第34卷 第14期 计 算 机 工 程 2008年7月
Vol.34 No.14 Computer Engineering July 2008
·软件技术与数据库· 文章编号：1000—3428(2008)14—0081—03 文献标识码：A 中图分类号：TP311

基于本体和粗糙集理论的网格服务发现算法

朱郑州，吴中福，邓 伟
(重庆大学计算机学院，重庆 400044)

摘 要：针对网格服务发现的查全率、查准率效率较低的现状，基于本体技术和粗糙集理论，设计了一个服务发现算法OGSDA-RS，在服务匹配之前先进行3步预处理操作：规范化请求服务，根据请求服务对发布服务进行不相关属性约减和依赖属性约减。实验结果表明，与UDDI和OWL-S相比，服务发现的查全率、查准率要高出50%~75%，而且当发布服务的规模较小时，效率比OWL-S最高能高出2.7倍。

关键词：本体；粗糙集理论；网格服务发现；不相关属性约减；依赖属性约减

Grid Service Discovery Algorithm Based on Ontology and Rough Sets Theory

ZHU Zheng-zhou, WU Zhong-fu, DENG Wei
(College of Computer Science and Technology, Chongqing University, Chongqing 400044)

【Abstract】 To improve the recall, precision, and efficiency of grid services discovery, based on ontology technology and rough sets theory, an algorithm OGSDA-RS is designed. Before services matchmaking, according to service request, OGSDA-RS performs the standardization of service request, irrelevant properties reduction and dependent properties reduction. Experiment indicates that the recall and precision of OGSDA-RS tower is above 50%~75% comparing with UDDI and OWL-S. When the number of services advertised is not very large, the efficiency of OGSDA-RS is 2.7 times higher than OWL-S.

【Key words】 ontology; rough sets theory; grid services discovery; irrelevant properties reduction; dependent properties reduction

1 概述

传统的基于UDDI的服务发现采用的发现机制局限于关键字匹配，是一种静态匹配的方式，尽管查找速度比较快，但自动化程度不高，而且不能保证找到所有满足需求的服务[1]。基于OWL的本体技术应用于服务描述，使得服务描述具有语义信息。OWL-S本体中的Service Profile描述了服务的属性，主要有2类：(1)功能属性，如输入、输出、前提条件和后置条件；(2)非功能属性，如服务名称、服务类别和QoS等。基于OWL-S的服务发现能够较好地克服UDDI匹配的弱点，提高服务发现的质量[2-3]。但该方法仍然存在准确度和效率较低的问题。

错误的肯定和错误的否定是影响服务发现准确度的2个主要因素，错误的肯定导致查准率降低，而错误的否定导致查全率降低。因此，为了提高服务的查全率和查准率，应该尽量减少错误的肯定或否定。

粗糙集理论(Rough Sets Theory, RST)用于处理不精确的、不一致的和不完整的各种不完备信息，并从中发现隐含的知识，揭示其潜在的规律。RST特别适用于不要求精确数值结果的不确定性问题[4]。本文结合本体技术，把粗糙集理论应用于服务发现，不但能够提高服务发现的准确度，而且能够降低服务发现的开销，提高效率。

2 算法设计

网格环境下的服务匹配就是请求服务的属性和发布服务的属性之间的匹配，也就是Service Profile中描述的功能属性和非功能属性。为简化起见，本文所采用的服务匹配是基于输入属性的匹配。

2.1 基于RST的网格服务发现模型

本文基于本体技术和粗糙集理论，提出了OGSDA-RS (Ontology-based Grid Service Discovery Algorithm with Rough Sets)。该算法的工作模型如图1所示。

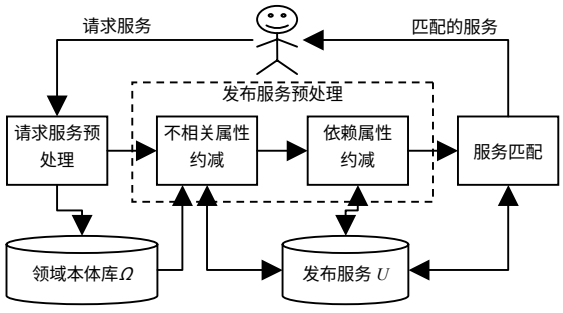

图1 基于RST的服务发现流程

用户向系统发出服务请求，系统根据领域本体，对用户的请求服务信息的描述进行规范化处理，然后根据领域本体库，以请求服务为输入，对发布服务的属性进行不相关属性约减，然后对其进行依赖属性约减，最后对经过预处理规范

基金项目：国家“十一五”重大科技攻关基金资助项目“数字教育公共服务平台中的若干关键技术研究”(2006BAH02A24-6)；国家发展改革委员会科学研究计划基金资助项目“CNGI 远程教学公用通信平台系统”(CNGI-04-15-3A)

