framework,简称WSRF)[6]的出现,更进一步地推进了网格技术和Web服务技术的融合.在国内也出现了符合OGSA/OGSI规范的服务网格系统CROWN[7]以及与网格计算和面向服务计算兼容的iVCE环境[8].上述研究成果都充分说明,利用Web Services技术可以有效地实现网格计算的“虚拟化” 问题.

利用Web服务技术解决资源“虚拟化”的首要问题是如何使资源的访问对其使用者透明,要想达到上述目标,Web服务必须能够提供动态、灵活的服务发现和替换,这就要求Web服务能够同时描述和理解使用者的功能性需求以及非功能性需求[9,10].服务行为和QoS可以认为是Web服务的功能性需求和非功能性需求的典型代表.在服务的发现和替换中,经常会发生Web服务在执行不同的服务行为时可能导致不同的QoS,而在某些QoS约束的前提下,某些服务行为将无法满足的情况.因此,在保证服务行为一致的前提下,还需保证QoS的约束一致.然而,传统的Web服务将服务行为和QoS属性信息分别描述,缺乏统一的模型,造成了对服务行为和QoS之间的关系进行统一的服务一致性综合判断的难度.

为了有效地在 Web 服务动态发现和替换时综合判断服务行为和服务质量的一致性,我们在以下 3 个方面进行了工作:

第一,使用 Petri net 形式化方法对服务行为和服务质量关系进行统一描述,并封装在服务构件模型中,这为 Web 服务动态发现和替换时综合判断服务行为和服务质量的一致性提供了模型基础;

第二,利用面向对象范型中的行为继承理论[11−13]给出了服务发现的调用一致性规则和服务替换的观察一致性规则,并在参考服务质量因素的情况下将准则进行了适合服务质量一致的扩展;

第三,通过扩展现有标准的 SOA 架构实现了一个中间件平台用以支持在服务发现和服务替换时动态判断服务行为和服务质量的一致性,以选择最优的服务行为和服务质量组合的服务.

本文第 1 节描述网格计算环境下的新型 Web 服务开发模式,强调新开发模式必须在服务发现和替换时综合满足服务行为和服务质量的一致性.然后给出一个能够支持服务行为和服务质量关系统一描述的服务构件模型,其中包含用 Petri net 进行统一表述的服务行为和服务质量规范.第 2 节给出服务发现和服务替换时服务行为和服务质量的一致性的规则,通过该规则可以选择最优的服务行为和服务质量组合的服务.第 3 节说明支持动态判断服务行为和服务质量一致性的 SOBECA(service-oriented, behavior and capability support architecture)中间件系统的设计与实现.第 4 节比较相关工作.第 5 节进行总结.

1 支持网格系统的服务构件模型和服务规范

1.1 网格计算环境下的新型Web服务开发模式

在传统的SOA架构中包含 3 个不同的角色:服务提供者(services provider)、服务请求者(services requester)和服务注册者(services registries).服务提供者发布服务描述(services description)到服务注册者中,服务请求者根据特定的服务描述从服务注册者中查询服务提供者,服务注册者返回给服务请求者一个符合服务描述的服务提供者的引用供其选择和绑定[3].在这个过程中,传统的服务描述主要关注服务的接口和静态属性信息,这样虽然可以保证服务发现时的接口一致性,但是由于缺乏对服务行为和服务质量的有效描述,可能会导致服务运

行时的功能性属性不匹配或者非功能性属性不能保证.

在新型的基于 Web 服务的网格系统中,由于网格系统所处的环境复杂、多变,因此在对服务进行描述时必须考虑到各种功能性以及非功能性属性的要求,将服务行为和服务质量统一描述到 Web 服务中,并在服务的注册/发现、组合以及替换等各个阶段进行服务一致性的综合判断,以更好地实现 Web 服务交互之间的强壮性、有效性和准确性,从而保证在网格计算环境下软件系统所应具有的动态协同、无缝适应的特点.图 1 展示了一个改进的能够适应网格计算环境的 Web 服务开发的场景.它与传统 Web 服务开发的主要区别在于,无论是 Web 服务的服务开发者还是应用开发者都需要将服务行为和服务质量统一描述到服务注册者中.

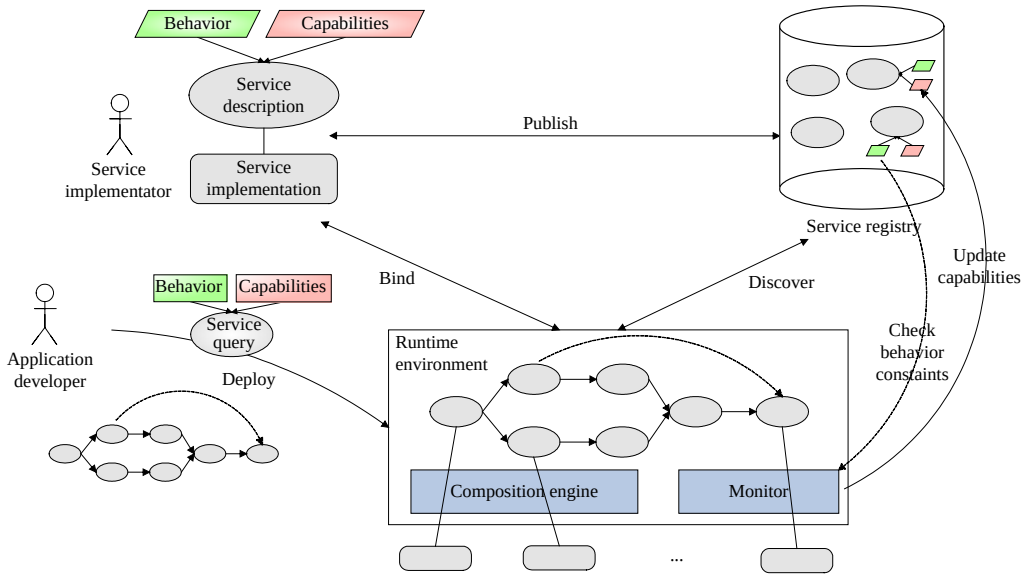

Fig.1 Extended application development process based on SOA

图 1 基于 SOA 扩展的应用开发过程

1.2 服务构件模型

为了将服务行为和服务质量统一进行描述,我们采用一种Petri net的变体WF-net[14]来描述Web服务的服务行为,而将服务质量组织为一个QoS向量,并与WF-net的跃迁建立映射,从而建立了服务行为与服务质量的统一表示,上述信息被统一地封装到服务构件模型的服务规范中,以扩充现有的服务规范,下一节将专门介绍服务规范的形式化表示.采纳服务构件模型作为扩展Web服务模型的原因是为了使改进的Web服务实现与现有的Web服务实现兼容.服务构件模型的概念是Cervantes等人在文献[15]中提出来的,目的在于将面向服务的概念与面向构件的概念结合起来,以获得两者的优点.我们采纳该模型的主要目的在于,利用Web服务使用显示的方式将服务规范和服务实现分离,使其在运行时具有动态发现多个服务实现的能力;又可利用构件模型能够打包并区分构件实现和构件实例之间的关系,使其具有在多个构件实例之间互相替换的能力,从而保证网格计算环境下软件系统所应具有的动态协同、无缝适应的特点.

我们定义的服务构件模型包含 4 个元素,分别是服务规范(service specification)、服务构件实现(services component implementation)、服务构件(service component)和服务消费者(services consumer).其中:服务规范定义了与某个服务构件实现有关的信息;服务构件实现用于实现服务规范定义的功能,并在系统运行时实例化多个服务构件;服务消费者根据服务规范在系统运行时绑定到相应的服务构件上.

