第 7 卷第 3 期
1996 年 3 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 7, No. 3
Mar. 1996

一个基于 MultiServer 系统的分布计算模型*

汲 化 谢 立 孙钟秀

(南京大学计算机系 南京 210093)

摘要 本文针对开放分布式处理的要求,分析了传统分布计算模型 Client/Server 的不足,提出和设计了一个基于 MultiServer 系统的分布计算模型,它具有面向对象、服务静态和动态调用、对多服务关系层次的支持等优点,具有较好的服务透明性、可扩充性和容错性.

关键词 开放分布式处理,MultiServer,服务,服务关系.

开放分布式处理 ODP(open distributed processing)是分布计算发展的方向,它试图解决分布计算环境下软件的接口问题,以达到分布式系统的可移植性、透明性、互操作性.[1]开放分布式处理参考模型 ISO RM_ODP 即将成为国际标准(ISO 10746 和 CCITT X.900 系列)[2],它定义了 ODP 的目标框架,但没有给出其具体实现环节.为了达到上述目标,目前尚存在一些关键问题需要解决[3~5],例如:透明性:分布式系统的一个重要问题是透明性,即使得用来克服分布问题和实现的机制相对用户应用无关,如存取透明性、位置透明性,它是分布式系统开放程度的一个重要标志.动态连接:在分布式系统中,构成系统的部件将由于系统扩张、改良、排障或负载平衡等原因而不断动态地变化,这就要求系统必须能够支持服务的动态连接,即顾客(Client)必须能在运行时刻发现哪一个服务员(Server)可提供所需的服务,它满足 Client 所要求的服务类型和特性,并动态连接到该服务所在的 Server 上.

本文分析了分布计算模型 Client/Server 对 ODP 需求的局限,提出和设计了一个基于 MultiServer 系统的分布计算模型 MSOM,它具有面向对象、服务静态和动态调用、对服务关系,即 MultiServer 的支持和独立于实现细节等优点,具有较好的服务透明性、可扩充性和容错性.

1 MultiServer 系统

随着分布式系统的规模越来越大,网络中存在大量的、各种各样的服务可供利用,同时,对于任一个服务功能利用,已不再是原来的单一的 Client/Server 的关系,分布式系统中存在着若干个完成相同服务(EqualService)、超集服务(SuperService)、子集服务(SubService)

* 本研究得到国家 863 高科技项目基金资助.作者**汲化**,1969 年生,博士,主要研究领域为分布式系统.**谢立**,1942 年生,教授,博士导师,主要研究领域为分布式计算,并行处理.**孙钟秀**,1936 年生,中国科学院院士,教授,博士导师,主要研究领域为分布式系统.

本文通讯联系人:汲化,南京 210093,南京大学计算机系

本文 1995－08－30 收到

功能的 Server(如打印、显示、计算等),即存在着 1:N($N\geqslant 1$)的关系,Client 面对的不再是单一的 Server,而是存在着逻辑关系的 Server 集,我们把这种性质的系统称为 MultiServer 系统(如图 1(a)所示);而在传统的 Client/Server 模型中,从任何一个 Client 的角度来看,Client 面对的都是某个特定的服务 Server,在每次服务获取的过程中,Client 与特定的 Server 之间是一种 1:1 的关系.[6,7]我们把这种性质的系统称为 SingleServer 系统(如图 1(b)所示). MultiServer 系统与基于传统 Client/Server 模型的 SingleServer 系统相比,可以更好地满足 ODP 的要求.下面我们定义 SingleServer 系统和 MultiServer 系统,并据此进行分析.

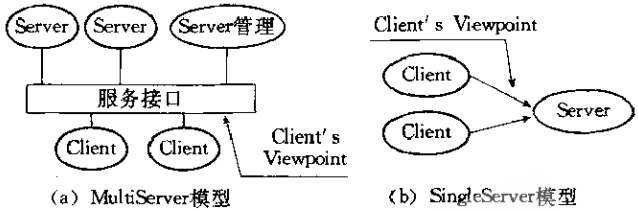

图1

定义 1.1. SingleServer 系统(S 系统):SingleServer 系统表现为传统的 Client/Server 模型,定义为 Client 对象与一个完成所需服务功能的某个特定的 Server 对象的二元关系. S:=〈Client,Server〉,Client 与 Server 的交互采用请求/调用/返回的语义,具体表现为远程过程调用(RPC)的方式.

