文章编号:1001 - 9081(2009)03 - 0892 - 04

任务计算中基于语义 Web 的上下文感知服务选择

黄润才[1,2],庄怡雯[1],周集良[1],曹奇英[1]
(1. 东华大学 计算机科学与技术学院,上海 201620; 2. 上海工程技术大学 电子电气工程学院,上海 201620)
(hrc0427@163.com)

摘 要:针对普适环境中任务计算的特点,分析了任务计算的层次结构和复杂任务的分解策略,设计了一种任务计算中服务发现的体系结构。此体系结构依赖于服务和内容的语义表示,而这种语义表示是基于本体的、共享的。它不仅可以增强精确度和调用率,而且能够实现知识共享、基于能力的搜索、自主推理以及语义匹配等功能。据此,提出了一种基于语义 Web 的上下文感知动态服务选择机制,这种机制可以根据待匹配的服务的动态上下文属性来对其进行过滤和排序,从而进一步优化了发现的过程、节约了用户的时间和精力。此外,该选择机制可用于发现有用的静态或动态上下文信息,从而为用户提供最合适、最相关的服务。

关键词:普适环境;语义 Web;上下文感知;任务计算;服务选择
中图分类号: TP393.01;TP182 **文献标志码**:A

Semantic Web-based context-aware service selection in task-computing

HUANG Run-cai[1,2], ZHUANG Yi-wen[1], ZHOU Ji-liang[1], CAO Qi-ying[1]
(1. *College of Computer Science and Technology, Donghua University, Shanghai* 201620, *China*;
2. *College of Electrical and Electronic Engineering, Shanghai University of Engineering Science, Shanghai* 201620, *China*)

Abstract: The hierarchy of task-computing and the decomposition strategy of complex task were analyzed according to the characteristics of task-computing in pervasive environment. The architecture relied on the semantic representation of service and content based on shared ontology. It could not only improve the accuracy and recall, but also realize the functions such as knowledge sharing, capability-based search, autonomous reasoning and semantic matchmaking. Thus, a dynamic context-aware service selection mechanism based on semantic was proposed to filter and rank the matching services according to their dynamic context in order to optimize the discovery process and save the time and energy of users. In addition, the selection mechanism could be used to discover the useful static and dynamic context information to provide users with the most suitable and relevant services.

Key words: pervasive environment; semantic Web; context-aware; task-computing; service selection

0 引言

任务计算是一个面向用户的体系结构,它使非专业用户能在应用、设备、服务丰富的环境中完成复杂的任务。任务计算为用户提供诸多的方法与普适计算环境进行交互。

传统的面向任务系统存在一些基本的局限性,例如:1)功能被设计在应用中,而应用程序被设计或固化在系统的响应中;2)系统脆弱地依赖于利用输入的语法特性来正确地推测输入的含义;3)系统利用因果机制或触发响应机制,即一定形式的输入对应一个单一的行为或应用激活,没有更复杂用户组合与初始工作流的支持。

任务计算是一个试图解决这些限制的有效模式,是一个实现普适计算的重要方法[1]。普适计算环境所面对的关键挑战是开发服务发现机制,以便用户和应用可以发现并与其最关联的服务进行交互,这些服务是由环境中的很多设备和软件组件提供和发布的。另外,在这种环境中的服务发现技术应及时、安全、有效地处理动态出现和消失的设备和服务,并不侵害用户的隐私。服务发现包括一系列的过程,例如:服务描述、服务发布、服务匹配、服务选择、服务执行等。

上下文感知计算定义上下文为“环境本身以及环境中各实体所明示或隐含的可用于描述其状态(含历史状态)的任何信息,其中,实体既可以是人、地点等物理实体,也可以是诸如软件、程序、网络连接等虚拟实体。”上下文可以划分为三种典型的类型[2]。

计算上下文 如网络的可用性、网络带宽、通信开销等。

用户上下文 包括用户的个性、位置、周围的人员,甚至社会关系等。

物理上下文 如光线的明暗、噪声的大小、交通状况、气候、温度等。

另外,上下文感知是普适计算的一个关键特性,可以定义为“系统的一个属性,它利用上下文为用户提供相关的信息和(或)服务,其相关性取决于用户的任务”。因此,上下文感知服务发现可以被界定为一种功能,能利用上下文信息为用户提供最相关的服务。