作者简介：朱郑州(1979－)，男，讲师、博士研究生，主研方向：计算机网络教育，网格计算，本体技术；吴中福，教授、博士生导师；邓 伟，讲师、博士研究生

收稿日期：2007-08-30 **E-mail：**zzz@cqu.edu.cn

化的请求服务和经过预处理标注的发布服务进行服务匹配，并把匹配结果返回给用户。

定义 1 发布服务集$U=\{u_1,u_2,\cdots,u_N\}$，$N\geq 2$ 为服务提供者发布的所有服务的集合。

定义 2 发布服务的属性集$P=\{p_1,p_2,\cdots,p_K\}$，$K\geq 2$ 为U中任一服务的属性集合。

定义 3 发布服务的相关属性集$P_A=\{p_{A1},p_{A2},\cdots,p_{AM}\}$，$M\geq 1$ 为依据领域本体Ω，与某一请求服务属性相关的发布服务属性集合，其中$P_A\subseteq P$。

定义 4 对于任意 $X\subseteq U$，X 的下近似和上近似分别为：

$\underline{R}(X)=\{u\in U|[u]_P\subseteq X\}$

$\overline{R}(X)=\{u\in U|[u]_P\cap X\neq\Phi\}$

若$\underline{R}(X)\neq\overline{R}(X)$，称$X$为$R$粗糙集。根据RST，对于$\forall p\in P_A$，有：

$\forall u\in\underline{R}(X)$，$u$肯定有属性$p$；

$\forall u\in\overline{R}(X)$，$u$ 可能有属性 p；

$\forall u\in BN(X)$，u 肯定没有属性 p。

其中，$BN(X)=\overline{R}(X)-\underline{R}(X)$，称为$X$的边界。

2.2 发布服务的不相关属性约减

当根据请求服务的描述信息搜索发布服务时，服务请求者使用的属性与服务发布者使用的属性在领域本体中可能不相关，这就需要把发布服务的不相关属性消除掉，然后开始进行服务匹配。

假设p_R是请求服务的属性，p_A是发布服务的属性，两者之间的匹配程度有以下 5 种情况[5]：(1)完全匹配：p_R和p_A是等价的，或者p_R是p_A的子类；(2)插拔匹配：$p_R\subseteq p_A$； (3)包含匹配：$p_A\subseteq p_R$；(4)兼容匹配：$p_A\cap p_R\neq\Phi$；(5)不匹配：$p_A\cap p_R=\Phi$。

定义 5 不相关属性。发布服务中需要约减的不相关属性有 2 类：(1)与请求服务属性不匹配的发布服务的某个属性，标记为“NOMATCH”；(2)在发布服务的本体库中，属性值可能为空的属性。

算法 1 不相关属性约减

输入：请求服务的属性集$P_R=\{p_{R1},p_{R2},\cdots,p_{RN}\}$，发布服务属性集$P_A=\{p_{A1},p_{A2},\cdots,p_{AM}\}$

输出：发布服务的属性标注集$Mark_A=\{Mark_{A1},Mark_{A2},\cdots,Mark_{AM}\}$

算法思想：用请求服务的所有属性去和发布服务的每个属性进行匹配，过滤掉那些发布服务中存在而请求服务中不存在的服务属性，也就是匹配失败的属性，这些属性就是不相关属性。

算法描述：

```
for(i=0;i<N;i++)
{
Mark_Ai=MATCH;
for(j=0;j<M;j++)
{
if(p_Aj与p_Ri不匹配)
continue;
}
if(j>=M)
Mark_Ai=NOMATCH; }
}
```

2.3 发布服务的依赖属性约减

RST 中的依赖属性在服务匹配中是非决定性的。通过对发布服务的依赖属性的约减，可以减少服务匹配的次数，从而提高服务发现的速度。

定义 6 决定性属性集$P_A^D=\{P_{A1}^D,P_{a2}^D,\cdots,P_{AL_D}^D\}\subseteq P_A$，$L_D\geq 1$ 表示在本体Ω中与请求服务R相关的发布服务决定性属性的集合，其中L_D表示决定性属性的个数。

定义 7 不可分辨关系 $ind(P_A^{Dep})=\{(x,y)\in U:\forall p_{Ai}^{Dep}\in P_A^{Dep}$，$f(x,p_{Ai}^{Dep})=f(y,p_{Ai}^{Dep})\}$，其中，$f$ 是从属性到发布服务的映射函数。

