ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.8, August 2007, pp.1883−1895
DOI: 10.1360/jos181883

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

用户驱动的服务聚合方法及其支撑框架*

刘譞哲[1,2], 黄 罡[1,2+], 梅 宏[1,2]

[1](北京大学 信息科学技术学院 软件研究所,北京 100871)
[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

Consumer-Centric Service Aggregation: Method and Its Supporting Framework

LIU Xuan-Zhe[1,2], HUANG Gang[1,2+], MEI Hong[1,2]

[1](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)
[2](Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China)
+ Corresponding author: Phn: +86-10-62757670, E-mail: huanggang@sei.pku.edu.cn, http://www.sei.pku.edu.cn/~huanggang

Liu XZ, Huang G, Mei H. Consumer-Centric service aggregation: Method and its supporting framework. *Journal of Software*, 2007,18(8):1883−1895. http://www.jos.org.cn/1000-9825/18/1883.htm

Abstract: The Virtual Computing Environment (VCE) should “on demand” collect and aggregate distributed resources, and provide efficient publication, discovery and subscription mechanisms. Since resources are usually virtualized as services, the goals of the VCE largely rely on service oriented computing technologies. However, Service Oriented Architecture (SOA) organizes services in a “provider-centric” manner, which brings a critical problem: More services come to being, more complex and difficult the service discovery and subscription become. A “consumer-centric” approach is proposed to addressing this problem. First, function similar services according to consumer’s functionality requirements are dynamically aggregated as a service pool. The pool will act as a virtual service so that consumers only discover and subscribe the service pool instead of a large number of candidate services. The qualities of the services in a service pool could form a spectrum of QoS so that consumer’s quality requirements can be satisfied by automated QoS negotiation. Based on the service pool, the VCE could provide an “on demand of consumers” way for resource sharing.

Key words: service oriented architecture; service aggregation; service discovery; service subscription; service pool

摘 要: 虚拟计算环境的重要目标之一,就是在动态、开放、多变的网络环境中对分布异构资源按需进行聚合,并提供有效的资源发布、发现、订阅等机制.通过服务来抽象和封装资源是资源虚拟化的主要手段之一,但现有面向服务的体系结构(service oriented architecture,简称 SOA)所采用的“以服务提供商为中心”的服务组织模式容易导致用户发现和订阅服务的难度随着服务的增加而增加.提出一种用户驱动的服务聚合技术,即根据用户需求,将功能相似的服务聚合成服务池并封装为单一的虚拟服务,随后根据用户 QoS 需求进行自动协商,选出最佳服务或服务组合.该方法的特点在于将服务池作为用户发现和订阅的唯一实体,从用户角度将大量服务聚合成为相对稳定和统一的资源视图,屏蔽资源的复杂性、多样性和多变性,支持自动 QoS 协商,从而有效提高用户对服务资源的利用.

* Supported by the National Natural Science Foundation of China under Grant Nos.90612011, 90412011, 60403030 (国家自然科学基金); the National Basic Research Program of China under Grant No.2005CB321800 (国家重点基础研究发展计划(973))
Received 2007-03-01; Accepted 2007-05-31

关键词: 面向服务体系(service oriented architecture,简称 SOA);服务聚合;服务发现;服务订阅;服务池
中图法分类号: TP393 **文献标识码:** A

随着Internet平台的快速发展与广泛应用,在开放、动态环境下,实现灵活、可信、协同的信息资源(包括计算资源、数据资源、软件资源、服务资源等)共享和利用,已经成为信息化社会的重大需求.虚拟计算环境的目标之一就是将大量成长和自治的资源按照一定的方式加以聚合和综合利用,有效地进行资源发布、发现和组织,为终端用户或应用系统提供和谐、安全、透明的一体化服务的环境[1].其中,资源聚合对实现虚拟计算环境至关重要.通过在一个协同环境中的无缝资源整合,实现对资源的透明和一体化访问,并随着用户需求的变化,按照功能正确性指标、性能指标、安全性指标、可靠性指标等综合指标进行静态调整和动态演化,以尽可能地提高用户满意度.

近年来,基于服务概念的资源封装和抽象逐渐成为开放环境下资源发布、共享和协同的主流技术基础.因此,资源聚合问题也就演变为服务聚合的问题.目前的面向服务的体系结构(service oriented architecture)在服务组织上遵循一种“提供商驱动”的模式,即提供商快速开发服务并发布,服务不断增加,从而出现大量功能相同或相似的服务(如 Google 和 Yahoo!都推出了地图定位服务);而出于市场竞争和企业策略的需要,提供商也将对服务不断扩展和更新,服务功能及质量呈现出持续变化的特点.这种提供商驱动的服务组织模式带来了服务的多样性和多变性,导致服务的描述信息(即有关服务属性的描述信息,如服务接口、服务质量等)不断变化.对于用户来说,在发现和订阅过程中面对的将是一个复杂、多变的动态服务资源视图,很难对服务资源进行透明和一体化的访问.由于缺乏统一的资源视图,服务的发现和订阅的复杂度猛增.通过现有的服务发现模式(如“关键词查找”和“目录浏览”)都会返回大量的结果,不相关或者相关度不大的服务扩大了用户的查找范围,用户不得不对大量服务逐一进行筛选和试用,当选定服务出错后又不得不重复一次类似的发现过程.在服务资源飞速增长的同时,用户对资源的发现和使用却越来越困难,这显然不符合虚拟计算环境的目标.因此,在现有 SOA 的基础上,有必要从用户需求出发,完成对相关服务的汇聚、组织和管理,实现服务的透明化、一体化共享,进而提高虚拟计算环境的用户友好性.

本文提出了用户驱动的服务聚合方法并实现了相应的支撑框架.首先,对服务进行信息建模,根据用户需求将服务组织成相对简单和统一的服务资源视图;其次,建立有效的服务发现与定位机制,根据用户需求自动进行 QoS 协商,返回最佳服务或服务组合;最后,提供一体化的服务资源访问机制,屏蔽服务多样性和多变性带来的复杂性.本文的主要贡献在于:

1) 深入分析现有服务描述和发布方式,给出了基于相似度匹配和领域分类的服务聚合过程;
2) 根据服务聚合的结果自动生成其服务描述信息,从而对用户屏蔽服务多样性;
3) 针对给定的 QoS 需求,设计了 QoS 感知的服务匹配算法以自动完成 QoS 协商.

实验结果表明,该方法有效地实现了服务聚合,简化了用户发现和订阅的工作,并能为用户提供较高的服务质量.

