计算机科学1995Vol. 22№. 4

开放系统中的服务交易

唐雪飞　刘锦德
（电子科技大学微机所　成都610054）

摘　要　Interoperability is a main feature of open system. In open system, how to have dynamic selection of required services, manage these services effectively, and provide location transparence, is one of the most key problems for interoperability. Trading is a very efficient approach to solve the above problem. In this paper, we describe the trader (which is a facility to implement trading technology) in detail from the viewpoint of information and computation. At the end, we summarize the advantages of trader and point out some aspacts to be improved in the future.

关键词　*Trader, Service offer, Context, Open system.*

1　引　言

在一个巨大(跨企业、跨城市、乃至跨国家)的计算机网络环境中，实现可互操作性是构成开放系统的关键。按照文[1]中的定义，可互操作性是指在异种机网络环境中透明动用网内数据、处理能力和其它资源的能力。然而，在一个巨大的网络环境中，往往用互不兼容的技术建立各种应用系统，要使这些系统之间能够互操作非常困难，需要一种新的体系结构来解决这个问题。基于对象的技术是研究者们最为看好的一种途径[5]。此外，在一个巨大的环境中，被动用资源往往变化着，包括所处的位置及能提供的功能等自身的属性都可能发生变化。这种变化是客观存在着的，为了满足灵活性的需要，也希望允许存在这种变化。如何管理这些可能发生变化(或变化着)的资源，提供一个动用者和被动用者之间的灵活的连接方式是互操作性的研究者们必须面对的问题。下面用客户/服务器模型对该问题进行适当的描述，以引出解决的方法。

我们可以把提出动用请求方称为客户，而被动用的资源称为服务器。这种客户/服务器之间的交互作用如图1所示：

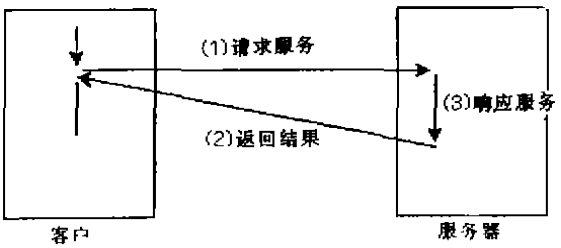

图1　客户/服务器模型

在该交互过程中，客户必须在发出请求之前对服务器有一个足够的了解(包括该服务器是否能满足客户的需要及所在的位置等)。但为了适应变化着的环境，应该使客户对服务器的了解及建立连接发生在客户运行之后。为达到此目的，需要除客户和服务器之外的第三方介入。NCS 的 LB[6]在这方面作了尝试，但它只实现了位置的透明性，即只适应了服务器的位置变化。近年来，出现了一种新的思想——交易器(trader)，能够很好地解决以上提出的问题。下面将从信息和技术的观点对交易器进行详细的描述，并对今后的发展作出展望。

2　什么是交易器

顾名思义，交易器就是一个提供交易的场所，服务器作为"卖"方在交易器中提供服务，客户则作为"买"方在交易器中选择所需的服务。在这里又称服务器为服务的输出者(exporter)，客户则称为服务的输入者(importer)，图2说明了一个交易器与客户/服务器之间的交互作用：

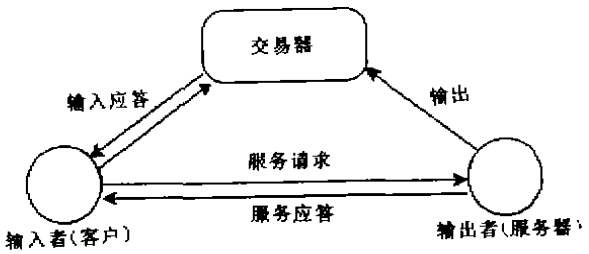

图2　交易器与其使用者

·服务输出：当服务输出者把要提供的服务向外公布时，交易器便可收到来自输出者的一个服务提交(service offer)。一个服务提交包含了输出者愿意提供的某个服务的特征(包括服务所在位置信息)。服务提交由交易器存贮在一个集中或分布的数据库中。