限于篇幅,这里对服务构件模型不作详细介绍,具体内容请参见文献[16]的第 4 节.下面我们给出一个实例,作为说明本文后续内容的基础.图 2 是一个服务构件实现 LogServiceImpl 的例子,两个服务规范包含在 LogSeriveImpl 中,一个是提供的服务规范 ics.LogService,即 LogServiceImpl 服务构件实现所实现其功能的规范;

另一个是需要的服务规范 ics.ManagedService,即为了有效实现 LogServiceImpl 功能所还需要的服务构件所应具有的规范.无论是提供的服务规范还是需要的服务规范,其组织形式都是一样的,其中都包含了对服务行为和服务质量的统一描述.图 2 还有另一层隐含的意思,即任何一个服务规范实际上都对应着多个服务构件实现,而多个服务构件也可以在满足服务规范的前提下相互替换.

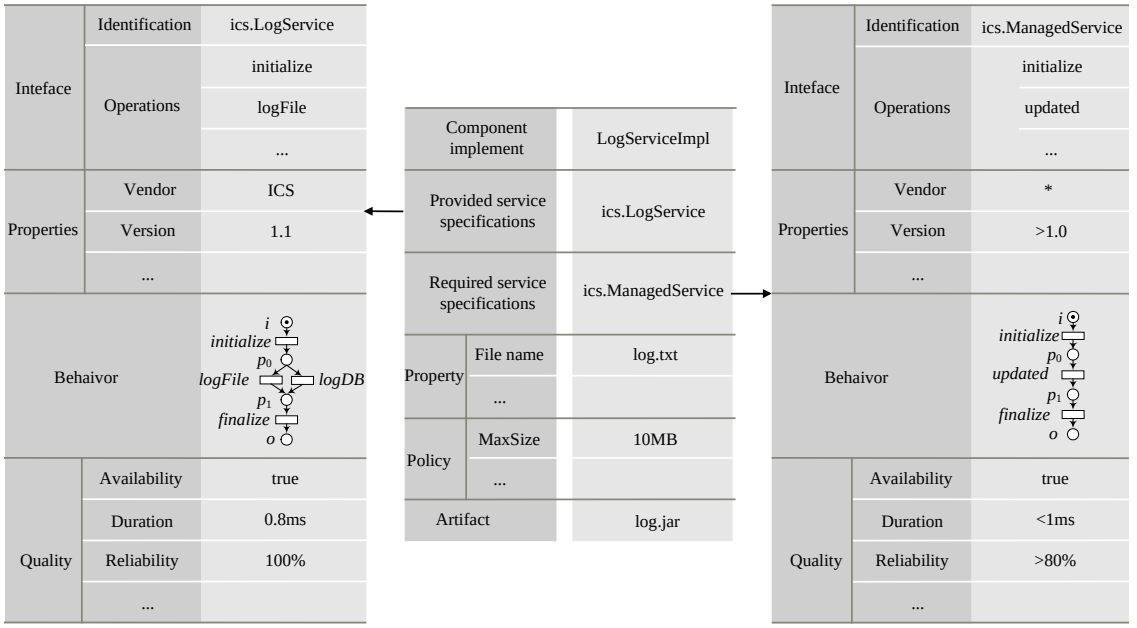

Fig.2 LogServiceImpl service component implementation

图 2 LogServiceImpl 服务构件实现

1.3 统一服务行为和服务质量描述的服务规范

服务规范是服务构件的访问通道,服务消费者通过服务规范了解实际的服务构件的信息,从而判定服务构件是否满足其服务接口、服务行为以及服务质量等要求.一个服务规范包含 3 部分内容,分别是服务接口、行为模型以及 QoS 向量.这些信息以 XML 文件的形式注册到服务注册者中,用于在服务的发现和替换时进行服务一致性匹配.在本节中,为了探讨简单,我们只给出行为模型和 QoS 向量的形式定义.

1.3.1 行为模型

行为模型定义了一个服务的合法的操作序列,该序列也是这个服务合法行为的一种约束,每个服务的调用者都必须在调用服务时遵守该服务的行为约束.我们采用一种Petri net的变体WF-net[14]作为行为模型的形式化方法.

定义 1(行为模型). 服务S的行为模型$BM_s=(P,T,F,i,o,TM)$是一个基于WF-net定义的Petri net的变体,BM_s是一个WF-net当且仅当如下条件成立:

- P为BM_s库所集合.
- T为BM_s变迁集合,且$P\cap T=\varnothing$.
- F为BM_s的弧线集合,且$F\in(P\times T)\cup(T\times P)$.
- i是BM_s的源库所,满足不存在任何$t\in T$,使得$(t,i)\in F$.
- o是BM_s的汇结库所,满足不存在任何$t\in T$,使得$(o,t)\in F$.
- $TM:T\to M_s\cup\{\varepsilon\}$是一个标号映射函数.即将$BM_s$中每一个变迁映射为服务$S$所具有的一个接口方法$m$,如果没有对应的接口方法可供映射,则设置为$\varepsilon$;如果接口方法$m$被连接到了$T$中的多个变迁,则给每个变

迁所对应的同一个接口方法m添加一个唯一的下标,记为m_i.

- 如果在BM_s中加入一个新的变迁t^*,那么所得到的扩展后的Petri网 $BM_s^*=(P,T\cup\{t^*\},F\cup\{(o,t^*),(t^*,i)\},i,o,TM\cup\{t^*\to\tau\})$是强联通的(strongly connected).

定义行为模型的目的在于在服务的发现和替换时进行服务行为的一致性匹配,行为模型在服务的发现和替换过程中既可以充当服务的行为规范,也可以充当服务的行为实例.区分这两个概念是为了说明在服务发现和替换过程中我们采用不同的行为一致性匹配规则的假设.在第 2 节中对此会有详细说明.

为了有效地阐述后续的行为一致匹配规则,我们给出如下定义:

定义 2(服务状态). 服务行为模型$BM_s=(P,T,F,i,o,TM)$的服务状态r是一组标记(token)在P上的分布(marking),对于每个服务行为模型,$\{1i\}$表示服务初始创建时的服务状态,而$\{1o\}$表示服务结束时的服务状态.

定义 3(点火规则). 服务行为模型BM_s中的一个跃迁在服务状态r下能够被点火,代表该跃迁在服务状态r下就绪,跃迁t就绪的条件当且仅当: (1) t的每个输入库所都包含一个标记(token);(2) 与t映射的接口方法m正被调用.当接口方法m被执行后,也就是变迁t被实施后,t从每个输入库所中消耗一个标记,并为每个输出库所产生一个标记.此时,服务状态也变化到一个新的服务状态r'.这可以表示为$r|t\rangle r'$,状态r'称作从状态r直接可达.

定义 4(可达图). 可达图G表示为一个元组$(\overline{r_0},E)$,其中,$\overline{r_0}$为图的点的集合,E为图的边的集合,有$E\subseteq\overline{r_0}\times\overline{r_0}\times TM(T)$,且$(r_i,r_j,TM(t))\in E$当且仅当条件$r_i|t\rangle r_j$成立.

定义 5(接口方法序列). 服务行为模型BM_s的一个合法接口方法序列$\langle m_1,\ldots,m_n\rangle$可以通过深度优先算法从$BM_s$的可达图中导出.一个服务$S$所具有的所有合法接口方法序列表示为 LMS_{BM_s}.