语义 Web 是由 Berners-Lee 提出的。他给出的定义是:“语义 Web 是对现有网页的一种扩展,在这种形式下,信息被

收稿日期:2008 - 09 - 16;**修回日期**:2008 - 11 - 06。 **基金项目**:国家教委重点科研项目(104086)。

作者简介:黄润才(1966 -),男,江西东乡人,副教授,博士研究生,主要研究方向:计算机网络与安全、普适计算、Web 技术; 庄怡雯(1983 -),女,上海人,硕士研究生,主要研究方向:智能计算、数据挖掘、普适计算; 周集良(1978 -),男,湖南怀化人,副教授,博士研究生,主要研究方向:计算机网络、普适计算、信息系统; 曹奇英(1960 -),男,浙江宁波人,教授,博士生导师,主要研究方向:优化决策、计算机辅助技术、网络家电。

赋予了定义良好的涵义,从而使得计算机和人们能够更好地进行合作”。语义 Web 的主要目标是为数据引入一个定义良好的涵义,此涵义表示了计算机可理解数据间进行交换的一种通用机器可读格式。通过使用语义 Web 技术,软件代理将能够对数据进行查找、共享、组合、理解以及推理等操作。

本文在分析普适环境中任务计算特点的基础上,探讨了任务计算的层次结构和复杂任务的分解策略,并设计了一种任务计算中服务发现的体系结构。在详细分析了该体系结构的性能特点基础上,本文提出了一种基于语义 Web 的上下文感知动态服务选择机制,并通过待匹配服务的动态上下文属性对选择结果进行了过滤和排序,从而进一步优化了发现过程、节约了用户的时间和精力。实验结果证明,该选择机制还可用于发现有用的静态或动态上下文信息,可为用户提供最合适、最相关的服务。

1　任务计算的层次结构

任务计算(Task Computing, TC)是面向用户的结构,它使得终端用户能在开放的、动态的、具有丰富网络连接的分布式环境中轻松地完成复杂的任务,这种环境中拥有大量的应用、设备和服务。任务计算为用户提供很多与普适环境进行交互的方式,利用语义 Web 作为其核心技术,如 OWL (Web Ontology Language) and OWL-S。在每个环境中,设备、系统、应用等功能也虚拟为服务,通过服务发现机制(如:UPnP),TC 客户端发现这些服务,并获取它们的 OWL-S 文件作为语义服务描述。利用这些 OWL-S 文件,TC 客户端能使终端用户及时操作(组合、执行、发布)这些服务。

图 1 描述了任务计算的层次结构,分别由实现层、服务层、中间件层以及表示层等四个不同的层次组成[3]。

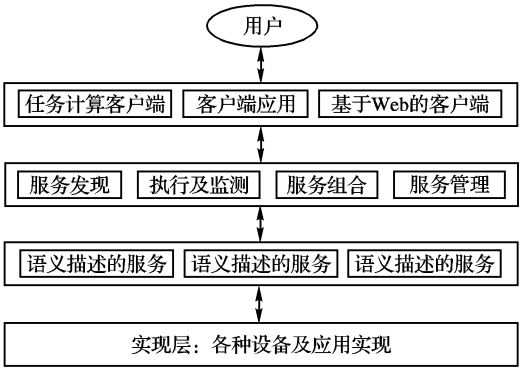

图 1　任务计算的层次结构

1)实现层:最低层包含了大量的设备、应用、电子服务以及其他内容,所有这些功能对用户都是可用的。

2)服务层:这些不同来源的功能形成了可计算的服务,服务的界面用来调用或执行这些功能。每个服务最少与一个语义描述相关联,并且可能被动态地创建。服务是任务计算环境中的功能抽象,而且这些服务的语义描述是为了使用户避开基本功能的复杂性,使得用户能利用这些资源完成自己感兴趣的复杂任务。

