带 Cache 的语义 Web 服务发现研究

徐德智 陈稀伟 陈建二

(中南大学信息科学与工程学院 长沙 410083)

摘 要 针对现有语义 Web 服务发现模型的查询效率较低的现状,提出了一种新的基于 Cache 的语义 Web 服务发现模型 SWSDM-Cache。在模型中,对传统 UDDI 框架进行分布式扩展,语义 Web 服务按领域划分,存放在各自的领域 UDDI 库中。在发现过程中首先进行领域匹配,再进行 Cache 搜索。试验结果表明,该模型的服务发现效率较传统的策略有明显的提高。

关键词 Web 服务,OWL-S,UDDI,语义 Web 服务发现,Cache

中图法分类号 TP393.01 **文献标识码** A

Research on Semantic Web Service Discovery with Cache

XU De-zhi CHEN Xi-wei CHEN Jian-er

(College of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract In view of the lower efficiency of query in the existing semantic Web services discovery model, this paper proposed a new Semantic Web Services Discovery Model called SWSDM_Cache. It extended the traditional UDDI to a distributed framework, divided the semantic Web services on domain, and stored them in respective domain UDDI database. In the discovery process, domain matchmaking was executed firstly, and Cache retrieval was executed secondly. Experimental results show that the model significantly improves the efficiency compared with the traditional discovery strategy.

Keywords Web service, OWL-S, UDDI, Semantic Web service discovery, Cache

1 引言

Web 服务发现是指客户以某种方式在不同类型的 Web 服务中找到其想要的服务。它根据用户的需求在注册中心搜索适合用户的服务,是 Web 服务组合、Web 服务编排中的关键部分。

基于传统 UDDI 的 Web 服务发现机制,为 Web 服务提供了信息的注册、发布和发现。但这种方法的不足之处在于它是在语法描述的基础上通过关键字匹配来实现的,不能较好地支持基于概率和语义约束的模糊匹配。而服务的功能描述不是依靠若干关键字就能完整表达的,还需要语义的支持。语义 Web 服务是语义 Web 与 Web 服务相结合而得的产物,利用语义描述和服务本体论是达到该目标的有效途径。

随着 Internet 上 Web 服务数量的日益增多,如何从众多的服务中准确地找到符合要求的服务,成为服务发现的难点和关键。在当前的语义 Web 服务发现研究中,Web 服务发现的效率一直没有得到充分的关注。针对此问题,本文提出了一种新的 Web 服务发现模型。

本文第 2 节介绍了相关研究工作及所存在的问题;第 3 节详细阐述了扩展的 UDDI 框架模型;第 4 节阐述了 SWSDM_Cache 服务发现过程模型,并给出了相应的算法;最后给出了试验结果及分析。

2 相关工作

最近几年,基于 OWL-S 的 Web 服务发现已为语义 Web 服务领域内十分活跃的研究课题,OWL-S(Ontology Web Language for Service)语言能描述 Web 服务的功能和特性等语义信息。如卡内基梅隆大学的 OWL-S/UDDI matchmaker,它将 OWL-S 与 UDDI 结合起来,从而在 UDDI 中添加了语义信息。

OWL-S 规范中并没有提供服务质量(quality of service, QoS)描述的类和属性的详细定义,也没有提供一种有关 QoS 度量的计算和评价方法。现在国内外大多文献[1-3]主要将注意力集中在如何对 OWL-S 进行扩展,从而提供对非功能属性的支持。

现有的语义 Web 服务发现框架按匹配算法的执行地点分为两种:(1)在服务器端执行服务匹配的语义 Web 服务发现框架;(2)在客户端执行服务匹配的语义 Web 服务发现框

到稿日期:2009-09-16 返修日期:2009-12-14 本文受 863 国家重点自然科学基金项目(60970096),湖南省国土资源厅科技计划项目(200718)资助。