本文第 1 节概述用户驱动的服务聚合方法及其关键技术挑战.第 2 节介绍服务池技术的设计思想和主要过程.第 3 节通过一组实验对该框架在服务聚合精确度、QoS 协商机制和服务调度机制方面进行评估,以验证方法的可行性和有效性.第 4 节介绍相关工作.第 5 节总结全文并展望未来工作.

1 用户驱动的服务聚合方法

1.1 基于服务池的SOA模型扩展

服务聚合需要根据用户需求以及问题复杂度动态地收集服务资源,提供符合服务级别协商(service level agreement)的动态资源供给和管理能力.在文献[2–4]中,我们探讨了服务池(service pool)机制.服务池是一组功能相似但具有不同的服务质量的服务的集合.基于服务池机制,我们扩展了现有的 SOA 模型,如图 1 所示.该扩

展模型的特点是,用户提交功能性和非功能性需求,服务池由此动态创建,首先将满足用户功能性需求的服务资源组织成统一的资源视图提供给用户,从而屏蔽服务多样性,减轻用户选择服务时的负担.然后,服务池根据用户非功能(如 QoS)需求,通过一系列自动化的 QoS 协商机制,选择出最佳的服务或服务组合.用户调用服务时,只需对服务池发出请求,服务池将绑定请求转发给服务资源.在服务绑定和调用中,在约定的 QoS 无法达到时,还需要通过重新绑定服务提供商或通过自适应机制将损失减到最小.

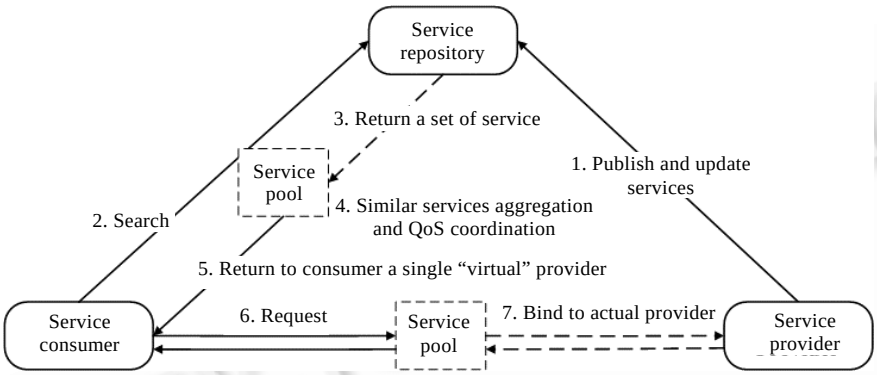


Fig.1 Extended SOA with service pool

图 1 服务池扩展的 SOA

1.2 用户驱动的服务聚合过程

本质上,服务池是一种增强的代理机制,根据用户需求对服务资源进行了重新组织.图 2 表示了其工作过程.

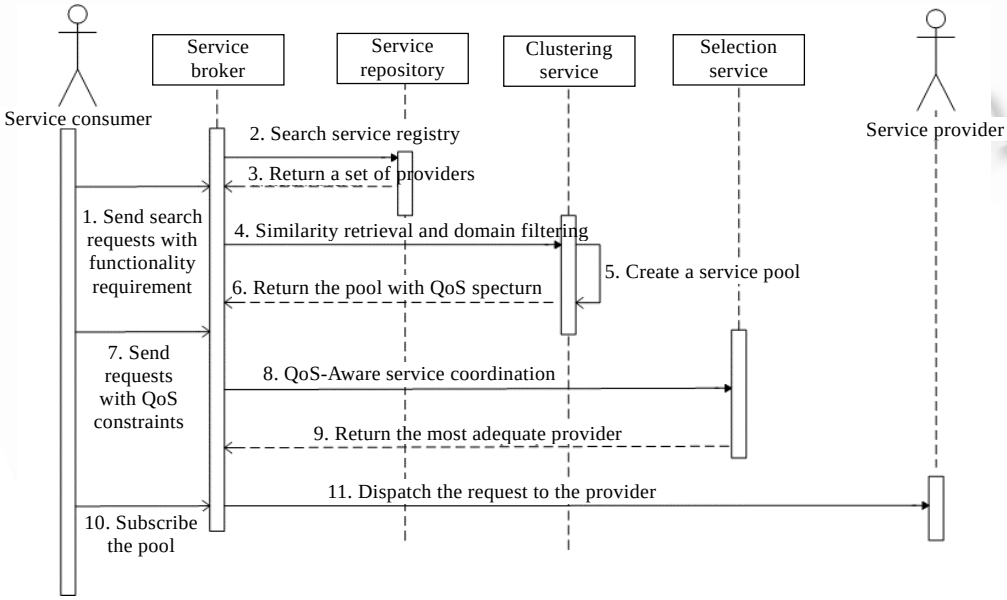


Fig.2 Process of consumer-centric service aggregation

图 2 用户驱动的服务聚合过程

当用户向服务代理(如UDDI或类似于Woogle[5]的服务搜索引擎)提交查询请求后,得到的一组结果不会直接返回给用户备选,而是由服务代理调用Clustering Service对结果进行处理,通过相似度匹配和领域分类,滤去不相干或者相关度不大的服务,生成服务池,作为唯一的服务资源返回给用户,并返回服务质量谱系(QoS

spectrum,包含多个质量维度,每个维度有多个取值).此时,用户仅看到 1 个服务.在参考服务质量谱系后,对服务代理发出服务质量请求,服务代理通过一系列的QoS协商机制,从服务池中得到用户满意度最高的服务或服务组合.最后,用户向服务池发出调用请求,服务池将请求路由到成员服务,并将处理结果返回给用户.

2 支撑框架

2.1 相似服务聚合

实现用户驱动的服务资源聚合,首先需要辨识一组功能相似的服务.由于服务池处理用户请求时需要同时考虑功能性和非功能性因素,因此,需要对服务描述信息建模.根据用户功能需求,通过相似度匹配得到相似服务聚合,则需要定义相似度模型,然后通过相应机制生成服务池.

2.1.1 服务描述信息建模和 QoS 扩展

进行相似服务聚合时,需要在服务语义层次上考察相似度.理想的方法是基于服务语义本体的方法,但目前各个行业的领域本体库正在制定中,到其完备并可以真正投入使用还需要很长一段时间.而且即使形成比较完备的领域本体库,库中的概念实体也很难覆盖服务本体定义中的所有信息.而 WordNet,HowNet 等词汇库比领域本体库相对成熟和完备一些,因此,可以用词汇语义相似度作为度量相似度的手段之一.

WSDL[6]描述了一个Web服务的各项基本信息,如服务名称(服务在UDDI注册的服务名)、所提供的操作和相应输入/输出等,因此,从WSDL层次考察相似服务是一种可行的方法.