1.3.2 QoS 向量

服务质量代表了一个服务的非功能性属性,当用户选择和替换服务时,不仅要保证服务的功能性属性,还要保证服务的非功能性属性.一般来说,对于一个基本的服务,通常可以考察下面 3 个通用的质量属性:

- 执行时间(execution duration). 执行时间表示对服务的接口方法发送请求和接受结果的时间.该时间由两部分构成:一个是接口方法的处理时间,另一个是消息的传输时间.接口方法的处理时间是 Web 服务发布的响应值,而传输时间是一个历史经验的估计值.
- 信任度(reputation). 信任度是依赖使用服务接口的用户经验.其值是每个最终用户所给定的信任度值的平均.
- 成功执行率(successful execution rate). 成功执行率是对服务接口方法发送的一个请求在最大期望时间段中正确响应的概率.

在文献[9]中,Zeng 等人提出了一个服务质量向量的概念,参考上述想法,我们给出类似的一个 QoS 向量的形式定义,并通过将 QoS 向量与行为模型的跃迁建立映射,从而得到服务质量与服务行为之间的映射关系.

定义 6(QoS向量). 服务S的QoS向量$Q(op)_s=(q_i(op)_s,1\le i\le 3)$是一个基本QoS指标$q_i(op)_s$在服务接口$op$上的值的向量,其中,$op$为行为模型$BM_s$的接口方法,满足$op\in M_s$.

从上述定义可以看出,对于一个行为模型的任意一种接口方法,都有一组基本 QoS 指标与其对应.建立 QoS 向量的目的一方面可以对复杂的 QoS 指标进行多策略的综合分析,另一方面也有利于未来扩充新的 QoS 指标.

在本文的后续章节中,为了有效地说明服务发现和替换中的 QoS 一致性问题,我们给出了一个策略相容的概念.之所以称为策略相容,指的是服务的 QoS 是否满足,需要根据用户以及其他系统环境等多种情况进行多策略分析.这里,我们给出了一个简单的对策略相容的定义.以后可以根据新的用户要求扩充策略的含义,从而给出新的策略相容的定义.

定义 7(策略相容). 服务S'的QoS向量 $Q(op)_{s'}=(q_i(op)_{s'},1\le i\le 3)$ 与服务S的QoS向量$Q(op)_s=(q_i(op)_s,1\le i\le 3)$策略相容包含两种:

- 局部相容:对于 $op_j\in(M_s\cap M_{s'}),1\le j\le n$, 满足 $bound(q_i(op)_{s'})\subseteq bound(q_i(op)_s),1\le i\le 3$. 其中,$bound(q_i(op_j)_s)$表示一个基本QoS指标$q_i(op_j)_s$在服务接口$op_j$上值的范围.

- 全局相容:对于$op_j \in (M_s \cap M_{s'}), 1 \le j \le n$,满足$bound\left(\sum_1^n q_i(op_j)_{s'}\right) \subseteq bound\left(\sum_1^n q_i(op_j)_s\right), 1 \le i \le 3$. 其中,$bound\left(\sum_1^n q_i(op_j)_s\right)$表示一个基本QoS指标$q_i(op_j)_s$在各个服务接口$op_j$上的值之和的范围.

从上述定义中可以看出,在局部相容策略中,只需发现或者替换的服务所指定的某个操作满足原服务质量要求即可,而在全局策略中,则需要综合考虑各个操作的服务质量之和.

2 服务发现和替换中统一服务行为和服务质量的一致性规则

网格计算环境下的资源"虚拟化"要求在服务的动态发现和替换过程中,服务的服务行为与服务质量相一致,避免资源访问时不满足用户的功能性和非功能性需求,从而保证网格系统具有动态协同、无缝适应的特点.在本组已有的工作中[17,18],我们在借鉴了由van der Aalst等人[11]、Ebert等人[12]和Harrel等人[13]发现的行为继承理论的基础上,提出了服务发现的调用一致性规则和服务替换的观察一致性规则.这两个规则可以分别保证在服务的发现阶段和服务的替换阶段的服务行为一致性.然而,上述工作并没有考虑服务质量的选择和服务行为的选择之间会互相影响,缺乏统一的对服务行为和服务质量一致性的综合判断,无法有效地支持在动态、复杂、多变的网格环境下的服务选择和替换.因此在本节中,我们在服务一致性规则的基础上,给出了一个QoS一致性规则,通过该规则可以在服务的发现和替换过程中,出现多个行为一致的服务情况下,选择满足最优服务质量的服务.

2.1 无QoS影响的服务行为一致性规则

服务行为一致性规则的定义借鉴了由Ebert和Engles为定义类及其子类行为描述的相互关系而提出的两种对象行为一致性关系:调用一致性和观察一致性[12]以及由van der Aalst等人采用Petri net所描述的两种行为一致性关系:协议一致性和投影一致性[11].在文献[19]中,van der Aalst特别指出协议一致性与调用一致性概念相同,而观察一致性与投影一致性概念相同.因此在本文中,我们采用调用一致性和观察一致性的概念来定义服务行为的一致性规则.

Ebert和Engles对调用一致性和观察一致性给出了如下解释:调用一致性给出了子类提供行为的下限(lower bound),即在一个超类中可被调用的任何行为序列都能够在子类中被调用;观察一致性给出了子类提供行为的上限(upper bound),即任何从子类可观察到的行为经过投影映射后都应当与超类的可观察行为对应[12,19];van der Aalst也给出了另一种角度不同但意义相同的解释,当子类中新增的行为被阻塞(block)时,其保留的行为与超类行为一致,则称为协议一致性;而当子类中新增的行为被隐藏(hide)时,其保留的行为与超类行为一致,则称为投影一致性[11].

在将行为一致性规则应用到Web服务的查找和替换时,我们有如下场景假设:服务发现应用场景是指应用开发者设计所需服务的规范,然后由服务注册者返回符合服务的引用供其绑定来组成Web服务应用.由此可以假设,当发现服务时,由于所发现的服务来源不确定,所以有可能出现服务的某些行为操作无法正常执行的情况;而服务替换应用场景是指当整个Web服务应用运行了一段时间后,某个绑定的Web服务不再适合用户或环境的需求而需要用其他Web服务加以替换.在此场景下,可以假设所替换的服务应该通过正确性验证来保证行为操作的正常执行,并且替换前后的服务应当保证Web应用不出现语义偏差.两种场景下可能导致服务发现和替换出现异常的例子可以参考本组工作[17,18].

因此,在这样的假设下,当服务发现时,必须要求所发现的服务行为能够满足服务请求者提出的服务规范的行为下限(lower bound),即当某些行为操作无法正常执行时,将其阻塞(block)掉,所得到的服务行为仍与规范行为保持一致,从而不会出现服务行为调用偏差.即满足行为调用一致性;而当服务替换时,为了保证替换前后的服务不会导致 Web 应用出现语义偏差,在服务行为操作都能正常执行的前提下,必须保证替换的服务行为满足被替换的服务行为上限(upper bound),即将替换后服务新增的行为效果隐藏(hide)后,所能观察到的行为效果也能在替换前被观察到,即满足行为观察一致性.将上述两个行为一致性规则分别应用到服务发现和替换中,可以

得到服务发现行为调用一致性和服务替换行为观察一致性规则,其具体定义如下:

规则 1(服务发现行为调用一致性). 考虑在服务发现时,没有QoS影响下的服务行为匹配的情况,则服务实现*SS*能够满足服务请求者提出的服务规范*SR*,当且仅当行为模型BM_{ss}与行为模型BM_{SR}调用一致.

一个行为模型 $BM_{s'}$ 与另一个行为模型BM_s调用一致的条件是 $LMS_{BM_s} \subseteq LMS_{BM_{s'}}$ 成立.