定义 1.2. MultiServer 系统(M 系统):定义为 Client 对象与一个抽象的 Abs(Server)的二元关系.

$$M:=\langle Client, Ab(Server)\rangle$$

$$Abs(Server):=\langle Set(server), Set(relation)\rangle$$

其中 Abs(server)定义为 Server 的集合与 Server 之间关系集合的二元组,Client 与 Abs(server)之间是请求/调用/返回的 RPC 语义. Server 之间通过服务关系被组织、管理和相互合作.

传统的 Client/Server 模型能有效地支持 SingleServer 系统,但对于 MultiServer 系统,特别是针对 ODP 的要求,传统的 Client/Server 模型则存在不足之处,比较图 1 两个模型,可以得出结论:M 系统可以具有更好的服务透明性,不需要显式地了解某个特定的 Server,如 S 系统,只要系统中存在可满足的服务,即可透明地获得服务;M 系统可以具有更好的可扩展性,与 S 系统相比,能够支持系统平滑地演化,而不影响 Client 的服务获取,并提高服务的利用率;M 系统可以具有更好的容错性,通过服务和服务关系的管理,与 S 系统相比,具有更好的坚定性,Client 不会因为某个特定的 Server 的崩溃而导致服务获取的失败.

2　MSOM 模型的概念及设计思想

下面我们介绍一个支持 MultiServer 系统的分布计算模型 MSOM,首先定义如下概念.

2.1　概念定义

定义 2.1. 服务(Service):在分布式系统中,一个服务(Service)定义为一个抽象意义上的对象(Object)对其私有数据集上的操作集(Operation). 如文件服务、打印服务、计算服务等. 一个服务由其提供的操作集定义该服务的行为(Behaviour). 形式定义为:

$$Service:=Object[\{private:Dataset\}\cup\{public:Operations\}]$$

$$Operations:=[op1\cup op2,\ldots\cup opi\ldots],$$

其中 opi∩opj=NULL,i≥1 在服务概念的基础上,针对 MultiServer 系统的定义,我们引入如下概念:

定义 2.2. 相同服务(EqualService):对于 2 个服务 Service(i),Service(j),相同服务(EqualService)定义为 Service(i)可以用在任何需要 Service(j)的地方,反之 Service(j)也可以用在任何需要 Service(i)的地方,即可以互相替换.若 Service(i)与 Service(j)是相同服务,则存在如下关系,即 {Service(i).Operations⊆Service(j).Operations}∩{Service(j).Operations⊆Service(i).Operations},

Server 之间的相同服务关系是 MultiServer 系统中最常见、最重要的关系,它说明了这 2 个 Server 之间存在可以互相替换的现象.

定义 2.3. 超集服务(SuperService)& 子集服务(SubService):同理可定义对于两个服务 Service(i),Service(j)的超集子集和子集关系,即一个服务 Service(i)可以代替另一个服务 Service(j),但反之不成立.若 Service(i),Service(j)存在超集/子集关系,则满足{Service(i).Operations⊂Service(j).Operations}.

定义 2.4. 服务实例(Service Instance):服务实例定义为在分布式系统中,能够提供某种服务的活动单元,它可以是在生命周期内的进程、线程或更高级的抽象,如 PVM(parallel virtual machine)任务.服务实例通过请求/调用/返回的语义为系统提供服务.在分布式系统中表现为 RPC 调用的方式.每一个服务实例唯一对应一个 RPC 服务程序活动单元;每一个活动单元唯一对应一种服务.形式表达为:Service_Instance:=Instance[Service].