定义 1. 根据 WSDL 的 Schema,一个 Web 服务可以标识成一个五元组:

$$WS=\langle ServiceName,Op,Input,Output,Profile\rangle$$

其中,*ServiceName* 是该 Web 服务的名称,*Op*,*Input/Output* 和 *Profile* 分别定义如下:

- 操作(*op*):WSDL 中每个操作 *op* 对应一项功能 *f*(如 *getTemperature*),标识为一个四元组$\langle name,input,output,portType\rangle$,其中,*name* 表示操作名,*input/output* 表示该操作的输入/输出参数,*portType* 为提供该操作的接口.一个 Web 服务所能对外提供的操作为所有操作的集合.
- 输入(*Input*):每个输入标识为一个二元组$i=\langle s_p,op\rangle$,s_p是该输入的参数属性(参数名和类型),*op*则是包含该输入的操作.一个Web服务的输入/输出为其所有操作输入/输出的集合.
- 输出(*Output*):定义与输入类似.
- 服务概要(*Profile*):描述服务的所有黄页和白页信息,如服务在 UDDI 中的注册 ID、服务提供者 ID、服务提供者名称、商业领域分类(*domainkey*).

另一方面,由于服务池需要根据用户需求进行 QoS 协商,因此需要对现有服务注册机制进行扩展:

(1) 增加服务 QoS 描述,以帮助服务发现和协商;

(2) 增加 QoS 监测机制,对注册服务所声称的 QoS 质量、用户对服务的使用反馈进行动态监测,及时对 QoS 进行更新,对不能保证其 QoS 或不可用的服务提供商发出通知或删除.

在服务注册时,需要在统一的服务描述语法和语义中增加关于 QoS 的描述信息.但是,目前 UDDI 规范对其实现细节没有进行严格的规定,也没有提出参考的实现规范.因此,在保证与 UDDI 规范兼容的前提下,可以对其实现规范进行必要的扩展.目前版本的 UDDI 规范更多地关注基于功能约束的服务发现问题,而将服务选择和排序完全交给用户来完成,同时,又没有为用户提供完成这些工作所需的足够的 QoS 信息.因此,需要扩展现有的机制增加 QoS 信息.

我们扩展了现有 UDDI 中的 *add_publisherAssertions* 接口,在 Service Entity 中加入 *qualityInformation* 信息,增加了 3 类 QoS 属性描述,如图 3 所示:

第 1 类是服务提供者提供并注册在信息服务的 QoS 属性,如服务价格;

第 2 类是由服务消费者在使用服务过程中对服务的反馈信息,如服务评分;

第 3 类是在服务提供者与服务消费者之外,由独立的第三方观测到的服务提供者的 QoS 属性,包括服务可信度、响应时间、性能等.

```
- <serviceList generic="1.0" xmlns="urn:uddi-org:api" operator="www.ibm.com/services/uddi" truncated="false">
  - <serviceInfos>
    - <serviceInfo serviceKey="9421dfael-8afd-0yj5-4ea9-3c99b1fa8bf3" domainKey="b42b5fef-85df-4fbf-b468-62a356089ea8">
        <name>GlobalWeatherForecast</name>
      - <qualityInformation>
          <declaration>The quality information is maintained by PKUAS SOA Group</declaration>
        - <updateInformation>
            <publishTime>14:12:22, January 14, 2007</publishTime>
            <lastUpdate>23:37:42, January 31, 2007</lastUpdate>
            <totalUpdate>127</totalUpdate>
          </updateInformation>
        - <qualityAttributes>
            <price>0.85$</price>
            <responsetime>288.2 ms</responsetime>
            <successability>0.972</successability>
            <reputation>neutral</reputation>
          </qualityAttributes>
        </qualityInformation>
      </serviceInfo>
    - <serviceInfo serviceKey="8303kaotr-5wpk-0yj3-2mbv-4k02atcc8af0" domainKey="b42b5fef-85df-4fbf-b468-62a356089ea8">
        <name>WeatherFetcher</name>
      - <qualityInformation>
          <declaration>The quality information is maintained by PKUAS SOA Group</declaration>
        - <updateInformation>
            <publishTime>09:48:44, January 24, 2007</publishTime>
            <lastUpdate>23:37:22, January 31, 2007</lastUpdate>
            <totalUpdate>83</totalUpdate>
          </updateInformation>
        - <qualityAttributes>
            <price>0.70$</price>
            <responsetime>149.2 ms</responsetime>
            <successability>0.998</successability>
            <reputation>high</reputation>
          </qualityAttributes>
        </qualityInformation>
      </serviceInfo>
    </serviceInfos>
  </serviceList>
```


Fig.3 Service description with QoS

图 3 支持 QoS 的服务描述

QoS属性会经常性地发生变化(如网络延时、提供商对QoS进行调整等).为了实现动态环境中的QoS信息描述,我们加入了一个第三方的QoS监测服务.在服务注册时,要求服务提供商提供相应的测试脚本[2],并提交给监测服务.此后,监测服务根据测试脚本周期性地对服务提供商QoS属性进行测试,并收集用户的反馈,更新在UDDI上的注册信息.

2.1.2 相似服务辨识

Web服务相似度识别在Web服务搜索领域已经有了一些成熟的算法[5,7,8].我们讨论了如何利用这些算法进行相似服务辨识[2,4].本节简单介绍该过程,具体算法实现可参见相关文献.

对于两个服务功能f_1,f_2,可以通过分别计算服务名称、提供该功能的操作及其输入和输出以及概要相似度来获得.

定义 2. 根据定义 1,两个Web服务s_1,s_2,判定其提供的功能f_1,f_2是否相似,可定义为

$$Sim(f_1,f_2)=\mu_1 Sim(name_1,name_2)+\mu_2 Sim(op_1,op_2)+\mu_3 Sim(input_1,input_2)+\mu_4 Sim(output_1,output_2)+\mu_5 Sim(p_1,p_2).$$

其中,$name_i$表示s_i服务名称,op_i表示s_i中提供f_i的对应操作,$input_i/output_i$对应op_i的输入/输出参数,p_i表示s_i的概要.$\mu_j(j\in[1,5])$代表各个指标在计算相似度时所对应的权重,可根据情况来指定(例如都取 1/5),$\sum\mu_j=1$.