例 1:图 3 中, BM_{SS_1} 与BM_{SR}调用一致,*init*,*DB*,*File*,*final*分别代表*initialize*,*logDB*,*logFile*,*finalize*,可以得到{⟨*init*⟩,⟨*init*,*File*⟩,⟨*init*,*File*,*final*⟩}⊆{⟨*init*⟩,⟨*init*,*File*⟩,⟨*init*,*File*,*final*⟩,⟨*init*,*DB*⟩,⟨*init*,*DB*,*final*⟩},即 $LMS_{SR} \subseteq LMS_{SS_1}$,而 BM_{SS_2} 与BM_{SR}不满足调用一致关系.该例说明当服务发现时,用户提出一个服务的行为规范*SR*,服务SS_1的实现行为符合规范*SR*,因为SS_1的行为中提供的*logFile*可以与规范一致地被访问,而SS_2的实现行为不符合规范*SR*,因为当用户按照规范访问*logFile*时发现,由于有*trap*操作效果未知而无法正常执行*finalize*.

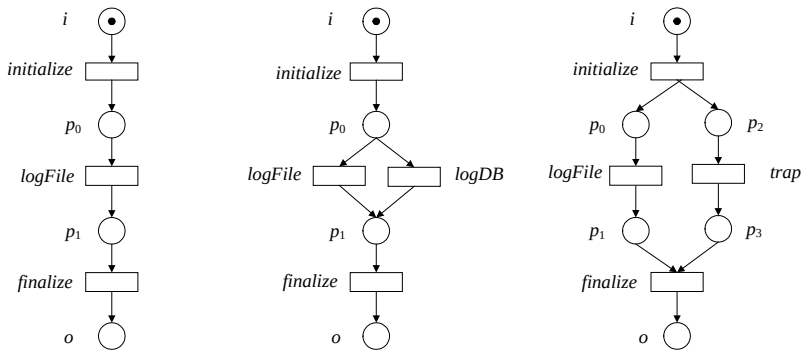

Fig.3 An example of behavior consistent service discovery and substitution

图 3 行为一致服务发现和替换实例

规则 2(服务替换行为观察一致性). 考虑在服务替换时没有 QoS 影响下的服务行为匹配的情况,则新的服务实现SS_{new}能够与老的服务实现SS_{old}在替换时满足行为一致,当且仅当行为模型 $BM_{SS_{old}}$ 与行为模型 $BM_{SS_{new}}$ 观察一致.

一个行为模型 $BM_{s'}$ 与另一个行为模型BM_s观察一致的条件是 $(LMS_{BM_{s'}} \uparrow M_s) \subseteq LMS_{BM_s}$ 成立.

$LMS_{BM_{s'}} \uparrow M_s$ 的含义是将每个合法方法序列 $lms_{BM_{s'}} \in LMS_{BM_{s'}}$ 在顺序一致的情况下只选取包含M_s的成员后所组成的新的方法序列的集合.

例 2:图 3 中, BM_{SS_2} 与BM_{SS}观察一致,因为 $LMS_{BM_{SS}} = \{\langle init\rangle, \langle init, File\rangle, \langle init, File, final\rangle\}$,而$M_{SS}$={*init*,*File*, *final*}, $LMS_{BM_{SS_2}} = \{\langle init\rangle, \langle init, File\rangle, \langle init, File, trap\rangle, \langle init, File, trap, final\rangle, \langle init, trap\rangle, \langle init, trap, File\rangle, \langle init, trap, File, final\rangle\}$,这样, $LMS_{BM_{SS_2}} \uparrow M_{ss} = \{\langle init\rangle, \langle init, File\rangle, \langle init, File, final\rangle\} = LMS_{BM_{SS}}$,而 BM_{SS_1} 与 BM_{SS} 不满足观察一致关系.

该例说明,当服务替换时,用户原有的服务实现*SS*可以被服务实现SS_2替换,而服务实现SS_1不能替换.因为SS_1的行为中提供的*logDB*可能导致服务替换后,服务无法达到*logFile*的状态而只执行*logDB*;而SS_2中的*logFile*如果被正常执行,则一定能够达到*logFile*状态.

2.2 有QoS影响的服务行为一致性规则

在动态、复杂、多变的网格环境下的服务选择和替换,不仅需要满足服务的行为一致,还需要考虑服务质量是否一致.通常情况下,当服务选择和替换时,都会出现多个符合行为一致性的可选服务存在,此时,可以将服务质量作为选择以及替换最优服务的重要指标.在本节中,我们在给出服务一致性规则的基础上给出了一个非形式化的 QoS 一致性规则,该规则中的策略相容可以作为将来判断更为复杂的统一服务行为和服务质量一致规则的基础.同时我们还发现,服务行为的选择可能会导致不同的服务质量,而在某种服务质量的约束下也可以

判定某些服务行为不合适,因此,今后还可以建立更复杂的有关服务行为模式和服务质量向量之间的关系,以便在确定某种服务质量的前提下更快地选择适合的服务行为模式.

规则 3(服务发现 QoS 一致性). 考虑在服务发现,有 QoS 影响下的服务行为匹配的情况时,有多个服务实现 ss_i 能够满足服务规范 SR,组成候选集合 $INV_{SR}=\{ss_i \mid BM_{ss_i}\ inv\ BM_{SR}, inv$为调用一致关系$\}$,则候选的服务实现 $ss_i \in INV_{SR}$ 与服务规范 SR 符合QoS一致的条件是 $Q(op)_{ss_i}$ 与 $Q(op)_{SR}$ 策略相容.这里,策略相容是根据用户的服务质量策略[20]来决定的.

例 3:图 4 显示,在服务发现以后,服务实现 SS_1 和 SS_2 符合服务规范 SR 的候选服务实现,因为两者的服务行为 BM_{SS_1}, BM_{SS_2} 与服务规范的行为模型 BM_{SR} 调用一致,当继续关注两者行为上的操作 $LogFile$ 的质量服务时,发现如图 4 所示的质量服务特征,即在服务规范 SR,服务实现 SS_1 和 SS_2 的QoS向量 $Q(File)_{SR}$, $Q(File)_{SS_1}$, $Q(File)_{SS_2}$ 三者之间满足 $ser_{SS_1} \le ser_{SR} \le ser_{SS_2}$, $ed_{SS_2} \le ed_{SS_1} \le ed_{SR}$, $r_{SR} \le r_{SS_1} \le r_{SS_2}$,其中,$ser$表示成功执行率,$ed$表示执行时间,$r$表示信任度.此时发现,如果按照成功执行率高于规范来选择,则应该选择服务实现 SS_2;而如果按照执行时间低于规范来选择,则两个服务实现都可以满足;而如果按照信任度大于规范来选择,则两个服务实现也都可以满足.在综合考虑时,则只有服务实现 SS_2 符合要求.从例 3 中可以看出,用户可以根据自身需要来定义选择服务时的质量策略.在该例中,服务质量指标关系的确定是根据一些常规假设,如, $r_{SS_1} \le r_{SS_2}$ 的设定是因为从服务行为的含义来看,SS_1 的 $logFile$ 操作可以被 $logDB$ 取代,而 SS_2 在执行完 $logFile$ 后还会通过 $getErrMsg$ 来确保执行的正确性,因此认为 SS_1 的 $logFile$ 的信任度低于 SS_2.这也隐含表示服务行为模式的不同实际上会导致不同的服务质量.