·服务输入：当服务输入者要求一项服务时，交易器便可收到来自客户的一个服务请求。一个服务请求就是输入者在需要某一服务时给出的对该服务的需求表达。

·交易与输入应答：交易器搜索它的服务提交数据库，以匹配输入者的服务请求。交易器选出最合适的服务提交(一个或多个)，返回给输入者。成功地匹配之后，客户便可利用服务提交中的服务器的地址信息与被选择的服务器交互作用。

从以上的过程可以看出，对服务器的匹配和选择是由交易器在运行时间进行的，这使得客户能够在事先对能满足其要求的服务所在位置不甚了解的情况下，便可加入到系统中去。这不仅达到了服务的位置透明性，也使系统具有了可伸缩性(scalability)。下面我们将利用 ODP 的信息观点和计算观点[2,3]来考察交易器。

3 信息观点

从信息的观点出发，可以把交易器的信息成份划分为：服务、服务提交、上下文及其结构、服务请求和输入请求，交易器的信息观点如图3所示。

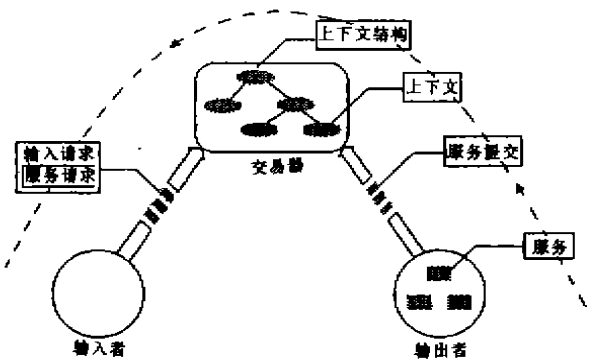

图3 交易器的信息观点

下面将按照图3中虚线箭头所示顺序从服务器开始分别介绍各信息成份。

3.1 服务

所谓服务是指对象在一计算界面上提供的一个功能，是在某对象的界面上可获得的能力集合，可以是下面所列的任何一种：

·一个原子操作，如"写"。

·一个操作序列，如"打开"、"写"、"关闭"。

·一个操作集合，如以任意次序执行的几个"写"操作。

·非操作性的，如提供/耗用一个信息流。

此外，每个服务还都具有相关的属性。

一个服务实际上是一个服务类型的实例。服务类型又由界面类型和服务属性类型构成，即：服务类型＝(界面类型，服务属性类型)。界面类型决定着服务界面的计算行为，也就是说，具有相同类型的界面将提供相同的功能且具有相同的操作特征和相同的计算行为。然而，相同界面类型的服务在一些非计算方面却存在着差别，这便是所谓的服务属性。服务属性是服务属性类型的实例，它由属性名和属性值的集合构成。

从交易的角度看，服务属性可划分成静态和动态两种。静态属性对应于服务的固定属性，如机器的 MIPS 值。由于这种属性即使改变也是极不频繁的，所以存贮在交易器的数据库中。动态属性则对应于频繁变化着的服务属性，如机器的 CPU 负载，动态属性的值存贮在服务方，在需要时由交易器获取。

服务类型及相关的界面类型是存储在一个独立的数据库中，称为类型库(type repository)。

3.2 服务提交

服务提交描述了一个正在被交易的服务。它包括：

·服务界面标识符：标识了一个计算界面，可以通过该界面获得服务。

·输出者标识符：标识了服务提交的输出者(应含位置信息)

·服务类型描述：标识了所提交服务的类型。它可用一个服务类型标识符来描述，或者用一个界面类型标识符加上一组服务属性类型标识符来描述。

·服务属性：输出者对其提交的服务所做的断言。一个服务提交必须为其所有无缺省值的静态属性指定值。

·服务提交属性：关于服务的一次特别提交的断言，比如指定本次提交的期限。

·输出策略控制器的界面标识符：任选。它能使输出者执行附加的动态策略控制。该标识符实际上为交易器提供了与输出者交互作用的句柄，从而使交易器能够获取动态服务属性的值，以及获取一些附加的指导，用以帮助对服务的选择(如执行访问控制检查)。