相似度计算过程简述如下:

- 服务名称相似度.根据WordNet等语义知识词典计算服务名称相似度是基本的词语相似度匹配,可以通过TF/IDF(term frequency/inverse document frequency)[9]来计算.
- 操作和输入/输出参数相似度.一般来说,操作的相似度可以通过计算操作名的词汇语义描述间的相似度来度量,而服务的输入/输出相似度则可以对参数名和类型进行考察[5].从定义 1 发现,操作和输入/输出存在对应关系,在计算相似度时可能会互相影响,例如,City Weather Check[10]服务中获取天气的操作

GetWeatherByCity,输出为⟨*TemperatureF*,*WindChill*,*Humidity*⟩,它们之间存在概念关联(都和“Weather”相关).因此,需要将操作和输入/输出的相似度综合起来考虑.文献[5]给出了根据操作和输入/输出的概念间关联关系进行切分和聚类来计算相似度的算法,具体过程在此不再赘述.

- 概要相似度.服务概要中,对于辨识相似服务最有价值的信息是服务商业领域分类信息.其原因在于,功能相似的服务往往被归类在近似领域中,进行领域相似度计算可以使被聚合的服务相似度更高,提高服务发现的精确度.因此,概要相似度通过计算领域相似度来表示.在UDDI规范中,Web服务领域信息标识为*domainkey*,对应为UDDI分类中的一个节点.在分类树中,两个概念的相似度可以通过计算其语义距离来度量[11].距离越短表示相似度越高.例如,两个提供“按照区号查询天气服务的服务”功能的服务City Weather Check[10]和Fast WeatherFetcher[12]在注册中心[13]中分别注册在Business&Economy和Communication内.根据文献[11]给出的算法得到上述两个领域在UDDI中的距离为 3,可以认为这两个服务处在相似领域内.

经过上述相似度计算后,当两个功能之间的相似度$Sim(f_1,f_2)$大于某个预定值时,则认为s_1和s_2是提供相似功能的服务.

2.1.3 服务池生成

针对某项功能对一组Web服务进行相似度计算后,得到一组提供相似功能的服务$S=\langle S_1,S_2,\ldots,S_n\rangle$,即形成一个服务池.此时,服务池管理层需要记录该服务池的描述信息,包括服务池名称、服务池功能描述和接口规约、成员服务列表及映射关系、QoS范围等,见表 1.

Table 1 Description of a service pool

表 1 服务池描述

Elements	Description
Service pool name	Identifying the service pool
Service pool category	Identifying the functionalities provided by the service pool
Service pool specification	Identifying the operations, inputs and outputs of service pool
Service pool interface	Identifying the global invocation interface for the service pool
Service list	Identifying the services registered in the pool
Mapping	Identifying the mapping between the global interface and the real services in the pool
Qos spectrum	Identifying the QoS spectrum of the pool

为了方便用户调用并支持服务池中服务的灵活变化,服务池被设计为以一种统一的方式访问.对所有访问服务池的用户来说,所得到的服务描述都是一样的.例如,假设 3 个提供“按照区号查询天气服务的服务”功能的服务:

GlobalWeather=⟨*GlobalWeather*,*GetTemperatureByZipCode*,*ZipCode*,*CityTemperature*,*Communication*⟩

City Weather Check=⟨*WeatherFetcher*,*GetWeatherByCity*,*ZipCode*,⟨*TemperatureF*,*WindChill*,*Humidity*⟩,
Business&Economy⟩

FastWeatherFetcher=⟨*WeatherForecaster*,*GetTemperatureByZipCode*,*AreaCode*,
LocalWeatherByZipCodeResult,*Communication* ⟩

为简单起见,μ_j均取 1/5,进行相似度计算后,认为上述 3 个服务可以被归入同一个服务池.使用文献[5]中判定概念关联度的算法对变量进行合并,其服务可被描述为

$$\boldsymbol{SP}=\langle Weather,GetTemperature,Zipcode,Temperature,WeatherReport\rangle.$$

其中,*Weather* 是服务池名称,*GetTemperature* 是操作名,*ZipCode* 和 *Temperature* 是该调用服务池的输入和输出,*WeatherReport* 则表示该服务池提供天气查询功能.需要说明的是,尽管在服务池管理层注册的成员服务会发生变化,但服务池一经形成,其对外服务描述和访问方式就不会再发生变化.这就为用户在请求服务时提供了一个相对统一和稳定的资源视图.

在生成服务池时,所有成员服务的当前 QoS 值将作为一张表在服务池管理层中记录,作为服务池可对外提供的服务质量谱系(QoS spectrum).当QoS监测服务对成员服务作更新时,服务池的QoS质量谱系也将相应发生

变化.

此外,服务池管理层维护服务池和成员服务间的映射关系,不仅需要记录成员服务资源在 UDDI 中注册的 ID 和提供商的 URL,还需要维护服务池描述中的名称空间、接口、操作、参数及其类型的转换.在服务绑定时刻,服务池的 WSDL 作为元信息,通过动态服务发现接口(DDI)、反射机制以及动态编译机制,动态生成成员服务的 *PortType* 和服务代理.

2.2 用户驱动QoS协商

服务池根据用户QoS需求进行QoS协商,找到符合要求的最佳服务或者服务组合.实际上,由于QoS往往是用户选择服务时最为关心的指标,如何从大量的服务中动态地选择最能满足用户需求的服务是一个亟待解决的问题.在这方面已有很多相关研究工作[14−17].一般来说,QoS可以分为定性(用户反馈、可信度等)和定量(价格、响应时间、可靠性等)两种类型.对于定量QoS,我们采用Web服务质量建模规范(Web services quality modeling,简称WSQM)[18]中的定义进行计算;而对于定性QoS,服务提供者和服务消费者的理解可能不一致,为了简化问题的复杂度,我们也参考WSQM的相关定义和区间标度的方法[3],对相关的因素作量化处理.

服务池所能提供 QoS 的范围是成员服务 QoS 的并集.因此,服务池 QoS 向量表示为

$$Q(\boldsymbol{SP})=\{\langle q_{11},q_{12},\ldots,q_{1n}\rangle\langle q_{21},q_{22},\ldots,q_{2n}\rangle\ldots\langle q_{m1},q_{m2},\ldots,q_{mn}\rangle\}.$$

其中,n为服务池中成员服务的数量,m为QoS属性种类,$\langle q_{i1},q_{i2},\ldots,q_{in}\rangle$表示服务池所能提供的第$i$个质量属性的范围.因此,$Q(\boldsymbol{SP})$表示了当前服务池所能提供的QoS属性及其在每个成员服务中对应的值.