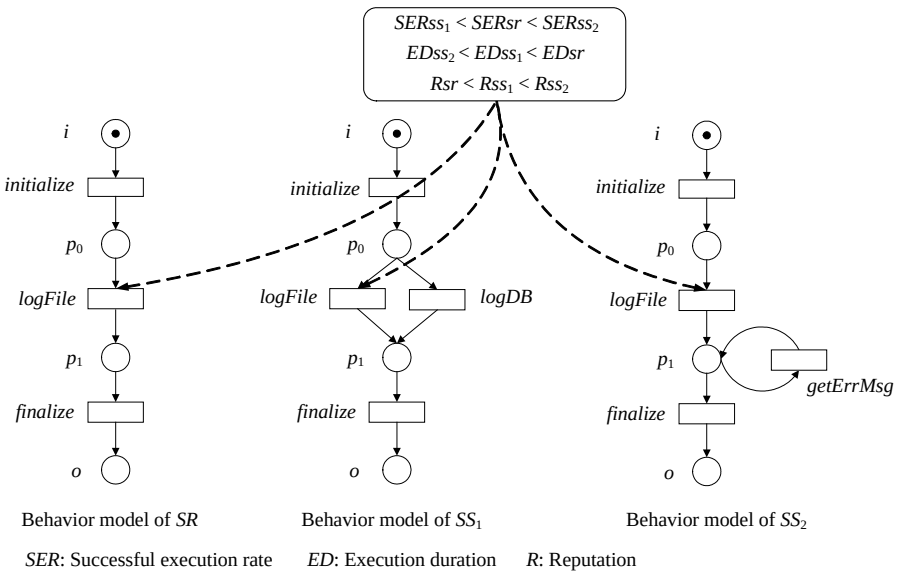

Fig.4 An example of QoS consistent service discovery

图 4 QoS 一致服务发现实例

规则 4(服务替换 QoS 一致性). 考虑在服务替换,有 QoS 影响下的服务行为匹配的情况时,有多个服务实现 ss_i 能够替换现有的服务实现 SS,组成候选集合 $OBV_{SS}=\{ss_i \mid BM_{ss_i}\ obv\ BM_{SS}, obv$为观察一致关系$\}$,则候选的服务实现 $ss_i \in OBV_{SS}$ 与现有的服务实现 SS 符合QoS一致的条件是 $Q(op)_{ss_i}$ 与 $Q(op)_{SS}$ 策略相容.这里,策略相容是根据用户的服务质量策略来决定的.

例 4:图 5 显示,在服务替换以后,服务实现 SS_1 和 SS_2 可以替换服务实现 SS 的候选服务实现,因为两者的服务行为 BM_{SS_1}, BM_{SS_2} 与服务实现 SS 的行为模型 BM_{SS} 观察一致,当继续关注两者行为上的操作 $LogFile$ 的质

量服务时,发现如图 5 所示的质量服务特征,即在原有服务实现SS,新的服务实现SS_1和SS_2的QoS向量$Q(File)_{SS}$, $Q(File)_{SS_1}$, $Q(File)_{SS_2}$ 三者之间满足 $ser_{SS_2} \le ser_{SS} \le ser_{SS_1}$, $ed_{SS_1} \le ed_{SS_2} \le ed_{SS}$, $r_{SS} \le r_{SS_2} \le r_{SS_1}$.此时,如果按照成功执行率高于原有服务实现来选择,则应选择服务实现SS_1;而如果按照执行时间低于原有服务实现来选择,则两个服务实现都可以满足;而如果按照信任度大于原有服务实现来选择,则两个服务实现也都可以满足.在综合考虑时,则只有服务实现SS_1符合要求.在该例中,服务质量指标关系的确定也是根据了一些常规假设,如,$r_{ss_2} \le r_{ss_1}$的设定是因为从服务行为的含义来看,SS_1是在*logFile*操作之后才执行*getErrMsg*以轮询验证其正确性;而SS_2是在*trap*操作的事件监控下保证其正常完成.因此可以认为,SS_1的*logFile*的信任度是高于SS_2的.这也再次隐含表示出服务行为模式对服务质量的影响.

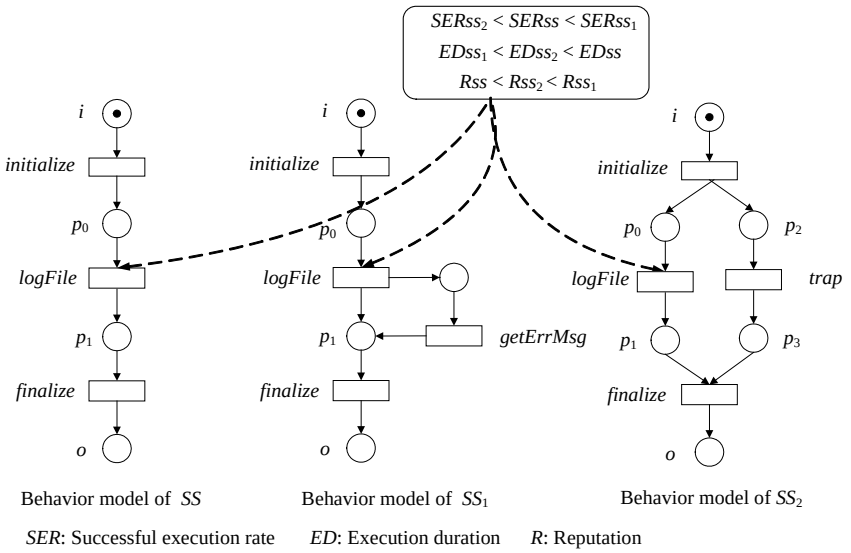

Fig.5 An example of QoS consistent service substitution

图 5 QoS 一致服务替换实例

3 SOBECA 中间件系统的设计和实现

在上述研究工作的基础上,我们设计并实现了一个基于 SOA 架构的中间件平台 SOBECA.由于该平台能够在服务查找和替换时动态支持服务行为的一致性检查和 QoS 匹配,因此,为进一步尝试支持网格计算环境下资源“虚拟化”技术提供了基础支撑.

当前的SOA架构并不支持服务行为一致性检查和QoS匹配,也没有相应的工具包帮助开发和部署服务的行为和质量属性.因此,我们在主流的面向服务的构件模型OSGi上设计了支持服务行为和服务质量的中间件系统.该中间件系统包括一个构建在OSGi[21]平台上的运行支持环境和一个以Eclipse Plugin形式提供的集成开发工具.图 6 给出了SOBECA系统的基础架构.

SOBECA 系统的运行环境自底向上分为 OSGi 平台、扩展中间层和应用层 3 部分.OSGi 平台提供了一些用于部署可扩展和可加载的服务应用的设施;扩展中间层扩展了原有的服务发现、注册和替换机制,加入了服务的行为和质量属性,并在运行时监测服务行为和更新服务质量.扩展服务发现者和扩展服务注册者通过加入服务的行为和质量属性的方式封装了原有的交互机制,实现 bundle 在注册服务时加入服务的行为约束和预定质量属性,扩展服务注册者保存这些信息,并进一步向 OSGi 平台注册服务.当服务发现时,应用可以提出服务行为和质量需求,扩展服务发现者根据底层平台返回的提供者列表,通过检验服务的行为一致和质量匹配找到最佳的服务实现返回给应用.在服务首次被调用时,扩展服务发现者创建服务实现的代理,代理服务检测运行时服务行为和 QoS;应用层包含了各种服务和应用的实现,它们通过行为中间层提供的功能去保障系统行为和质量

属性.算法 1 给出了服务发现时行为调用一致性验证算法,算法 2 给出了服务发现时 QoS 一致性验证算法.