3.3 上下文及其结构

交易器收到的所有服务提交构成了一个服务提交空间，该空间可以划分成组（集合），称为上下文（context）。每个上下文都具有一定的属性，包括：①上下文自身的某些方面（如服务提交存贮位置的物理结构）；②服务提交的某些共同属性（如只有具有一定安全许可的输入者才能获得的提交）。上下文属性可以用作存放（或查寻）服务提交的依据。具有相同类型的服务可以放在同一上下文中；而要输入该类型的某一服务时，则应在该上下文中查找。

因为上下文是由服务提交组成的集合，所以上下文也遵循集合的一些性质。一个集合可以包含另一集合，被包含集合代表的那个上下文称为子上下文，包含集合代表的上下文称为超上下文。上下文结构表达了上下文（集合）之间的包含关系，如图4所示：

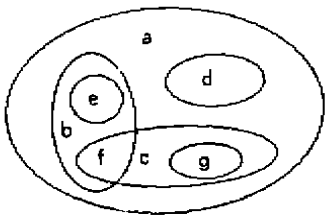

图4 上下文结构

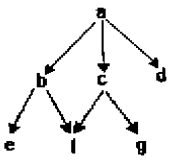

图5 上下文结构表达

在上下文 a 中，可以找到交易器中所有的服务提交。相同的服务提交可以放在不同的上下文中，上下文 b 和 c 就具有相同的服务提交，后者构成了上下文 f。上下文结构可以用一个有向无回路图来表达（如图5所示），其中的节点代表上下文；弧代表上下文之间的关系；箭头指向的节点是子上下文；箭头背向的节点是超上下文；一个节点可以同时是一个上下文的子上下文又是另一上下文的超上下文。

上下文及其结构的目的是便于管理（存贮和查寻）交易器得到的服务提交。

3.4 服务请求

服务请求由客户发出，它请求交易器返回能满足其需求的服务提交。服务请求所含的信息是服务的需求规格说明。该说明包含两部分：一是要求的计算服务行为，用服务类型标识符或界面类型标识符表示；另一部分是要求的属性，用一个匹配约束条件来表示。匹配约束条件是一个布尔表达式，规定了某些属性的取值范围。当某服务提交的相应属性值在指定的范围内时，该布尔表达式便为真，说明该服务提交满足该匹配约束条件。下面是一个匹配约束条件的例子：

Printer_type = Laser and Cost_Page < $1.00

3.5 输入请求

服务请求是输入请求的一部分，输入请求还包含以下三方面的内容：

·输入者的标识：指明请求是由谁发出的，以便交易器返回匹配的服务提交。

·搜索范围：限定在哪些上下文中查找所需的服务提交，也考虑时间限制和搜索后返回表的长短限制等。

·搜索方法：即在指定范围内对上下文进行搜索的顺序。

仅当按照搜索方法在指定的搜索范围内找到服务请求所要求的服务提交时，才能说完成了输入请求的匹配。

4 计算观点

从计算的观点来考查交易器，我们看到的是交易器与其环境之间的界面。这些界面为系统中的其它应用提供了使用交易器的手段，也为交易器访问其周围的环境提供了便利。

这些界面可以分成由交易器提供的界面和交易器可获得的界面。前者又可划分成管理界面集合和交易界面集合；后者则主要涉及输出策略的控制及服务（界面）类型的检查等。交易器的计算观点如图6所示：