定义 2.5. 服务关系(Service Relation)& 服务相关集(Service Related Set):

服务关系定义为在 MultiServer 系统中,服务(Service)与服务实例(Service Instance)集合间的二元关系.即对于一个服务,存在一个服务实例集;服务实例集中的每个服务实例与此服务存在一个映射关系,它可以是 Equal,Super 和 Sub 等关系.形式地表达为:

{Service_Relation:=[Service,Set[Service_Instance]]}∈{Equal,Super,Sub}

Set(Service_Instance):=[Service_Instance1,...Service_Instancei,...],i≥1

同理,可以定义服务相关集为:对于一个服务,所有存在服务关系的服务实例的集合.即

Service_related:={Service_Instance(1),...Service_Instance(i)},i≥1

(Service_instance(i),Service_Instance(j))∈{Equal,Super,Sub},i,j≥1

定义 2.6. RPC 服务与 RPC 接口(RPC_Signature):RPC 的性质,有别于 Message-passing 方式的仅仅提供数据发送、接收、同步等原语,而是类似于程序中的局部函数(Local Function)调用,具有请求/调用/返回的语义,所不同的是局部函数与调用它的应用程序在同一台机器上,而 RPC 调用一般却是在另一台机器上执行完毕后,再将结果返回调用它的应用程序.RPC 交互模型提供一个高层接口,隐蔽分布的程序间通信的细节.这样分布式程序开发者不必关心网络间复杂的数据传送、接收、同步等问题而获得局部函数调用的语义,从而使得开发变得简单、方便.所以 RPC 是分布式系统中的一个重要中间件(Middleware)[5~7],是分布计算模型的基石.

RPC 接口定义为一个 RPC 过程的函数名,输入参数类型和返回值类型.例如一个 RPC 过程: $F(x1:\sigma1,x2:\sigma2,\cdots,xn:\sigma n)\rightarrow(y1:\rho1,y2:\rho2,\cdots,ym:\rho m)$

则其 RPC_Signature 为: $RPC\_Signature:=(F,\sigma1,\sigma2,\cdots,\sigma n,\rho1,\rho2,\cdots,\rho m)$

MultiServer 系统中每个 RPC_Signature 属于某个服务实例,每个服务实例都包含一个或若干个 RPC_Signature.

定义 2.7. 服务接口(Service Interface):服务接口是一个服务实例(Service Instance)对于外部世界的接口或外部世界可观察到的行为(Behaviour),定义为提供服务的服务实例的标识(Sid)和该服务实例所具有的 RPC 接口信息集的二元组.形式表达为:

$$Service\_Interface::=[Sid,Set(RPC\_Signature]$$

$$Set(RPC\_Signature)::=[RPC\_Signature1,...,RPC\_Signaturei,...]$$

定义 2.8. 服务空间(Service Space):服务空间定义为分布系统中服务接口的集合,它描述了所有可利用的服务资源,即 $Service\_Space::=\bigcup\{Service\_Interface\}$

2.2 MSOM 设计思想

MSOM 的设计目标是:设计一个面向对象的、具有静态调用和动态调用接口的、支持多服务器(MultiServer)的以及独立于实现细节的分布计算模型(参见图 2).下面我们介绍 MSOM 的设计思想.

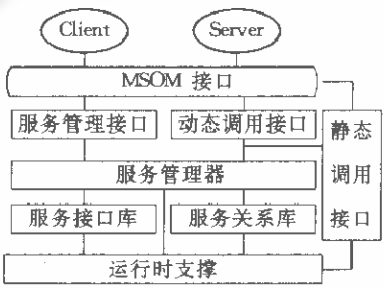

图2 MSOM Core Model

2.2.1 面向对象的服务获取

Client/Server 模型中,一个服务的获取要显示地面对网络中的机器结点(如以太网卡地址)和进程的概念,各个服务实例没有一个全局的,与服务所在节点地址无关的标识 ID,从而没有很好的服务获取透明性.我们采用面向对象(Object Oriented)的方法定义分布式系统中的服务.即每一个服务实例(Service Instance)为一个服务对象(Service Object),一个服务对象由对象名(Object Name)唯一地在服务空间(Service Space)中标识和定位.对象名可以是字符串或一个整数,如 PVM 系统中的任务标识 Tid,因此对服务的获取,不再需要其它附加的信息来标识一个 RPC 服务,如传统的过程号、版本号等.