定义 8 属性依赖度即属性集P对Q的依赖程度，记为$r_Q(P)$，定义为

$r_Q(P)=card(POS_Q(P))/card(U)$

其中，$card(.)$表示集合的基数；$POS_Q(P)$是属性集Q在$U/ind(P)$中的正区域。

(1)若$r_Q(P)=1$，则称P完全依赖于Q；

(2)若 $0<r_Q(P)<1$，则称P粗糙依赖于或部分依赖于Q；

(3)若$r_Q(P)=0$，则称P完全独立于Q。

定义 9 依赖属性集$P_A^{Dep}=\{P_{A1}^{Dep},P_{A2}^{Dep},\cdots,P_{AL_{Dep}}^{Dep}\}\subseteq P_A$，$L_{Dep}\geq 1$ 表示根据本体Ω，与请求服务R完全依赖的发布服务的属性集合，其中L_{Dep}表示依赖属性的个数。

定理 1 $P_A^D=P_A-P_A^{Dep}$。

算法 2 依赖属性约减

输入：发布服务的属性标注集$Mark_A$，$P_A=\{p_{A1},p_{A2},\cdots,p_{AN}\}$，$P_A^D=\{p_{A1}^D,p_{A2}^D,\cdots,p_{AL_D}^D\}$，$P_A^{Dep}=\{p_{A1}^{Dep},p_{A2}^{Dep},\cdots,p_{AL_{Dep}}^{Dep}\}$

输出：P_A^D

算法思想：把拥有非空属性值的个数最多的发布服务作为查找依赖属性的目标。不使用依赖属性，目标服务仍然可以唯一地识别。为了最大可能地消除依赖属性，阻止个体依赖属性的所有可能组合。

算法描述：

```
for(i=0;i<N;i++)
{
if(p_Ai∈P_A^Dep)
{
P_A^Dep=P_A^Dep+{p_Ai}
P_A^Dep_Core=Ø;
P_A^Dep_Core=P_A^Dep_Core+{p_Ai};
}
}
for(i=2;i<sizeof(P_A^Dep);i++)
{
计算P_A^Dep中i个属性的所有可能组合PC={pc_1,pc_2,⋯,pc_i}；
if(∃pc_k∈PC∧pc_k∈P_A^Dep)
{
P_A^Dep_Core=Ø;
P_A^Dep_Core= P_A^Dep_Core+{pc_k};
continue;
}
else if (PC∩P_A^Dep=Ø)
break;
}
P_A^D=P_A-P_A^Dep_Core;
return P_A^D
```

2.4 网格服务匹配

网格服务匹配就是拿请求服务的各个属性和经过 2 次约减的发布服务的所有属性进行匹配。

定义 10 请求服务属性集$P_R=\{P_{R1},P_{R2},\cdots,P_{RM}\}$表示请求服务$R$中的$M$个属性的集合，其中$M\geq 1$。

定义 11 匹配度$m(p_{Ri},p_{Aj})$是在本体Ω中，请求服务p_{Ri}和发布服务p_{Aj}之间的匹配程度。

定义 12 属性值$v(p_{Aj})$是属性p_{Aj}的一个值，$p_{Aj}\in P_A^D$，$1\leq j$

$\le L_D$。

请求服务属性和发布服务属性之间匹配度的计算规则如表 1 所示。

表 1　匹配度计算规则　(%)

匹配类别	匹配度
完全匹配	100
插拔匹配	75
包含匹配	50
兼容匹配	25
不匹配	0

特别地，如果和服务请求的所有属性都匹配，有空值的决定性属性的匹配度是 50%。

定义 13 相似度 $Sim(R,S)$是请求服务 R 和发布服务 S 之间的相似程度。根据请求服务，每一个发布服务都有一个相似度。

当匹配请求服务的所有属性时，每一个用于鉴别发布服务的决定性属性都有一个最大匹配度，相似度 $Sim(R,S)$可以通过下式计算得出：

$$Sim(R,S)=\sum_{j=1}^{L_D}\sum_{i=1}^{M}\max(m(p_{Ri},p_{Aj}))\Big/L_D$$

算法 3 服务属性匹配

输入：$P_A=\{p_{A1},p_{A2},\cdots,p_{AN}\}$，$P_A^D=\{p_{A1}^D,p_{A2}^D,\cdots,p_{AL_D}^D\}$，$P_A^{Dep}=\{p_{A1}^{Dep},p_{A2}^{Dep},\cdots,p_{AL_{Dep}}^{Dep}\}$，$P_R=\{P_{R1},P_{R2},\cdots,P_{RM}\}$