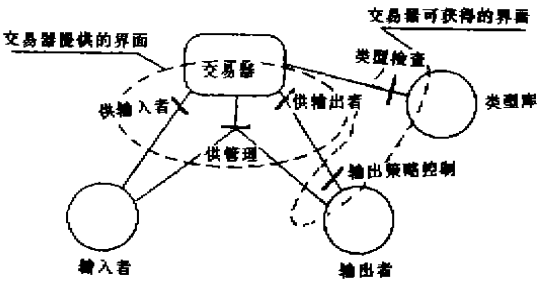

图6 交易器的计算观点

下面分别介绍各种界面所提供的操作。

4.1 管理界面的操作

对交易器的管理主要涉及上下文和输入策略（包括搜索范围和搜索方法）。当然，对交易器提供的交易界面也可管理，但一般采用的是固定的交易界面，不允许外界干涉。

（1）上下文管理操作

·增加上下文：将一上下文加入到上下文结构中（需指定其超上下文）。

·删除上下文：将一上下文从上下文结构中删

除，同时也删除了其子上下文。

·列表上下文：列出指定上下文的标识符、上下文属性和与其它上下文的关系。

·列表上下文内容：列出指定上下文中的服务提交。可以只列出满足某种条件的部分服务提交(指定匹配约束条件)。

(2)输入策略管理操作

·创建输入策略：即定义搜索范围和方法(也可在输入请求时给出该策略)。

·删除输入策略：取消已定义了的输入策略。

·加入 Leads-to 关系：在输入策略中加入源上下文与一组上下文之间的 Leads-to(导至)关系。关于 Leads-to 关系的精确定义及在定义搜索路径中的作用还在讨论中。比如，可用 Leads-to 表示搜索路径中从一个上下文到另一上下文的箭头。

·删除 Leads-to 关系：取消一个 Leads-to 关系。

4.2 交易界面的操作

交易的两个主要阶段为输出和输入，所以交易界面的操作可分为两组：提供给输出者的操作和提供给输入者的操作。

(1)提供给输出者的操作

·输出：将服务提交存入指定(或缺省)的上下文中，在该上下文中为输出的服务赋予唯一的输出标识符。

·取消：将指定的服务提交从其上下文中去掉。

·替代：用新的服务提交取代旧的服务提交，新服务提交具有与旧服务提交相同的输出标识符。这实际上是一个具有原子性的操作序列：(取消(旧服务提交)，输出(新服务提交))。

(2)提供给输入者的操作

·列表：请求返回一指定服务提交的详细内容，包括：服务属性和服务提交属性。

·搜索：请求返回指定上下文(可以是多个)中的一组服务提交，后者满足输入者所要求的服务类型描述和匹配约束条件。输入者可以指定搜索范围和方法(即输入策略)。

·选择：请求从匹配的服务提交中返回"最佳"的一个(按照输入者的选择标准)。

4.3 交易器可获得的操作

交易器可获得的操作主要用于输出策略控制和对服务(界面)类型的操作。如果在服务提交中包含了输出策略控制器标识符，则交易器可进行以下的操作：

·状态查询：请求服务输出者返回指定的动态服务属性的值，如，负载情况。这是一个极有用的操作，对负载平衡及提高效率是非常必要的。

·获取输出策略：当网络拥挤，或原来的服务所在地负载重，甚至出现故障，则需要另一服务来替换。关于出现某种情况时该使用哪个服务界面标识符，便是所指的输出策略。此外，象对输入者的限制也属于输出策略。该操作是请求服务输出者返回其输出策略。

服务是服务类型的实例，而服务类型又由界面类型和服务属性类型构成。有关服务类型及其相应的界面类型的信息由类型库维持着，同时还维持着服务(界面)之间的关系(等价及超、子类型)。为了进行服务请求和服务提交之间的匹配，交易器要与类型库进行交互作用，交易器可使用的操作包括：