用户对目标服务的 QoS 期望值 $q'=\langle q'_1, q'_2,\ldots, q'_m\rangle$,对服务池的 QoS 属性的需求表示为 $Req(q',Q(SP))$,其中,$Req\in\{<,\le,=,\ge,>\}$,需求向量表示为

$$Req(q',Q(SP))=\{Req_1(q'_1,\langle q_{11},q_{12},\ldots,q_{1n}\rangle),Req_2(q'_2,\langle q_{21},q_{22},\ldots,q_{2n}\rangle)\ldots Req_m(q'_m,\langle q_{m1},q_{m2},\ldots,q_{mn}\rangle)\}.$$

Req_i为用户针对第i个QoS属性的需求约束.例如,若$\langle q_{11},q_{12},\ldots,q_{1n}\rangle$表示服务池中所有服务的价格,$q'_1$表示用户期望的价格(200),$Req_1$为“<”时,则表示用户希望得到价格低于 200 的服务.

我们区分了两种不同的基于QoS协商的服务发现方案:

- 声明式服务发现(实时过滤).当服务池生成时,其所能提供的 QoS 范围也随之确定.在实际情况中,用户对 QoS 属性有其偏好排序,总会优先去考察最关心的 QoS.所以,当服务池的 QoS 范围对用户可见时,用户可以按照偏好顺序逐次过滤服务.在使用时,用户选择 QoS 不断地与服务池进行交互,通过 AJAX (asynchronous JavaScript and XML)可实时对服务池的 QoS 属性进行过滤并返回新的结果.如图 4 所示,用户经过某次查询后得到一组相似服务组成的服务池.此时,用户可以看到该服务池可提供的 QoS 属性及其范围,如先考虑价格($price$<200),服务池将过滤掉所有价格大于 200 的服务,然后考虑响应时间,以此类推,直至用户不再有 QoS 需求.这样,服务聚合的结果可以根据用户需求动态地变化,从而体现了用户为中心的思想.
- 编程式服务发现(自动匹配).这种服务发现方式则是用户编写QoS需求模板,在多个QoS维度之间进行自动匹配.这种方法适用于当前服务池中没有服务可以完全满足用户QoS需求的情况.由于基于多维QoS属性的服务选择本质上是背包问题[16],在复杂度上是NP难的,因此,我们设计了一种优化算法[3],对问题进行合理的数学假设,按照用户对QoS的权重建立指标矩阵,将复杂度降到$O(n^2S)$,S为服务池中的服务数量.本文中不详细讨论算法实现和相关定理证明,可以参见文献[3].类似于声明式服务发现,用户也可以通过调整QoS需求模板来完成对聚合的控制.

服务池进行 QoS 协商后的结果将记录其在服务池中的成员 ID,以便于服务池对服务资源进行绑定.

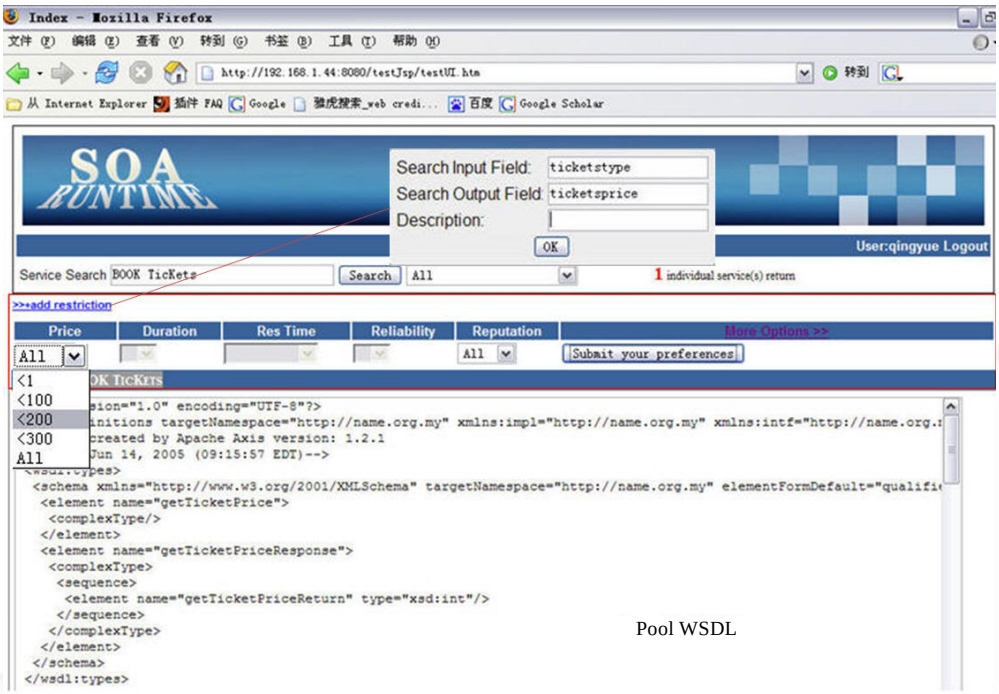


Fig.4 Service discovery user interface

图 4 服务发现用户界面

2.3 服务池访问

服务池需要根据发现结果建立用户和服务池中资源的绑定.其工作过程如下:用户将绑定请求发送给服务池,由服务池代理完成与成员服务资源的绑定,进行参量名称映射和类型转换,调用成员服务,将结果返回给用户.

由于服务池只是虚拟服务资源,运行时刻的实例是其成员服务的实例.因此,需要再提供相应的消息转发和代理生成机制.为了屏蔽不同的绑定请求方式带来的调用复杂性,我们在文献[3]中提出一种基于服务器端请求模式(server-side request),通过扩展 AXIS 的动态调用接口 DII(dynamic invocation interface),同时加入反射机制和动态编译机制,实现了一个统一的、与提供商无关的 Web 服务动态定位和调用接口 DDI(dynamic discovery and invocation interface).工作方式如下:首先,将发现层得到服务池中符合当前用户需求的服务 ID 记录在当前会话中;然后,用户通过 DDI 将绑定和调用请求发送给服务池.此时,DDI 自动创建获取一个 JAX-RPC Service,并调用 *createCall*()方法实例化 JAX-RPC Call 方法(调用 Web 服务的操作).用户使用 setter 方法来配置 Call 实例,调用的参数(名称空间、操作名等)均是服务池的 WSDL 的信息;接下来,DDI 读取发现的服务 ID,查询服务池管理层中的成员映射信息,通过反射机制完成参数名称映射和类型转换,并动态生成服务代理;最后,DDI 将经过转换的调用请求发送到对应的成员服务,完成调用并返回结果.