徐德智(1963—),男,教授,CCF 会员,主要研究方向为 Web 计算、语义网、人工智能等,E-mail:hunan.xu@mail.csu.edu.cn;陈稀伟(1976—),男,硕士生,主要研究方向为语义网、人工智能,E-mail:teacherchenboy@qq.com(通讯作者);陈建二(1954—),男,教授,博士生导师,主要研究方向为计算机网络、计算机高等算法和优化等。

架。

基于第一种框架，文献[4]提出了一个扩展的 UDDI 互联构架。各 UDDI 节点按编码进行分类存储，不需要进行数据的同步和复制。它没有探讨匹配算法，也没为 QoS 提供支持。随着网络带宽的提高及随机存储器、硬盘存储器成本的降低，在服务器端部署各种缓存将成为可能。

基于第二种框架，文献[5]提出了一个基于本地缓存的 Web 服务发现机制，其优势在于客户可以选择使用哪种匹配算法以及该算法的执行方式，缺点是需要大量的工作针对自身的应用来实现一个特定的匹配算法。另外，当现有的服务端框架组件或者软件随时间改变时，还要花费大量的时间和费用来维护系统。

3 基于 Cache 的改进 UDDI 模型

3.1 传统的 UDDI

UDDI(Universal Description, Discovery and Integration)就是统一描述、发现和集成标准，定义了 Web 服务的发布与发现的方法。通过 UDDI 服务提供者向服务注册处发布 Web 服务描述(WSDL)，服务请求者检索服务注册处(通过查找操作)以获取满足某个标准的一个或多个服务描述，即得到一个满足服务请求要求的服务描述清单。

各个 UDDI 节点之间进行数据的同步和复制，服务请求者从不同的 UDDI 节点进入是没有差别的，它面对的都是整个 UDDI 体系提供的服务发现功能；服务提供者在某一个 UDDI 节点进行服务的发布，然后通过数据同步和复制把服务扩散到整个 UDDI 体系中。这种结构随着 Web 服务呈指数增长，越来越暴露了它的缺点：(1)UDDI 服务注册结构不提供服务功能性和非功能性的描述，即语义信息；(2)服务发现时的匹配过程是基于框架的关键字匹配；(3)传统的 UDDI 是集中式结构，具有较低的可扩展性。

3.2 改进扩展后的 UDDI

根据程序局部性原理，在一定时间段内，服务请求者对于某些 Web 服务的使用次数要大于其他的服务。这些经常使用服务的概率占到了全部的 80%以上[5]，说明服务请求者的服务访问有很大的时间局部性，访问过的数据可能马上又会再次访问。本文在 OWL-S/UDDI matchmaker[6,7] 基础上，引入 Cache，提出了分布式 UDDI 架构框架模型，如图 1 所示。

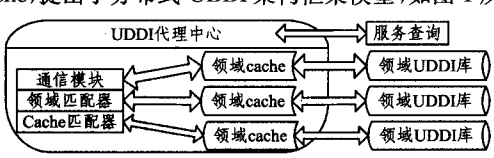

图 1 分布式 UDDI 架构

定义 1(领域 UDDI 库) 指在分布式 UDDI 架构中存储各领域 Web 服务的子 UDDI 节点。

定义 2(领域 CACHE) 指各领域 UDDI 库和总代理中心的缓存，用以存储当前访问概率比较高的 Web 服务。

利用领域本体库对 UDDI 节点进行分类存储，各个 UDDI 节点在 UDDI 代理中心注册，对 UDDI 节点进行分类存储。服务发布和请求时，UDDI 代理使用领域推理机在服务本身和已注册的 UDDI 节点之间进行匹配，并把服务提交给匹配度最佳的 UDDI 节点，进行实际的服务发布和请求处理。通过 UDDI 代理中心把多个领域 UDDI 连接起来，形成一个分布式 UDDI 架构。

它的设计目标是提高分布式系统中服务查询效率以及减少总注册中心服务器的负担。采用这种结构，一方面不需要进行数据的同步和复制，UDDI 代理把服务的发布和请求分配到各个 UDDI 节点上；另一方面通过引入领域本体对 UDDI 节点进行分类存储，充分利用它们之间的领域特性差异。