3)中间层:这一中间层组件的最终目标是发现服务、决定如何构建服务、执行服务并监测服务的执行,最终完成序列任务的管理,包括创建和发布语义描述的服务。

4)表示层:任务计算最重要的概念就是表示层。利用底层的各种功能为用户提供一个任务,它来自无论哪个底层复杂性的抽象。表示层给用户提供一个环境,在这里可把那些被传递和动态创建的各种功能进行实时组合变成用户执行的任务。由于中间层组件可以开放定义完好的服务 API,因此表示层能针对任何应用环境创建客户端应用。

各层的独立性既是逻辑的也是物理的,它有利于构建独立于系统并能使用户执行复杂任务的环境,从而增强了资源和功能的可利用性。由于动态服务发现、服务的发布和管理、任务的创建及执行等操作的有效性,从技术上看任务计算是完全可行的。

2　任务计算中的服务发现

上下文感知的任务计算体系结构设计以普适环境为平台,它提供一种能唯一识别环境中实体的机制,允许这些实体以安全的方式交换信息,并支持实体的移动性和异构性。图 2 表示了这一机制的整体结构[4]。如图所示,体系结构中的主要实体是用户(User)或代理、共享本体(Shared Ontology, SO)、服务(Services)、发现组件(Dicovery Component, DC)、以及上下文引擎(Context Engine, CE)。每个实体用唯一的 ID 标识,实体之间通过服务特性的数据(s-data)发送和上下文信息(c-info)相关联,它们之间的关系可用式(1)表示如下:

$$\text{TC-ID}(C) = \{\text{U-ID}, \text{SO-ID}, \text{S-ID}, \text{DC-ID}, \text{CE-ID}: \text{s-data}, \text{c-info}\} \quad (1)$$

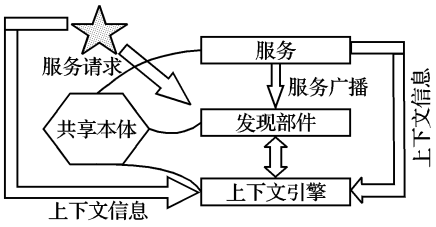

图 2　上下文感知任务计算的体系结构

上下文信息和服务描述采用共享于各实体之中的可读机制本体,上下文引擎负责获取和维护上下文信息,发现组件主要负责存储服务发布,响应来自用户或代理的服务请求。为了灵活性和可扩展性,上下文引擎和发现组件是彼此独立的。无论何时,一旦用户发出了一个服务请求,发现组件协调上下文引擎获取用户的 ID 标示,查询关于用户的上下文信息及可利用的服务。然后,发现组件依据可恢复的上下文信息调用一个语义匹配算法,利用本体的机读格式查询提供的服务,从中发现那些最适合的服务。最后,组件排列匹配的服务,把排列最优先的那些服务的描述反馈给用户/代理。

如果一个匹配服务的描述是合适的,用户/代理便可根据它的调用机制来利用这个服务,这种调用可以是人工的也可自动完成。若是后者,则由软件代理或程序检测发现查询的结果,详细分析调用的必要条件(例如输入/输出的消息、通信机制),并通过提供的相应输入(可能由用户给出)来激活这个服务。另一方面,人工调用需要人为地通过详细分析调用的各个方面并相应地为代理编程来预先配置软件代理。由于采用 OWL-S 本体来描述服务,本模式可在被激活的服务中获取参数的语义描述。因此,本体系结构可以通过开发一个组件来获得对自主服务激活的扩展支持,该组件能够详细描述一个 Web 服务的调用细节,并能自主地激活它而无需任何预先配置和编程。这个组件可以基于 OWL-SAPI 工具平台进行开发,使得 Web 服务的执行可采用 OWL-S 本体进行描述。