在实验中我们发现,由于服务池成为实际调用中路由绑定请求的代理,当并发数或者消息长度增加时,服务池会成为性能瓶颈,并造成服务质量的下降.考虑到一些服务资源相对稳定(特别是一些完成重要任务的服务,如在线支付服务,其可靠性在一定时间内保持恒定),针对由这类服务构成的服务池,我们设计了一种静态 Stub Cache 方式(称之为客户端模式),以减轻并在运行时刻服务池的负载.这种方法的工作过程是:服务注册后,服务池管理层启动一个 Agent 来获取服务 Stub,将其保存在服务池中,形成一个稳定的 Stub Cache.只有服务池发生较大变化时,才会重建 Stub Cache.当服务发现层工作时,通过异步消息技术(AJAX)将发现结果的 Stub 下载到本地并生成一个静态代理.当用户调用时,不再通过 DDI 方式,只需简单地调用 Stub,将绑定消息直接发送给成员服

务,从而降低了服务池的负载.

3 系统实现和实验评估

针对前面提出的方法,图 5 给出了一个相应的实现框架.其中,服务注册层集成了 Apache 的 JUDDI 作为服务代理,并对其进行扩展支持 QoS 信息发布和监测.服务池管理层根据用户功能需求,进行功能和领域相似度匹配,生成对应的服务池.服务发现层根据用户质量需求进行 QoS 协商.服务绑定层集成 Apache AXIS,根据协商结果,建立用户和服务资源的绑定,完成服务调用.

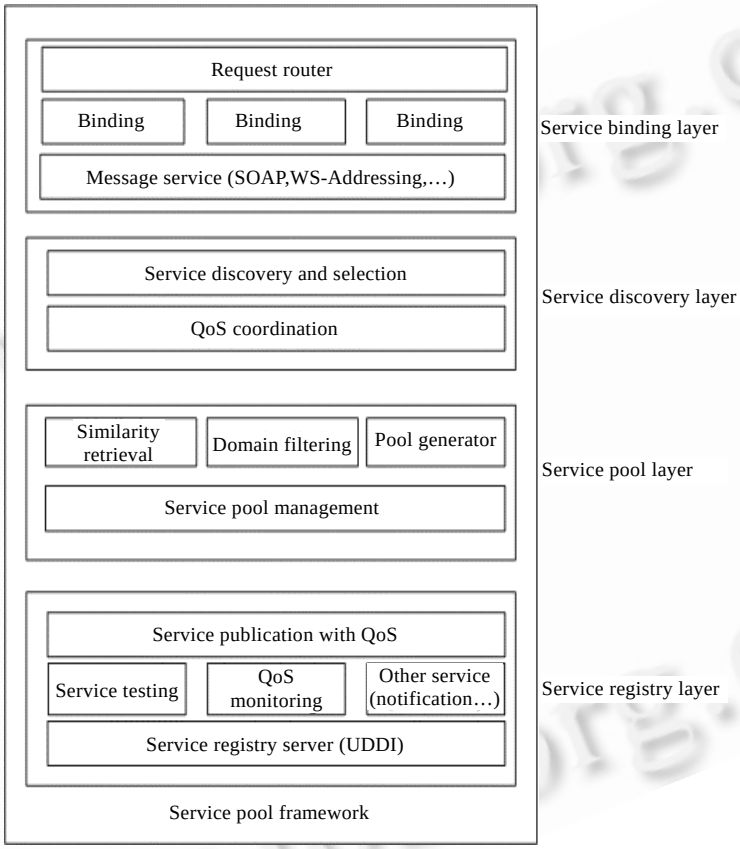


Fig.5 Supporting framework

图 5 支撑框架

应用服务池时,用户首先关心的是服务池所聚合的服务是否满足其功能需求.为了验证基于服务池的服务发现在精确度上的提高,我们设计了一组仿真实验.首先,从较为著名的UDDI服务器[13,19,20]上抓取了约 500 个Web服务,注册在本地的UDDI服务器上(扩展apache JUDDI实现),并对其进行服务聚类;然后,由 5 组用户分别通过 3 种发现模式:基于关键字的查找、文献[5]中的相似服务的查找和基于服务池的查找,查询“天气预报服务”、“股票查询服务”和“订票服务”,并分别对返回的前 5 个、前 10 个和前 30 个服务中的服务进行精确度评分,然后作平均.从图 6 中可以看出,尽管随着返回结果数的增加,整体上精确度会有所下降,但基于服务池的方法和文献[5]的方法,其结果明显优于关键字的方法.特别地,由于文献[5]的方法仅仅考虑到服务在输入/输出和操作上的相似度,可能会返回一些不满足要求的结果(比如,用户希望查找的航空订票服务一般都归在旅游服务领域中,但仅通过参数“book”和“tickets”会返回电影院订票服务,并不符合用户要求).而基于服务池的方法进一步地考虑了领域相似度匹配,在返回结果上精确度更高.

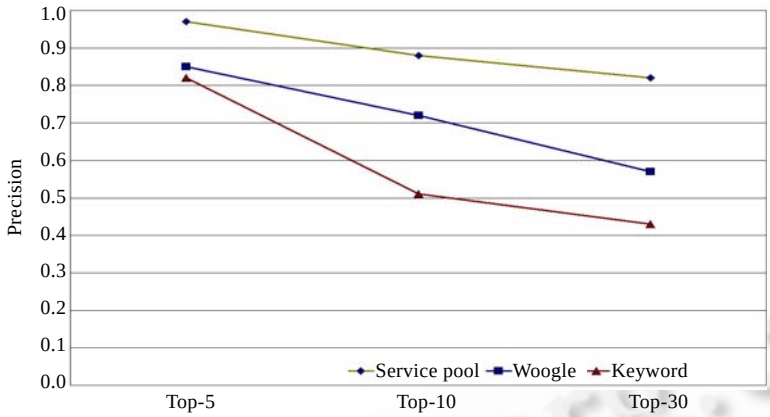


Fig.6 Discovery precision comparison

图 6 发现精确度的比较

图 7 说明了服务池的 QoS 协商机制的性能.根据多维 QoS 需求进行服务发现是一个背包问题.当服务资源数量增加时,服务发现时间将呈指数级增长.为了验证所提出的服务协商机制在服务池增长的同时仍能保证协商时间不会超过可容忍的范围,我们将文献[3]中的 QoS-Aware 算法与回溯算法做了比较.在测试阶段,我们设定 QoS 值更新时间为每 1 小时.从图 7 可以看出,当服务池内服务资源数小于 10 时,两种方法差别不大;但当资源数超过 15 时,协商时间出现了明显的差距.回溯算法随着资源数的增长,时间开销也急剧增长;而我们的算法尽管也出现增长,但幅度不大.其原因在于,我们的算法只有在 QoS 值发生变化时才会重建指标矩阵(该重建过程在服务器完成,与发现过程彼此独立),而回溯则每次都会试图遍历每一种可能符合需求的发现路径.因此,我们的发现算法更适合于存在大量服务资源的情况.