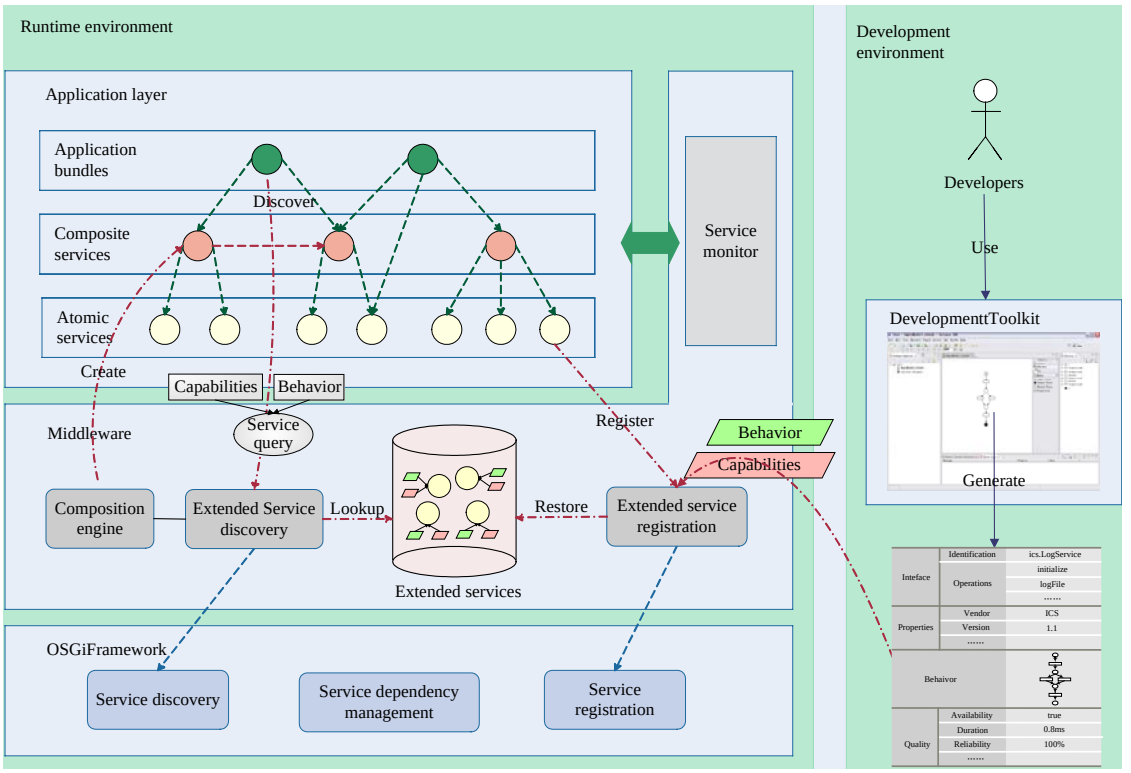

Fig.6 Architecture of SOBECA middleware

图 6 SOBECA 中间件基础架构

算法 1. BehaviorConsistentOfWSDiscovery.

Input: *sr*, *ss*:*WS*.

Output: *SSConsistentSR*: *bool*.

1. BGs_r=*GetBehaviorGraph*(*sr*);
2. *BGss*=*GetBehaviorGraph*(*ss*);
3. *RGsr*=*GetReachibilityGraph*(*BGsr*);
4. *RGss*=*GetReachibilityGraph*(*BGss*);
5. *LMSsr*=*depth-first-search*(*RGsr*);
6. *LMSss*=*depth-first-search*(*RGss*);
7. if *LMSsr LMSss*
8. return true;
9. else
10. return false;
11. end

算法 2. QoSConsistentOfWSDiscovery.

Input: *sr*, *ss*:*WS*, *opi*:*OP*, *policy*:{*LOCAL*,*GLOBAL*}.

Output: *SSConsistentSR*: *bool*.

1. if *policy=LOCAL*

```
2.      vsr=getQoSAtOP(opi);
3.      vss=getQoSAtOP(opi);
4.      if bound(vss) bound(vsr)
5.          return true;
6.      else
7.          return false;
8.  else if policy=GLOBAL
9.      for (op=opi; op!=NULL; op=op.next)
10.         vsr=vsr+getQoSAtOP(op);
11.         vss=vss+getQoSAtOP(op);
12.     if bound(vss) bound(vsr)
13.         return true;
14.     else
15.         return false;
16. end
```

SOBECA 系统的开发环境是基于 Eclipse 的.图 7 是一个开发环境界面,通过 Eclipse plugin 将定义服务行为的 Petri net 图形工具集成到 Eclipse 中,开发人员可以通过拖放基于 Petri net 的图形元素来定义服务行为,通过添加和赋值属性来描述 QoS.服务行为图和 QoS 属性信息最终转化为 XML 描述文件,通过扩展 bundle 中 manifest 的头信息指明文件名,供扩展中间层在运行时使用.XML 描述文件为了与 OSGi 中标准的服务注册发现机制相兼容,继承了 OSGi 相关的标签.

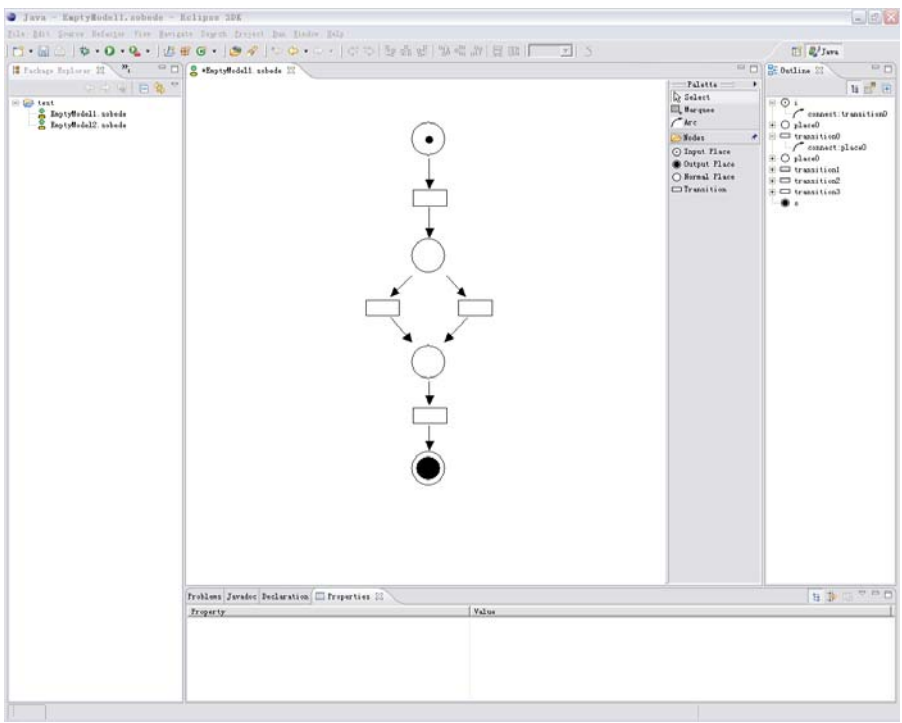

Fig.7 Screen snapshot of SOBECA IDE

图 7 SOBECA 开发环境界面图

4 相关工作

与本文研究工作相关的方面包括:将行为和QoS属性统一建模;描述Web服务行为;有关行为继承理论及应用;在Web服务发现和替换中增加QoS能力等.

在将行为和QoS属性统一建模的方面,典型的工作有文献[22,23].这些工作属于基于模型的性能分析的研究[24].这类研究的方法主要是通过扩展UML中动态模型(状态图或者序列图)的符号来嵌入性能属性等非功能性属性表示,然后,通过精化方法变为随机Petri net[25]、队列网络[26]以及随机进程代数[27]等模型来进行性能量化分析.我们的工作与其相比,首先是将行为模型和QoS向量映射到一个统一的Petri net模型中,使得行为和QoS之间关系的分析更加直观;其次,这些研究大部分是用统一的模型来分析实时系统或关键安全性系统中的实时性以及容错性等性能指标,而我们则是研究网格环境下的服务发现、替换和组合时行为以及QoS一致性的问题.