2.1 共享本体

Web服务本体OWL-S(Web Services Ontology)是用于描述Web服务的属性、特点和能力的本体论,它从语义角度描述了Web服务。

OWL-S本体由三个主要概念组成。1)ServiceProfile(服务轮廓)用于描述服务的能力(例如,输入/输出)。ServiceProfile描述了Web服务的功能性和非功能性方面,主要用于服务发现。更重要的是,它描述了服务的输入、输出、前提条件、劳动量和结果,其中,能力(输入/输出)则是由本体论或XML模式数据类型中的概念来进行表达的,前提条件和劳动量则是由公式来进行描述的。2)ServiceGrounding(服务基础)用于描述服务调用的细节(例如,通信方案、地址、端口等)。3)ServiceModel(服务模型)用于描述服务的子任务以及这些子任务的执行顺序,后者主要用于将Web服务组合分解成子任务从而实现理想的目标。

上下文信息和服务描述均由基于本体的方法表示,使用共享本体可以增强知识共享,改进推理和基于能力的搜索、确保对环境中所有实体的一致性理解。为了避免一开始就创建属于各自的本体,系统采用本体的可复用性和可扩展性,针对上下文信息的表示,采用了普适应用的标准本体(SOUPA)。SOUPA是一种设计用来模拟和支持普适计算的本体,它重新使用了其他本体的一些概念,例如DAML-Time、DAML-Space以及FOAF[5],代表了普适计算环境中的一般概念,例如:人员、代理、时间、空间、事件等。SOUPA虽然提供了对上下文信息的充分语义表示,但是缺乏对服务的清晰语义描述,因此,系统采用了OWL-S的本体方法,并基于服务发现的需求和目标对其进行了扩展。这种联合本体共享于环境中的实体之间,包括上下文引擎、发现组件、服务、用户、提供者[6]。

2.2 上下文引擎

上下文引擎维护着环境、提供者、用户以及服务的相关信息,这些信息是通过软件、硬件传感器(上下文传感器)以及环境中的服务获得的,在可能的聚集和其他操作完成后存储在共享实体数据库中。为了支持连续的查询、方便上下文交换以及收集,上下文信息的实际价值存储在一个发布/订阅系统中。也就是说,上下文引擎将引用参数存储在共享本体数据库的发布/订阅节点中。为了检索上下文属性的参数值,上下文引擎将收到一个SPARQL查询,上下文引擎解析查询并从相应的发布/订阅节点中检索实际值[7]。

2.3 服务发现

服务发现组件主要负责存储服务的发布信息,与上下文引擎协调响应来自用户/代理的服务请求,如前所述将上下文感知融入服务发现的过程之中可发现最适合的服务。一些现有的发现协议在一定程度上仅通过考虑服务和用户的位置信息来获得上下文,本体系结构在发现最合适的服务时,不仅考虑各种语义描述的上下文信息,而且利用一个动态服务选择和分类机制,节约了用户的精力和时间[8]。

3 服务的选择

3.1 任务计算的分解

任务计算允许用户在富有应用、设备、服务的普适计算环境中去合成和执行复杂的任务。用户有任务执行的需求时,首先根据自身任务的复杂度,在获得相应的上下文信息并进行分析后把复杂任务分解成相对独立的子任务。每个子任务按照自身的特点组合并执行其自身的服务细节,整个任务的完成依赖于各个子任务的分解和子任务中相应服务的执行。任务的分解可由式(2~4)分别表示,相应的分解如图3所示。

$$TC(C) = Subtc(1) \vee Subtc(2) \vee \cdots \vee Subtc(n) \quad (2)$$

$$Subtc(i) = S_1 \vee S_2 \vee \cdots \vee S_m \quad (3)$$

$$S_j = \{SD, SA, SR, SM, SS, SE\} \quad (4)$$

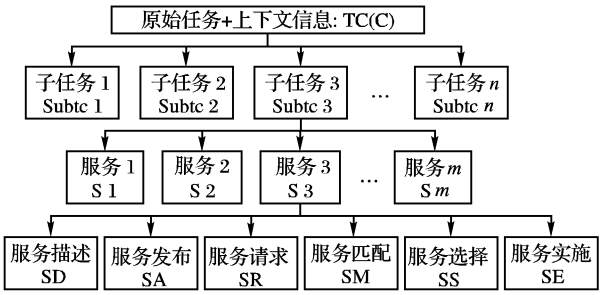

图3 任务计算的分解

3.2 动态服务的选择与排序

通常,在执行了服务请求并检索了匹配服务信息后,发现协议将会把包含结果的反馈信息发送给用户。然后,用户将检查结果并选择其中的一个匹配服务。如果选择的服务不太理想,那么用户将重新检查结果并选择另一个服务直到满意为止。此服务选择进程通常不是很令人满意,这是因为用户可能没有区分匹配服务并选择出最佳服务的能力。鉴于此,我们在体系结构中融入了一种服务选择和排序的机制,从而实现了最佳服务的发现,进而节约了用户的时间和精力。