4 基于 Cache 的语义 Web 服务发现

4.1 SWSDM_Cache 语义 Web 服务发现模型

在这里，我们设计了 SWSDM-Cache 框架模型(见图 2)，主要包含如下。

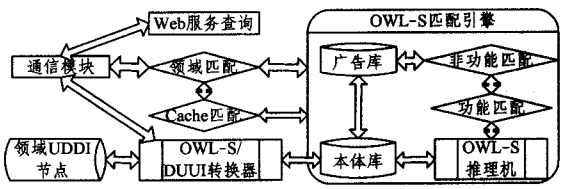

图 2 SWSDM-Cache 架构模型

(1)通信模块。所有外部的请求都由通信模块进入 DSWSD-cache 中，其中包括服务提供者发来的服务广告、服务请求者的查询和请求。

(2)领域匹配。对用户的 Web 服务请求进行领域匹配，相当于匹配引擎进行服务匹配前的一个预处理过程。

(3)Cache 匹配。对用户的查询进行 Cache 内的搜索。

(4)OWL-S/UDDI 的转换器。接受通信模块传过来的服务广告和服务请求，把用 OWL-S 描述的广告和请求翻译成 UDDI 格式，再传给外部的 UDDI 注册中心。

(5)OWL-S 的匹配引擎。图 2 右图是 OWL-S 匹配引擎的内部结构，本文改进了卡内基梅隆大学的 OWL-S/UDDI matchmaker[6]。其中最重要的部分是 OWL-S 的推理机，通过逻辑推理来进行服务匹配，以找到最合适的服务。

(6)领域 UDDI 节点。存储服务提供者发来的服务广告。

4.2 SWSDM_Cache 中的服务匹配

Web 服务发现是建立服务匹配的基础上的，即通过将用户请求与服务注册信息进行匹配，从而选出被匹配的服务。因此，服务匹配是服务发现的重要环节。

4.2.1 服务描述

OWL-S 是用 OWL 语言描述的 Web 服务的本体，它规范了一组用来描述服务的知识本体，使用语义标记使得 Web 服务能够被人和机器理解。OWL-S 的知识本体由 3 部分组成：Service Profile，ServiceModel，ServiceGrouding。ServiceProfile 描述服务是做什么的，Service-Model 描述服务是怎么做的，ServiceGrouding 描述怎么访问服务。

服务质量(QoS)表达了服务满足使用者需求的能力。由于 OWL-S 缺乏对非功能性属性的明确描述，这里对 OWL-S 规范进行扩展，将非功能属性加入到服务描述之中。为了便于讨论，本文只描述 Web 服务的以下几个 QoS 属性：服务价格：使用 Web 服务必须支付的费用，用 *SP* 表示；响应时间：客户从客户端提交对服务 *S* 的请求到获得服务响应的时间，用 *RT* 表示；可靠性：Web 服务正常运行的概率，用 *RB* 表示；调用频率：用 *CF* 表示。

定义 3(Web 服务) 可表示为一个五元组：

$WS=(Name, Description, Input, Output, QoS)$

式中，(1)*Name* 表示服务的名称；(2)*Description* 表示服务的基本描述；(3)*Input*={*i*1,*i*2,…}是服务的输入集合；(4)*Output*={*o*1,*o*2,…}是服务的输出集合；(5)QoS 表示非功能性描述。

4.2.2 领域匹配

各领域 UDDI 中的 Web 服务属同一类服务，用领域本体来描述。

定义 4(领域本体) 用一个二元组来表示：

$Wd=(Name,Description)$

式中，*Name* 表示领域名称，*Description* 表示该领域的基本描述。

下面讨论对用户的 Web 服务请求如何进行领域匹配，并计算名称相似度。

用两个概念的本体距离来计算名称相似性[8]：

$$\mathrm{Sim}(\langle Name_{Rs},Name_{Wd}\rangle)=\frac{dis_{mas}-dis(Name_{Rs},Dame_{Wd})}{dis_{mas}-dis_{\min}} \tag{1}$$