在描述Web服务行为方面,现有的Web服务规范中的BPEL[28]和OWL-S[29]可以描述Web Services的行为或者工作流特征,相对于规范不具有形式化的特征来说,本文用Petri net描述的行为能够为精确的形式化行为分析提供基础.文献[30]也提出了一种可以描述Web Services行为规范的语言PSML-S.PSML-S由4个模型构成,其中也包括行为模型,可以支持服务动态发现、重配置时服务的一致性检查,但是,其将QoS属性信息定义在结构模型中,与行为模型独立定义.与其他描述行为的形式化模型,如CSP,pi等相比,Petri net具有很好的图形化表述能力和强大的分析方法,并且可以与随机Petri net这样的性能分析模型兼容.

在有关行为继承理论方面,我们综合了文献[11−13]在面向对象范型中的行为继承理论,并将其应用到服务的发现和替换的行为一致性规则中,其中,在服务发现时,服务行为应符合调用一致性规则.在服务替换时,服务行为应符合观察一致性规则.文献[31]描述了根据投影继承关系(与观察一致性等价[19])在一个构件组合框架中替换构件.与这些工作相比,我们将应用领域映射到服务的发现和替换中,并扩展了调用一致性规则来支持服务发现.

在服务发现和替换中增加QoS的能力方面已有许多工作[9,10,32−38].归纳起来,我们的工作可以与这些工作从对QoS的系统支撑、对QoS的信息描述,以及对QoS的优化匹配等角度来进行比较:

- 在对QoS的系统支撑方面,大部分工作都是对现有UDDI进行适合QoS的注册、查询、绑定的扩展.文献[33]提出了G-QoSM系统体系,其中包括一个应用QoS管理器(application QoS manager)来完成对适合QoS的网格服务的查询、协商和管理;文献[36]实现了一个对UDDI扩展的原型系统ExUDDI来正确解析追加在服务发现API中的QoS需求;文献[35]使用一个服务代理(service broker)来完成对QoS的验证、选择和监控;文献[9]实现了一个QoS感知(QoS-aware)的中间件系统AgeFlow;文献[34]实现的MAIS框架可以为大规模的组合服务提供QoS协商和运行时优化;文献[37]通过QoS感知的网格信息服务、QoS感知的网格服务容器、QoS感知的元调度服务来对原有SOA体系结构做适合QoS的扩展.与上述工作相比,我们也采用了扩充UDDI的方案,实现了一个能够支持QoS的中间件SOBECA,其中,通过扩展中间层来完成对OSGi平台上服务的QoS支持.
- 在对QoS的信息描述方面,文献[9]通过服务本体来指定一个服务的多维QoS模型;文献[34]也采用了多维QoS模型,并且初步建立了服务质量与服务操作之间的映射;文献[38]用一种基于本体和可扩充服务质量模型对服务行为和服务质量进行描述,建立了QWSDL语言;而在文献[32]中,作者提出将通用的QoS机制封装到服务构件中.我们的工作与上述工作相比最大的区别在于,明确地将质量信息与行为信息统一描述为Petri net模型并且封装到服务构件中,以XML文件的形式扩展到服务的注册者(registry)中.这为提出更为复杂的对服务行为和服务质量之间关系的综合分析奠定了基础.
- 在对QoS的优化匹配方面,Menasce在文献[10]中提出了服务组合的不同模式会对QoS值有影响;而文献[9,34]也都陈述了类似服务组合对服务质量全局性能的影响,并提出了局部优化匹配和全局优化匹配策略.文献[38]给出了“3层次(基本、基调、规约)、5类型(精确、可替代、包含、相交、松弛)”的匹配模型,本文现阶段的QoS匹配方式是比较简单的单个行为的QoS匹配或者整体行为的QoS匹配,但是由于建立了统一的行为和QoS之间的Petri net模型,因此,借鉴上述服务组合中对QoS优化匹配的研

究,可以在将来支持通过选择不同的行为模式来对 QoS 进行不同策略的优化.

5 总 结

未来的网格软件系统是高度分布、面向服务的[4].以计算资源可以动态分配和共享为特征的网格资源“虚拟化”要求Web服务在适应服务发现、替换和组合这种动态性时,不仅能够提供满足诸如服务行为一致的功能性需求,还要满足像QoS一致这样复杂的非功能性需求[10].因此,现有的Web服务技术必须在模型、方法和中间件实现上做出新型的扩展以适应这种挑战.

本文的工作侧重于将服务行为与 QoS 进行统一描述并封装在服务构件模型中,以更高效地实现在开放、动态、难控的 Internet 环境下网格服务系统在行为一致和服务质量一致的前提下的服务发现和替换.与已有的研究相比,本文的工作有如下特点:

- 提出将服务行为模型与 QoS 模型进行统一描述并封装在服务构件模型中,在此基础上,有利于统一分析服务行为模型和 QoS 模型的最优选择;
- 利用行为继承理论定义服务发现调用一致性和服务替换观察一致性准则,并在参考服务质量因素的情况下将准则进行了适合服务质量一致的扩展;
- 在主流的面向服务的构件模型 OSGi 上开发了支持上述模型和技术的 SOBECA 中间件系统,证实了上述模型和准则的有效性.

本研究组的进一步工作将围绕如下工作展开:

- 现有的工作只强调了单个服务的发现和替换过程,将来希望将研究扩展到服务组合的环境中,也就是将单个服务行为和服务质量的一致性问题扩展到由于服务组合所形成的全局服务组合行为和全局服务质量的一致性问题;
- 在初步建立行为模型和QoS模型之间统一描述的基础上,提出更为复杂的对服务行为和服务质量之间关系的综合分析方法.提出一些可能的最优服务质量的行为模式[39].

References:

[1] Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. 2nd ed., San Fransisco: Morgan Kaufmann Publishers, 2004.

[2] Lü J, Ma XX, Tao XP, Xu F, Hu H. Research and progress on Internetware. Science in China (Series E), 2006,36(10):1037−1080 (in Chinese with English abstract).

[3] Alonso G, Casati F, Kuno H, Machiraju V. Web Services: Concepts, Architectures and Applications. Berlin, Heidelberg: Springer-Verlag, 2004.

[4] Foster I. Service-Oriented science. Science, 2005,308(5723):814−817.

[5] Foster I, Kesselman C, Nick J, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration. Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002. http://citeseer.ist.psu.edu/context/1947008/585645

[6] Alliance G, IBM, HP. Web service resource framework. 2004. http://www.globus.org/wsrf

[7] Huai JP, Hu CM, Li JX, Sun HL, Wo TY. CROWN: A service grid middleware with trust management mechanism. Science in China (Series E), 2006,36(10):1127−1155 (in Chinese with English abstract).

[8] Lu XC, Wang HM, Wang J. Internet-Based virtual computing environment (iVCE): Concepts and architecture. Science in China (Series E), 2006,36(10):1081−1099 (in Chinese with English abstract).

[9] Zeng LZ, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. QoS-Aware middleware for Web services composition. IEEE Trans. on Software Engineering, 2004,30(5):311−327.

[10] Menascé DA. Composing Web services: A QoS view. IEEE Internet Computing, 2004,8(6):88−90.

[11] Basten T, van der Aalst WMP. Inheritance of behavior. Journal of Logic and Algebraic Programming, 2001,47(2):47−145.

[12] Ebert J, Engels G. Observable or invocable behaviour—You have to choose. Technical Report, 94-38, Leiden University, 1994. ftp://ftp.wi.LeidenUniv.nl/pub/CS/TechnicalReports/1994/tr94-38.ps.gz

[13] Harel D, Kupferman O. On object systems and behavioral inheritance. IEEE Trans. on Software Engineering, 2002,28(9):889−903.