图4描述了排序策略[9]。

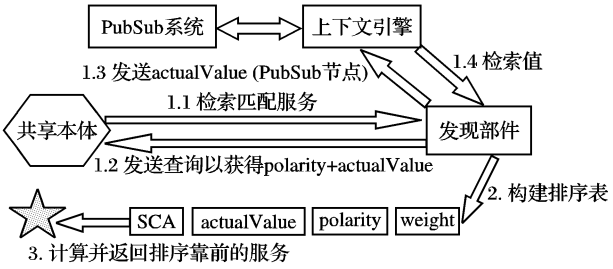

图4 排序策略

第一步,在获取了匹配服务列表后,如果匹配服务数多于2个,那么发现组件将会抽取每个匹配服务的动态上下文信息,本过程可以通过SPARQL查询来搜索匹配服务的ServiceContextAttribute属性的每个实例的polarity值和actual Value值来实现。例如,对于发布了当前打印任务的Print服务请求,发现组件将获取polarity值并将查询请求发送给上下文引擎,然后从适当的pubsub节点获取实际值。由于使用了OWL-S限制,需要保证相关的服务具有相同的SCA集[10]。

第二步,在获取了每个属性的polarity值和实际值后,发现部件将组建每个匹配服务的排序表。排序表由上下文属性列表及其权重、实际值和polarity值组成。权重表示在每个发现阶段中属性的"重要程度"。目前,为了使体系结构更具隐蔽性,所有属性的权重是相同的。当然,也可以根据用户的需求来改变权重,这点可以在请求消息中明确表明或在优先等级中隐含表明。

例如,Alice希望能找到最近的打印机且不考虑打印机当前的负载状态。那么,她的请求可以通过对location属性设置

比其他属性(例如,load 属性)都高的权重来实现。本文所提出的体系结构可以通过扩展来支持上述功能。表 1 表示了打印服务的排序表。

表 1　打印服务的排序表

ServiceContextAttribute 属性	actualValue 值	polarity 值	weight 权重
PrinterLoad	14	-1	0.50
Distance	21	-1	0.50

第三步:发现部件根据匹配服务的排序表来计算每个匹配服务的分值。假定匹配服务的上下文属性的总数为 n,第 i 个属性的 polarity 值为 P_i,第 i 个属性的实际值为 V_i,第 i 个属性的权重为 W_i,根据下面的公式计算每个服务的分值 S。

$$S = \sum_{i=1}^{n} V_i \times P_i \times W_i \tag{5}$$

例如,打印服务的上下文属性如表 1 所示,则 $S = [14 \times (-1) \times 0.50] + [21 \times (-1) \times 0.50) = -7 + (-10.5) = -17.5$。

我们注意到,一些上下文属性(底层上下文)可以被直接发布,例如打印机的 load 属性;而另一些属性则需要在被发布前进行某些处理或集成,例如,光线的状态可以被视为字符串(例如,关闭、开启),然后被转化成整型数值(例如,0/1)。同样地,不能直接提供用户和匹配服务之间的距离,这是通过用户和服务的 GPS 坐标计算得来的,而此信息是通过 LocationCoordinates 类的实例来获取的。上下文属性的数值类型可能有很大的不同。因此,为了保证排序公平,上下文属性的实际值应当先被标准化,其取值范围在 0 到 100 间。可以通过很多方法来实现上述过程,方法之一便是通过使用线形最小化-最大化标准化技术,在这种技术下,排序表中每个上下文属性的最小值和最大值都具有相同的数值类型(例如,所有匹配打印机的最小、最大负载值分别是 3 和 40)。然后,在最小值-最大值的基础上,上下文属性可以被标准化,取值在 0 和 100 之间。