基本描述的相似度为：

$$\mathrm{Sim}(\langle Description_{Rs},Description_{Wd}\rangle)=\frac{dis_{mas}-dis(Description_{Rs},Description_{ud})}{dis_{mas}-dis_{\min}} \tag{2}$$

由式(1)和式(2)计算领域相似度：

$$\mathrm{Sim}(\langle Rs,Wd\rangle)=\frac{1}{2}\mathrm{Sim}(\langle Name_{Rs},Name_{Wd}\rangle)+\frac{1}{2}\mathrm{Sim}(\langle Description_{Rs},Description_{Wd}\rangle) \tag{3}$$

在处理服务的发布和请求时，领域推理机对服务本身的领域特性，即 Web 服务的名称和基本描述，与各 UDDI 节点的领域特性进行匹配，通过式(3)计算领域相似度。相似度比较高的领域，UDDI 就是要找的节点，服务只会提交给这些 UDDI 节点而不是所有的 UDDI 节点。这样会大大提高服务查找的效率。

4.2.3 功能匹配

为简便起见，我们讨论的功能匹配仅指 IO 匹配。引入文献[8]的方法，取参数对之间的平均本体距离作为 *Ps*. *Input* 和 *Rs*. *Input* 之间的语义距离。*Ps* 指提供的服务，*Rs* 指请求的服务，即

$$Dis(Ps.input,Rs.input)=\frac{1}{l}\sum_{a=1}^{l}dis(Ps.input_a,Rs.input_a) \tag{4}$$

结合语义距离以及参数之间的匹配程度，可以采用以下公式计算服务输入参数的相似度：

$$\mathrm{Sim}(\langle Ps.input,Rs.input\rangle)=\frac{dis_{\max}-dis(Ps.input,Rs.input)}{dis_{\max}-dis_{\min}}\times\frac{l}{|Ps.input|+|Rs.input|-l} \tag{5}$$

同样，和式(5)一样有类似的输出参数相似度函数。

4.2.4 服务质量匹配

近几年，国内外 QoS 的 Web 服务匹配研究基于的文献比较多。本文在此讨论的 QoS 匹配主要考虑服务的价格 *SP*、响应时间 *RT*、可靠性 *RB*。本文对文献[9]服务质量相似度函数做了如下改进：

$$\mathrm{SimQos}(Ps,Rs)=\sqrt[3]{qosd(Ps,Rs,SP)qosd(Ps,Rs,RT)qosd(Ps,Rs,RB)} \tag{6}$$

$$qosd(Ps,Rs,SP)=\sqrt[3]{d\ \min(Ps,Rs,SP)davg(Ps,Rs,RT)d\ \max(Ps,Rs,RB)} \tag{7}$$

$$d\ \min(Ps,Rs,SP)=1-\frac{|\min(Ps.SP)-\min(Rs.SP)|}{\min(Rs.SP)} \tag{8}$$

另外几个函数的定义类似，用 *RT*，*RB* 代替式(8)中的 *SP*，用 avg，max 代替式(7)中的 min 即可。

4.3 Cache 管理

4.3.1 一致性校验

分布式环境中 Web 服务是动态的，它在一段时间之后可能被服务提供者修改、注销、暂时关闭甚至是删除。因此，不能按照缓存中失效的信息进行服务调用。如何测试缓存信息的可用性，成为了该服务发现框架可行的一个关键。我们采用两种策略来保证领域 Cache 和领域 UDDI 库的数据一致性：

(1)直写法。即服务提供者将修改的 Web 服务提交给 UDDI 代理中心时，注册程序查看领域 Cache 中是否有相同的标示符。如果有，则将修改的项写入 Cache 相应块中，同时写入领域 UDDI 中。