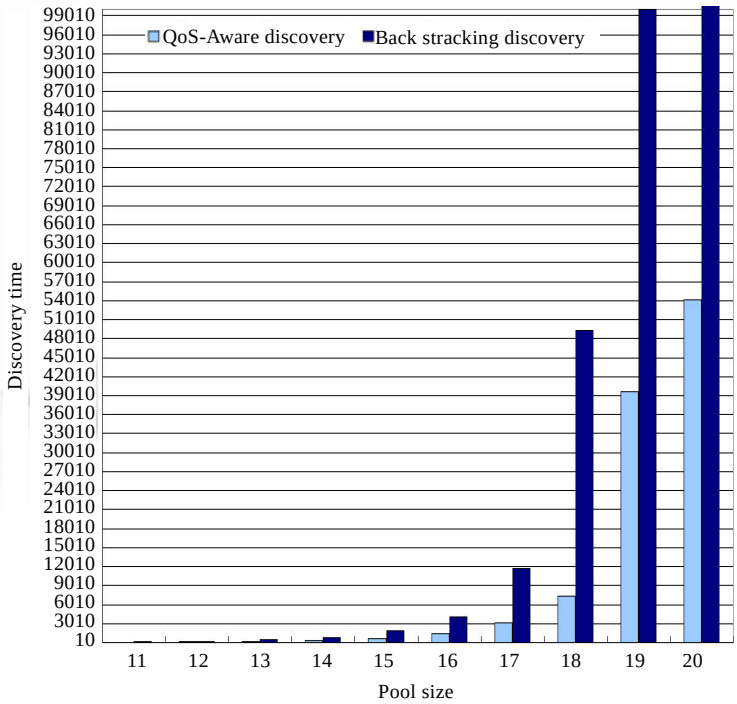
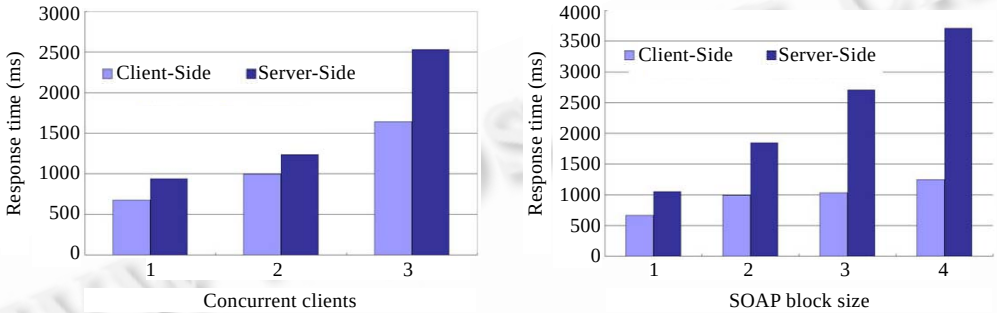


Fig.7 Performance comparison

图 7 性能比较

服务绑定和调用通过服务池路由给成员服务,可能会造成服务池性能下降,从而影响用户的满意度.我们比较了两种情况下客户端方式和服务器端方式的性能开销,如图 8 所示.图 8(a)中给出了当并发请求增大时,每个并发客户端的平均响应时间的比较.服务器端方式明显需要更高的响应时间,其原因在于,每个绑定请求都需要通过服务池转发请求,随着并发数的增大,服务池处理每个客户端的转发开销也随之增加;而客户端方式则不同,尽管平均响应时间仍然呈增长趋势,但服务器端的负载仅仅由于客户端下载 Stub 的影响,绑定和调用是端对端的.同时,对 Stub 的下载是异步进行的,所以平均下来对响应时间影响不大.图 8(b)中则显示了随着 SOAP 消息块大小的变化,单个客户端的平均响应时间.结果仍然表明,客户端的性能优于服务器端方式.随着 SOAP 消息块的增大,服务池的负载将增加,基于服务器端方式生成代理的时间也就更长;而客户端方式不通过服务池进行消息转发,因而响应时间仅会随着消息块变大而缓慢增长.



(a) (b)

Fig.8 Request mode performance comparison: Between client-side and server-side

图 8 客户端请求模式和服务器端请求模式的性能比较

4 相关工作

在面向服务环境下,面对服务资源的不断繁荣,用户查找和访问服务却变得越发困难.将异构的服务资源聚合起来形成相对统一的资源视图,方便用户有效地发现和利用资源,是目前 SOA 和网格计算的重要目标之一.

文献[1]提出了虚拟计算环境(internet-based virtual computing environment,简称 iVCE)的概念,提出资源聚合是实现虚拟计算环境的主要问题,即资源信息建模、资源发布与组织、质量感知的资源发现以及资源路由访问等等.本文所提出的服务池技术正是一种以用户为中心的服务聚合方式,即根据用户需求动态地发现和收集异构服务资源,并提供符合 QoS 协商(service level agreement)的动态资源提供和管理能力,为用户在虚拟计算环境下使用资源提供了有效的支撑机制.

文献[14,15]提出了一种自组织的目录服务社区框架 SELF-SERV.该方法的思想与服务池类似,提出服务社区的概念,将多个可满足相似用户需求的服务组织成一个虚拟服务,处理用户的发现和调用请求.文献[14]重点讨论了目录服务社区服务的发现技术,提出了 P2P 的查询模式和分布式的查询模式.文献[15]则讨论了目录服务社区如何作为服务副本的容器,在运行时刻将用户请求转发给成员服务,以实现动态服务组装.但是,服务社区是根据商务应用的不同种类,将服务也按其行业的相关性划分成不同的类别组成的.这种服务仍然是从服务提供者角度划分,粒度也较粗.本文所提出的服务池方法则是从多个角度考察用户对服务的功能需求来划分服务,因此,聚合的服务资源更符合用户的实际需求.

文献[21]所提出的 VINCA(a visual and personalized business-level composition language for chaining Web-based services)方法,是一种面向服务的最终用户个性化编程体系.该方法的主要特点是以一种自顶向下的设计方式协助用户进行个性化服务创建.通过加入领域知识本体,帮助用户定义业务服务模型,完成服务虚拟化计算(进行服务聚类).由服务社区管理服务资源,允许用户以编程的方式组合服务并定制个性化的服务空间.本文所提出的方法与 VINCA 方法在服务虚拟化方面非常类似,所不同的是,本文的方法更注重在形成用户需求驱

动的服务资源视图后,如何帮助用户快速发现最符合自己需求的服务资源,并提供动态的服务质量管理.比较而言,VINCA 则更关心如何通过服务组合实现用户自定义服务.