输出：服务匹配矩阵$\boldsymbol{M}=\{m(p_{Ri},p_{Aj})\}$

算法思想：计算请求服务属性和发布服务属性之间匹配度，根据匹配度对发布服务进行属性约减。

算法描述：

```
for(j=0;j<N;j++)
{
if(pAj∈PA^D∧v(pAj)≠NULL)
{
for(i=0;i<M;i++)
{
if((pRi=pAj)∨(pRiSubclassofpAj))
m(pRi,pAj)=1;
else if(pAi Subsumes pRj)
m(pRi,pAj)=0.75;
else if(pRi Subsumes pAj)
m(pRi,pAj)=0.5;
else if(pRi∩pAj≠Ø)
m(pRi,pAj)=0.25;
else
m(pRi,pAj)=0;   } }
}
for(j=0;j<N;j++)
{
if(pAj∈PA^D∧v(pAj)=NULL)
for(i=0;i<M;i++)
m(pRi,pAj)=0.5;
}
```

3　实验结果

3.1　服务发现的准确度

为评估 OGSDA-RS 算法在网格服务发现中的准确度，本文进行了 5 组测试，每组测试都设计了一些可被发现的目标服务，并分别使用 UDDI 匹配算法、OWL-S 匹配算法和 OGSDA-RS 匹配算法进行服务发现。表 2 是采用 3 种算法进行服务发现的准确度对比。

表 2　服务发现准确度评估表　(%)

不确定服务属性比率	UDDI 匹配		OWL-S 匹配		OGSDA-RS	
	查准率	查全率	查准率	查全率	查准率	查全率
0	50	100	100	100	100	100
30	25	100	0	0	100	100
50	0	0	0	0	100	100
60	0	0	0	0	100	100
70	0	0	0	0	100	100

在第 1 组中，OWL-S 匹配和 OGSDA-RS 算法的查全率和查准率都是 100%，这是因为本组中的所有服务的属性都是确定的。而基于 UDDI 匹配算法发现的服务尽管查全率为 100%，但只有 50%的查准率，这是因为 UDDI 匹配是基于关键词的静态匹配。在后面 4 组的测试用例中，发布服务均包含有不确定属性，因此，基于 OWL-S 的匹配算法查全率和查准率都是 0%。而基于 UDDI 的匹配算法尽管可以发现一些服务，但是查全率和查准率已经大大下降，尤其是当服务的确定属性低于 50%的时候，服务发现的查准率和查全率都是 0%。与此相对应，基于 OGSDA-RS 算法的服务发现却能够发现所有的服务，查全率和查准率都保持在 100%。

3.2　服务发现的效率

服务发现的效率是指服务发现的时间开销，也就是服务发现的速度。为了评估 OGSDA-RS 算法的服务发现效率，本文注册了 15 000 个 generations 服务，并把它的服务发现效率与基于 UDDI 匹配算法、OWL-S 匹配算法进行了比较，如图 2 所示。

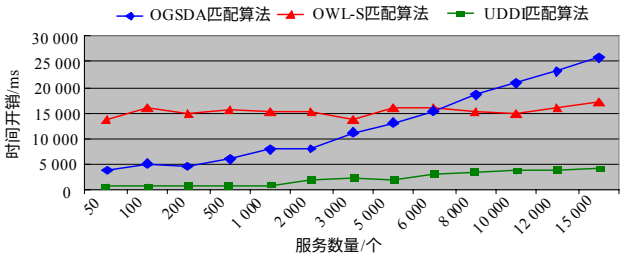

图 2　服务匹配的开销

从图 2 可以看出，UDDI 匹配算法的时间开销一直维持较低的水平，这是因为 UDDI 是基于关键字的静态匹配。而 OWL-S 匹配算法的时间开销一直维持中等水平，这是因为 OWL-S 要经过推理机进行推理，而基于 OGSDA-RS 匹配算法的时间开销一直呈现上升的趋势，当发布服务的数量小于 6 000 的时候，该算法优于 OWL-S 算法，但是当发布服务的数量比较大时(大于 6 000)，服务发现的时间开销越呈线性递增状态，这是因为发布服务的属性约减需要花费时间，发布服务的数量约大，属性约减所花费的时间越多。

4　结束语

本文给出一个基于本体技术和粗糙集理论的网格服务发现算法 OGSDA-RS。该算法在服务匹配之前进行了 3 步预处理操作：规范化请求服务；根据请求服务对发布服务进行不相关属性约减；根据请求服务对发布服务进行依赖属性约减。

本算法的优点是服务的查全率和查准率很高，而且当服务的数量不太大时，服务发现的效率也优于 OWL-S 匹配算法。OGSDA-RS 算法的缺点是当发布服务的数量较大时，服务匹配的时间开销呈线性递增状态。要解决这一问题，可以通过开发并行算法实现服务的不相关属性约减和依赖属性约减过程的并发性，进而降低服务发现的时间开销，提高服务发现的效率。

(下转第 86 页)