(2)时间点检测法。即在某一时间间隔后，缓存管理程序定期地按 Web 服务描述中的 Name，Description 检测其目的服务器的可达性和可访问性。如果失败，则移除。

4.3.2 Cache 置换

Cache 置换算法比较多，如 FIFO，LRU，LFU，SIZE 等各种改进策略[10]。我们在这里采用 LRU 算法。它是理想算法很好的近似。在计算机系统中，应用广泛的局部性原理给它打造了坚实的理论基础。而在实际运用中，这一算法也被证明拥有极高的性能。

算法 1 置换算法 CacheReplacement(*Ps*)

输入：当前已经匹配的 Web 服务描述

输出：被替换的 Web 服务描述

Step 1 Cache 容量是否已满，如果没满，转 Step 2，满转 Step 3。

Step 2 将当前访问的 Web 服务载入，并使 QoS. cf 等于 0；其它 Web 服务的 QoS. cf 的值加 1；返回。

Step 3 在当前领域 Cache 搜索 QoS. cf 值最大的 Web 服务。

Step 4 将 QoS. cf 值最大的 Web 服务替换成当前访问的 Web 服务，并置其 QoS. cf 为零。

上述 *QoS*. *cf* 为访问频率，值的大小分别反映访问时间的远近。

4.4 SWSDM_Cache 发现算法

具体的服务发现工作过程通过下面的算法来进行描述和总结。

算法 2 服务发现算法 Discovery(*Rs*)

输入：当前请求的 Web 服务描述

输出：相匹配的 Web 服务描述

Step 1 对用户的 Web 服务请求进行各领域匹配。根据式(3)分别计算领域相似度函数 Sim(⟨*Rs*,*Wd*⟩)，取值最大的领域即所成功匹配的领域。

Step 2 当前领域 UDDI 的 Cache 中是否有匹配的服务，即计算 IO 相似度函数和 QoS 相似度函数是否大于某一阈值。如果有，则匹配成功，转 Step 3；否则，转 Step 4。

Step 3 取 Cache 中匹配的 Web 服务的 QoS. cf 为 0，其它 Cache 中 Web 服务的 QoS. cf 加 1；返回。

Step 4　进一步搜索当前领域 UDDI 本体领域库。计算方法同 Step 2。如果 Web 服务匹配成功，转 Step 5；否则返回 NULL。

Step 5　返回找到的 Web 服务，并触发 Cache 置换函数 CacheReplacement(Ps)。

5　实验结果与分析

5.1　试验准备

本文试验在 Windows XP 环境下开发。采用 myeclipse7.0 作为开发工具，JBossCache 作为 Cache 实现机制。

5.2　试验设置

(1)使用 Protege 工具建立领域本体，包括类、子类和层次关系等信息，存放在 oracle 数据库中。领域库个数为 30。

(2)每个领域服务库中登记了大量用 OWL-S 描述的服务，服务个数 N=30～100 不等。

(3)设定服务的功能描述及非功能描述。

(4)每个服务的输入输出参数个数为 2～5。

(5)Web 服务的总个数为 1000。

(6)依据局部性原理，设计 10 个 50～500 长度的随机访问的 Web 服务队列。

(7)测试的软硬件环境如表 1 所列。

表 1　测试环境

名称	处理器	内存	容量	置换算法	操作系统
指标	2.60 GHz	1G	20	LRU	Windows XP

5.3　试验结果

鉴于目前没有与本实验相关的标准平台和测试数据集，本文采用随机生成的服务作为测试用例。服务本身并不执行有意义的操作，但是这些服务在接口上都遵循 OWL-S 规范。从测试角度看，它们与真实的服务没有区别。

我们分 3 种情况来做对比：基于传统集中式的 UDDI 模型，不带 Cache 的分布式 UDDI 模型，带 Cache 的分布式 UDDI 模型。3 种情况的查找时间如图 3 所示。