文献[22]从服务网格的角度关注资源管理,扩展了服务网格体系结构,设计了基于资源能力管理与服务预留的服务质量保证方法.在具有确定 QoS 保证的网格服务基础上,设计了 QoS 感知的服务发现和服务调度的实现机制.本文的工作更多地考虑从用户角度动态聚合和管理服务资源,力图使用户更加自由和方便地发现和使用资源.

此外,本文中所提出的支持 QoS 的服务发布机制基于对文献[17]中的方法的扩展.所做的扩展是,加入了一个第三方的监测服务,通过服务测试周期性地更新 QoS,并对不合格的服务进行淘汰.这种扩展保证了 QoS 的实时性和有效性,但也存在问题,主要是与现有 UDDI 规范不甚兼容,因为 UDDI 尚不支持第三方修改其结构;而且在现实商业环境中,QoS 的动态获得也不是容易的事情,需要与服务提供商进行协商.

5 结束语和进一步的研究工作

基于服务的资源抽象和封装成为虚拟计算环境下资源组织和发现的重要技术.在服务资源不断涌现的同时,也给用户选择和调用服务带来了困难.主要原因是,现有的服务发现和请求方式都是以提供商为中心,难以满足用户的个性化需求.本文提出一种用户需求为驱动的方法,聚合相似服务形成服务池,并以统一的方式发布供外部访问,可为用户维持相对统一和稳定的资源视图,简化了由于服务资源剧增后用户选择服务时的代价.通过对服务池的 QoS 协商机制和服务绑定方法进行实验,结果表明,服务池有效地降低了服务发现的复杂度,在调用机制上的性能损耗也在可接收的范围内.

本质上,在运行时刻,服务池作为一种代理机制完成用户对成员服务的请求转发.在实际情况中,一些成员服务很可能对安全有特定的要求,如访问控制、安全会话以及消息完整性等.因此,在今后的工作中将考虑服务池和成员服务间的安全管理,提供可靠消息传递服务、身份映射服务、授权服务、安全会话服务等相应机制.此外,我们还将进一步地研究服务聚合机制,基于现有 Web 服务发现技术获得更高的聚合精确度,同时,也将研究动态服务池管理,扩展服务选择算法的 QoS 维度,优化服务绑定机制等.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是北京大学信息科学技术学院软件研究所应用服务器实验组的同学和老师表示感谢.

References:

[1] Lu XC, Wang HM, Wang J. Internet-Based virtual computing environment (iVCE): Concepts and architecture. Science in China (Series E), 2006,36(10):1081−1098 (in Chinese with English abstract).

[2] Liu XZ, Huang G, Mei H. Towards service discovery and subscription based on community-of-Interest. In: Proc. of the 2nd IEEE Int'l Symp. on Service Oriented System Engineering (SOSE 2006). Shanghai: IEEE Computer Society Press, 2006. 167−174.

[3] Huang G, Zhou L, Liu XZ, Mei H, Cheung SC. Performance aware service pool in dependable service oriented architecture. Journal of Computer Science and Technology, 2006,21(4):565−573.

[4] Liu XZ, Zhou L, Huang G, Mei H. Towards a service pool based approach for QoS-aware Web services discovery and subscription. In: Proc. of the ACM 16th Int'l Conf. on Web Wide Web. Banff: ACM Press, 2007. 1253−1254.

[5] Dong X, Halevy A, Madhavan J, Nemes E, Zhang J. Similarity search for Web services. In: Nascimento MA, Tamerozsu M, Kossmann D, Miller RJ, Blakeley JA, Schiefer BK, eds. Proc. of the 30th Very Large DataBase (VLDB) Conf. Toronto: Morgan Kaufmann Publishers, 2004. 272−383.

[6] W3C, Web services description language (WSDL) 1.1. 2001. http://www.w3.org/TR/wsdl

[7] Rodrguez AM, Egenhofer MJ. Determining semantic similarity among entity classes form different ontologies. IEEE Trans. on Knowledge and Data Engineering, 2003,15(2):442−456.

[8] Wu J, Wu ZH, Li Y, Deng SG. Web service discovery based on ontology and similarity of words. Chinese Journal of Computers, 2005,28(4):595−602 (in Chinese with English abstract).

[9] Cover TM, Thomas JA. Elements of Information Theory. New York: Wiley-Interscience, 1991. 245−297.
[10] http://ws.strikeiron.com/InnerGears/WeatherByCity2?WSDL
[11] Maguitman AG, Menczer F. Algorithmic detection of semantic similarity. In: Proc. of the 15th ACM Int'l Conf. on World Wide Web. Chiba: ACM Press, 2006. 107−116.
[12] http://www.webservicex.net/WeatherForecast.asmx?WSDL
[13] Xmethods. 2006. http://www.xmethods.net/
[14] Paik HY, Benatallah B, Toumani F. Toward self-organizing service communities. IEEE Trans. on Systems, Man and Cybernetics, 2005,35A(3):408−419.
[15] Sheng QZ, Benatallah B, Dumas M, Mak EOY. SELF-SERV: A platform for rapid composition of Web services in a peer-to-peer environment. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases (VLDB 2002). Hong Kong: Morgan Kaufmann Publishers, 2002. 1051−1054.
[16] 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.
[17] Ran S. A model for Web services discovery with QoS. ACM SIGCOM Exchanges, 2003,4(1):1−10.
[18] OASIS: Web services quality model (WSQM) TC. OASIS, 2005. http://www.webkorea.or.kr/pds/data/pds1/WSQM-ver-0.3_20050909143621.doc
[19] Web service list. http://www.webservicelist.com/
[20] RemoteMethods. http://www.remotemethods.com/
[21] Hu HT, Li G, Han YB. An approach to business-user-oriented larger-granularity service composition. Chinese Journal of Computers, 2005,28(4):694−703 (in Chinese with English abstract).
[22] 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):1100−1123 (in Chinese with English abstract).

附中文参考文献:

[1] 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):1081−1098.
[8] 吴健,吴朝晖,李莹,邓水光.基于本体论和词汇语义相似度的 Web 服务发现.计算机学报,2005,28(4):595−602.
[21] 胡海涛,李刚,韩燕波.一种面向业务用户的大粒度服务组合方法.计算机学报,2005,28(4):694−703.
[22] 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):1100−1123.

刘譞哲(1980－),男,甘肃兰州人,博士生,主要研究领域为软件工程,面向服务的计算.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件复用,软件构件技术,分布对象技术.

黄罡(1975－),男,博士,副教授,CCF 会员,主要研究领域为软件工程,分布计算与中间件.