在对每个匹配服务进行了必要的处理、标准化和分值计算后,最后一步便是发现部件在可调阈值的范围内对服务进行排序并将排列靠前的服务在 Jabber/XMPP XML 消息中返回给用户[11]。

4　结语

普适计算正在成为现实,在普适计算环境下,服务发现是一种能满足用户或服务请求并发现服务的基本部件。而服务发现过程中的服务选择是针对服务匹配后多个可选服务的处理策略。目前的服务发现协议大多依赖于服务描述的句法表示和基于关键词的搜索机制。因此,当产生基于句法的服务请求时,即使许多匹配服务是适合用户需求的,但由于它们具有不同句法表示方式,因而是不会被发现的。同样,许多不相关服务被认为是匹配的,而这仅仅是因为它们具有与请求相似的语句表示。换言之,由于诸多协议是基于信息句法表示的,因而通常它们的精确度和调用率都不太理想。除此之外,这些协议不能将用户和服务的上下文信息融入到服务发现阶段,从而它们不能发现最合适的服务。

本文所进行的研究结合了语义 Web 技术与上下文感知计算,通过使用共享本体(语义)的协议,处于不同区域的软件代理或用户可以通过手头的设备请求多种软件和硬件服务并发现最合适的服务。

本文提出的上下文感知服务发现机制可以发现普适计算环境下的上下文信息,从而为请求该服务的用户或代理提供最合适、最相关的服务。系统建立了一个基于 OWL 的本体,它可以通过扩展或修改其他本体来实现上下文发现,从而捕获服务和上下文信息的语义描述。本方案的动态服务选择机制可以根据动态上下文信息来对匹配服务进行排序和区分。

参考文献:

[1] MASUOKA R, PARSIA B, LABROU Y. Task computing - The semantic Web meets pervasive computing [C]// The Semantic Web - ISWC 2003, LNCS 2870. London: Springer, 2003: 866 - 881.

[2] HENDLER J. On beyond ontology[EB/OL]. [2008 - 09 - 01]. http://iswc2003.semanticweb.org/hendler_files/v3_document.htm.

[3] SONG Z, LABROU Y, MASUOKA R. Dynamic service discovery and management in task computing [C]// MobiQuitous 2004. Boston, USA: IEEE, 2004: 310 - 318.

[4] HIDEBRAND J, MILARD P, EATMON R, *et al*. JEP-0030: Service Discovery[EB/OL]. [2008 - 09 - 01]. http://www.jabber.org/jeps/jep-0030.html.

[5] HONG DAN, CHIU D, SHEN V Y. Requirements elicitation for the design of context-aware applications in a ubiquitous environment [C]// Proceedings of the 7th International Conference on Electronic Commerce. New York: ACM, 2005: 590 - 596.

[6] JAEGER M, ROJEC - GOLDMANN G, LIEBETRUTH C, *et al*. Ranked matching for service descriptions using OWL-S [C]// Proceedings of Kummunikation in Verteilten Systemen (KiVS). Berlin: Springer, 2005: 91 - 102.

[7] KHAN O Z. Incremental deployment of context-aware applications [D]. Waterloo, Canada: University of Waterloo, Cheriton School of Computer Science, 2006.

[8] ROBINSON R, INDULSKA J. A context-sensitive service discovery protocol for mobile computing environments [C]// Proceedings of the Fourth International Conference on Mobile Business. Washington, DC: IEEE Computer Society, 2005: 565 - 572.

[9] CUDDY S, KATCHABAW M, LUTYYA H. Context-aware service selection based on dynamic and static service attributes [C]// Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. Montreal, Canada: IEEE, 2005, 4: 13 - 20.

[10] SRINIVASAN N, PAOLUCCI M, SYCARA K. Semantic Web service discovery in the OWL-S IDE [C]// Proceedings of the 39th Annual Hawaii International Conference on System Sciences. Washington, DC: IEEE Computer Society, 2006: 109.2.

[11] ZHU FENG, MUTKA M W, NI L M. Service discovery in pervasive computing environments[J]. Pervasive Computing, 2005, 4(4): 81 - 90.