[14] van der Aalst WMP. Verification of workflow nets. In: Azema P, Balbo G, eds. Proc. of the 18th Int'l Conf. on Application and Theory of Petri Nets (ICATPN). LNCS 1248, Heidelberg: Springer-Verlag, 1997. 407−426.

[15] Cervantes H, Hall RS. Autonomous adaptation to dynamic availability using a service-oriented component model. In: Proc. of the 26th Int'l Conf. on Software Engineering (ICSE). IEEE Computer Society, 2004. 614−623. http://www.informatik.uni-trier.de/~ley/db/conf/icse/icse2004.html

[16] Yin Q. An extended service-oriented development paradigm: Concept, model and supporting middleware [MS. Thesis]. Nanjing: Nanjing University, 2006.

[17] Yin Q, Hu H, Li J, Ge JD, Lü J. An approach to ensure service behavior consistency in OSGi. In: Proc. of the 12th Asia-Pacific Software Engineering Conf. (APSEC 2005). IEEE Computer Society, 2005. 185−192. http://www.sigmod.org/dblp/db/conf/apsec/apsec2005.html

[18] Yin Q, Hu H, Li J, Ge JD, Lü J. A behavior consistent service discovery and substitution mechanism in OSGi. In: Tsai WT, Hamza MH, eds. Proc. of the IASTED Conf. on Software Engineering and Applications (SEA 2005). IASTED/ACTA Press, 2005. 444−449. http://www.actapress.com/Abstract.aspx?paperId=23904

[19] van der Aalst WMP. Inheritance of dynamic behavior in UML. In: Moldt D, ed. Proc. of the 2nd Workshop on Modelling of Objects, Components and Agents (MOCA 2002). DAIMI Vol.561, Aarhus: University of Aarhus, 2002. 105−120.

[20] Menascé DA. QoS issues in Web services. IEEE Internet Computing, 2002,6(6):72−75.

[21] OSGi. Open services gateway initiative specification. Technical Report, OSGi, 2001. http://citeseer.ist.psu.edu/cis?q=osgi&submit=Search+Citations&cs=1

[22] Cortellessa V, Goseva-Popstojanova K, Appukkutty K, Guedem A, Hassan AE, Elnaggar R, Abdelmoez W, Ammar HH. Model-Based performance risk analysis. IEEE Trans. on Software Engineering, 2005,31(1):3−20.

[23] Bernardi S, Merseguer J. QoS assessment via stochastic analysis. IEEE Internet Computing, 2006,10(3):32−42.

[24] Balsamo S, Di Marco A, Inverardi P, Simeoni M. Model-Based performance prediction in software development: A survey. IEEE Trans. on Software Engineering, 2004,30(5):295−310.

[25] Loopez-Grao JP, Merseguer J, Campos J. From UML activity diagrams to stochastic Petri nets: Application to software performance engineering. In: Dujmovic JJ, Almeida VAF, Lea D, eds. Proc. of the 4th Int'l Workshop Software and Performance (WOSP 2004). ACM Press, 2004. 25−36. http://www.informatik.uni-trier.de/~ley/db/conf/wosp/wosp2004.html

[26] Petriu D, Shousha C, Jalnapurkar A. Architecture-Based performance analysis applied to a telecommunication system. IEEE Trans. on Software Engineering, 2000,26(11):1049−1065.

[27] Pooley RJ. Using UML to derive stochastic process algebra models. In: Davies N, Bradley J, eds. Proc. of the 15th UK Performance Engineering Workshop (UKPEW'99). 1999. 23−34. http://www.cs.bris.ac.uk/Events/UKPEW1999/proceedings/

[28] Andrews T, Curbera F, Dholakia H, Goland Y, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, Trickovic I, Weerawarana S. Business process execution language for Web services, Version 1.1. Technical Report, BEA Systems, Int'l Business Machines Corporation, Microsoft Corporation, 2003. http://citeseer.ist.psu.edu/cis?q=+Business+process+execution+language+for+Web+services&submit=Search+Citations&cs=1

[29] Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, McIlraith S, Narayanan S, Paolucci M, Parsia B, Payne T, Sirin E, Srinivasan N, Sycara K. OWL-S: Semantic markup for Web services. 2004. http://www.w3.org/Submission/OWL-S

[30] Tsai WT, Paul RA, Xiao BN, Cao ZB, Chen YN. PSML-S: A process specification and modeling language for service oriented computing. In: Tsai WT, Hamza MH, eds. Proc. of the 9th IASTED Int'l Conf. on Software Engineering and Applications (SEA). IASTED/ACTA Press, 2005. 160−167. http://www.actapress.com/Content_Of_Proceeding.aspx?ProceedingID=351

[31] van der Aalst WMP, van Hee KM, van der Toorn RA. Component-Based software architectures: A framework based on inheritance of behavior. Science Computer Program, 2002,42(2-3):129−171.

[32] Menascé DA. QoS-Aware software components. IEEE Internet Computing, 2004,8(2):91−93.

[33] Al-Ali R, ShaikhAli A, Rana O, Walker D. Supporting QoS-based discovery in service-oriented grids. In: Proc. of the Int'l Parallel and Distributed Processing Symp. (IPDPS 2003). Nice: IEEE Computer Society, 2003. 101−109.

[34] Ardagna D, Pernici B. Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering, 2007,33(6): 369−384.

[35] Serhani MA, Dssouli R, Hafid A, Sahraoui HA. A QoS broker based architecture for efficient Web services selection. In: Proc. of the 2005 IEEE Int'l Conf. on Web Services (ICWS2005). IEEE Computer Society, 2005. 113−120. http://www.informatik.uni-trier.de/~ley/db/conf/icws/icws2005.html

[36] Guo DK, Ren Y, Chen HH, Xue QW, Luo XS. A QoS-guaranteed and distributed model for Web service discovery. Journal of Software, 2006,17(11):2324−2334 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2324.htm

[37] Hu CM, Huai JP, Wo TY, Lei L. A service oriented grid architecture with end to end quality of service. Journal of Software, 2006, 17(6):1448−1458 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1448.htm

[38] Hu JQ, Zou P, Wang HM, Zhou B. Research on Web service description language QWSDL and service matching model. Chinese Journal of Computers, 2005,28(4):505−513 (in Chinese with English abstract).

[39] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1995.

附中文参考文献:

[2] 吕建,马晓星,陶先平,徐锋,胡昊.网构软件的研究与进展.中国科学(E 辑),2006,36(10):1037−1080.

[7] 怀进鹏,胡春明,李建欣,孙海龙,沃天宇.CROWN:面向服务的网格中间件系统与信任管理.中国科学(E 辑),2006,36(10): 1127−1155.

[8] 卢锡城,王怀民,王戟.虚拟计算环境 iVCE:概念与体系结构.中国科学(E 辑),2006,36(10):1081−1099.

[36] 郭得科,任彦,陈洪辉,薛群威,罗雪山.一种 QoS 有保障的 Web 服务分布式发现模型.软件学报,2006,17(11):2324−2334. http://www.jos.org.cn/1000-9825/17/2324.htm

[37] 胡春明,怀进鹏,沃天宇,雷磊.一种支持端到端 QoS 的服务网格体系结构.软件学报,2006,17(6):1448−1458. http://www.jos.org.cn/1000-9825/17/1448.htm

[38] 胡建强,邹鹏,王怀民,周斌.Web 服务描述语言 QWSDL 和服务匹配模型研究.计算机学报,2005,28(4):505−513.

胡昊(1975－),男,江苏南京人,博士生,讲师,主要研究领域为软件过程,工作流技术,软件行为理论,软件 Agent 技术.

殷琴(1981－),女,硕士,主要研究领域为软件行为理论.

吕建(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件自动化,软件 Agent 技术,网构软件.