·类型检查：检查服务提交和服务请求中类型的合法性。

·兼容性检查：检查两个服务(或界面)类型是否兼容。如果类型 A 与类型 B 等价，或是类型 B 的子类型，则称类型 A 兼容于类型 B。

·列出类型关系：请求给出指定类型的子类型关系。

5 交易器的作用及进一步研究方向

交易器这一新的服务对象管理机制所带来的好处是多方面的，主要表现在：

·实现了服务器的位置透明性。在建立客户应用时可以不必关心服务器所在的位置；而服务器的位置发生变化时也不必修改客户应用，仍可正常运行。

·使系统具备了一定的容错能力。在服务器所在机器发生故障时，可方便地把服务器迁至别的机器上，而无需修改所有的应用程序。

·改善了整个系统的性能。通过获取动态服务属性的值，了解各服务器之负载情况，从而为进行负载平衡提供了条件，有利于改善整个系统的性能。

·使应用系统具有规模可伸缩性。在具有交易器的开放系统中，更新和删除旧的服务(或客户应用)、增加新的服务(或客户应用)都极为方便，不需对应用做任何修改。

交易器的思想所带来的好处是明显的，但仍有许多问题需要解决。如负载平衡问题。当有多个服务满足多个客户的需求时，如何选择服务器来分配任务，以达到服务器的负载平衡是一个比较复杂且亟待解决的问题，它关系到带有交易器的开放系统的性能。另外，服务器的安全访问也需进一步地研究，在当前有关交易器的研究中，涉及安全性的比较少，但要使交易器实用，被交易服务的安全管理必须考虑。

计算机科学1995 Vol. 22 №. 4

主动数据库及其实时应用支持*)

刘云生　胡国玲　卢炎生
（华中理工大学计算机系　武汉430074）

摘　要　This paper analgzes the requirements, presents concepts and proposes an architecture of an active DBMS. And we discuss in detail the model of a trigger mechanism, as well as its structure. For concreteness, we take a prototype system ARTs-I as an example in the description.

关键词　Active database, Trigger, Event, Trigger monitoring, Event detection, Condition evaluation.

一、引　言

传统的DBMS技术对于要求存取大量共享数据和控制知识的应用，显得无能为力。如CIM、过程控制、合作处理、战斗管理、空中交通管制、网络管理等，这类应用的一个共同要求是DBMS能监视关于数据库（或非数据库）的事件和条件，当条件满足时，能自动、适时地作出相应的反应（执行特定的活动）。传统的DBMS是"被动"的，即仅当用户或应用程序显式地请求时，它们才执行事务或查询，要实现上述应用只有两种办法：周期地不断查询和嵌入条件评价与活动调用到应用程序中。这两种办法都不理想，前者除引起"抖动"、浪费大量资源，因而影响整个系统性能和及时性外，更重要的是为了及时地探测和响应要监视的情形，用户必须确定查询的周期，而这依赖于多种参数，其中许多是动态的、彼此关联的，如事件的类型及发生的频率、情形的及时性要求（即情形可能出现的时间间隔）等；后者则不仅转移有关定义事件与条件、表示条件查询、决定条件评价及其时机的负担给应用程序员，且损害软件模块性和应用开发；对事件、条件及活动的任何修改都将要求相关应用程序的修改。

数据库的非传统应用促使了主动数据库（ADB）的研究。主动数据库能提供自动、适时反应和模块性两者。它能自动监视由各种事件所引起且由一定的条件来判别的的情形，当情形出现时，调度相关任务并使其满足定时性和一致性要求，这些都无须用户干预。

我们可以定义一个主动的DBMS（ADBMS）就是一个扩展了下列功能的DBMS：

（1）用户可以显式地定义想要监视的情形（事件与条件）；

*)国家自然科学基金、国防预研资助项目。

主要参考文献

[1] 刘锦德、唐雪飞，关于我国军用计算机开放系统的轮廓描述，电子科技大学微机所内部报告，1992.8

[2] ISO/IEC JTC1/SC21 Draft Recommendation X.901(2,3): Basic Reference Model of Open Distributed Processing-Part1(2,3), 1993

[3] ISO/IEC JTC1/SC21/WGT/N743, Working Document on Topic 9.1-ODP Trader, Nov. 1992

[4] Alex Berson, Client/Server Architecture, McGraw-Hill Inc., 1992

[5] S. Rudkin, Templates, Types and Classes in Open Distributed Processing, BT Technol. J., Vol 11, No 3, 1993

[6] Hewlett-Packard Company, Network Computing System Reference Manual, Prentice Hall Inc., 1990

[7] Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing, IFIP Transactions C-20, 1994