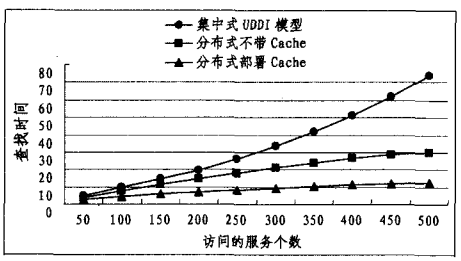

图 3　查找效率比较图

5.4　试验结果分析

从图 3 可以看到，后两者的查询时间随访问的 Web 个数的增加而增加的变化趋势明显地低于前者。

基于传统的集中式 UDDI 的语义 Web 服务发现时间随着访问数量的增加呈线性增长，主要是因为传统的方法需要对请求者的需求和服务注册中心的所有服务进行一一比较，所以效率非常低。

而分布式 UDDI 架构中，借助于领域匹配将实现相似功能和相关主题的服务划分在一起，组成领域 UDDI 库。在服务查找时，根据服务的名称和基本描述可以进入相关的领域 UDDI 查询，从而过滤掉大量无关服务，有效地减少了检索目标服务的数量，显著提高了服务的发现效率。

根据服务请求的局部性原理，引入领域 Cache 概念后，通过统计发现，能使 40%以上的服务请求不需要访问原始 UDDI 库，从而进一步提高了查找的效率。如果存在许多相同的请求，缓存更能大大提高效率。它的查找时间的增长速率明显呈减速状态。

结束语　如何快速、准确和高效地发现满足用户需求的 Web 服务，是现阶段急需解决的关键问题之一。针对这种情况，目前已有一些解决方法，但都或多或少存在着一些问题，并不很完善。本文据此提出了一个新的语义 Web 服务发现模型，在服务器端进行分布式构架，并引入 Cache 机制，先对 Web 服务按领域发布和匹配，再进行 Cache 搜索。试验表明，SWSDM_Cache 模型能显著提高服务发现的效率。

本文的研究只是一个开始，理论上，还有许多问题有待于进一步解决。例如，在考虑查找效率的同时怎样兼顾查找的准确率；采用怎样的预取技术来提高 Cache 的命中率。此外，如何在两个或更多的 UDDI 节点之间进行对同一服务请求的并行查询，将是今后本模型的研究重点。

参 考 文 献

[1] Fortes T G, Frank S. The QoS-MO ontology for semantic QoS modeling[C]// Proceedings of the ACM Symposium on Applied Computing. 2008: 2336-2340

[2] Wu Jinhong, Xia Huosong. Study of adaptive QoS evaluation model for semantic Web service[C]// Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII). 2008, 2: 480-483

[3] Duygu Ç, Atilla E. Semantic QoS model for extended IOPE matching and composition of Web services[C]// Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference) (OMPSAC). 2008: 993-998

[4] 马欣宇，刘琼，钱乐秋. 基于领域本体的分布式 UDDI 互联架构[J]. 计算机工程，2007，33(14)：49-53

[5] 李琳，葛孝堃，吴国文. 基于缓存和代理的 QoS 服务发现机制[J]. 计算机应用与软件，2008，25(12)：144-147

[6] Naveen S, Massimo P, Katia S. An efficient algorithm for OWL-S based semantic search in UDDI[J]. Lecture Notes in Computer Science, 2005, 3387: 96-100

[7] Naveen S, Massimo P, Katia S. Semantic Web service discovery in the OWL-S IDE[C]// Proceedings of the Annual Hawaii International Conference on System Sciences. 2006, 6: 109

[8] 孙萍，蒋昌俊. 利用服务聚类优化面向过程模型的语义 Web 服务发现[J]. 计算机学报，2008，31(8)：1340-1351

[9] 李春梅，蒋运承. 具有 QoS 约束的语义 Web 服务发现的研究[J]. 计算机科学，2007，34(6)：116-121

[10] 胡伟之，沈富可. 基于 Web 访问特性的缓存替换策略[J]. 计算机应用，2008，28(12)：48-50