2.2.2 静态服务获取

静态调用接口的采纳是为了提高服务调用的直观性和效率,即 Client 任务显式地了解 MSOM 中服务实例的 Sid 并获知调用接口,从而与 Server 进行连接(Binding).这种方式是 RPC 机制中普遍采用的方法.

2.2.3 动态服务获取

从上述和对 ISO/OSI RPC 模型分析,我们可以得出结论:目前的 RPC 服务机制大都采用静态连接调用机制,虽然它具有直观、效率高的特点,但同时也具有缺乏灵活性、可移植性、缺乏分布透明性的缺点.[6~9]

因此,在 MSOM 机制中,提供一个动态调用接口是很有意义的,即一个作为 Client 的任务能够在运行中动态查询、发现并调用所希望的 Server.

2.2.4 MultiServer 功能支持

为了支持 MultiServer 模型,MSOM 模型中引入了服务管理器(Service Manager)、服务接口库和服务关系库,对系统中的服务及服务关系进行统一管理.服务接口库保存着分布式系统中的可利用服务信息(Service_Interface),服务关系库保存着分布式系统中的可利用服务关系信息(Service_Relation),服务管理器用以保证 Client 获取服务的动态调用和透明

存取,支持服务相关集的交互行为,以增强分布式系统的可扩展性和容错性.

3 MSOM 参考模型

我们给出 MSOM 的基本参考模型框架(Core Model),定义了其组成的基本部件(Component).采用与具体实现细节无关(Implementation—independent)的方法,允许不同的厂商在保持一致的 API 的基础上,采用不同的实现方案,以保持异构性和互操作性.

MSOM 提供一个一致的 API 接口,以支持分布环境下具有 MultiServer 系统的分布计算模型. API 包含静态调用接口、动态调用接口和服务管理接口.

静态调用接口提供一个基于 Sid 的服务调用 API,如_service_invoke()等.

动态调用接口提供一个动态服务调用 API,可以在运行时刻查询、定位所希望的服务.如. service_import().

服务管理接口提供一些用于服务注册、服务撤销、服务关系注册、服务关系撤销等原语.如_service_export(),_service_relation_add(),_service_relation_delete(),_service_relation_query()等.

服务接口库负责存放系统中可利用的服务信息,即 Service_Interface. 它包含提供该服务的对象名 Sid 和操作方法集(如 RPC 调用过程集).服务接口库中还可以存放一些该服务对象的属性(Attributes),例如:服务质量(Qos)参数、量化的负载信息等等.

服务关系库负责存放系统中已定义的服务关系信息,包含相同、超集/子集等 3 种关系.

服务管理器是一个重要的部件,它负责管理维护 MSOM 服务空间中所有的服务和服务关系,并负责和其它的服务管理器相联系、合作,以提高系统中服务获取的透明性、坚定性.服务管理器主要完成 2 种行为.

(1)接收一个 Client 通过 RPC_Signature 对服务的申请,服务管理器依据 Client 提供的 RPC_Signature,查询服务接口库,获得能满足服务要求的 Server 标识 Sid,并返回给 Client,从而 Client 利用 Sid 动态地定位和激活相应的服务.

(2)当 Client 申请一个服务调用 Service(i)时,由于负载平衡问题,网络超时或结点崩溃而返回失败时,服务管理器利用查询原语对服务关系库操作,寻找可替换的服务对象,并透明地完成服务操作.其行为可以用类代码表示如下:

```
    Client.request[Service(i)];
If (wait(result)! =True) then
  {for j=1 to all relatedto(Service(i))
      {
        If {{{Service(j) is EqualService of Service(i)}∪
          {Service(j) is SuperService of Service(i)}}∩
          {Some other attributes features, eg, load information}}
      then Client.request[Service(j)];
      If(wait(result) :=True) then return(result);
      }
}
```

服务管理器的行为(Behaviour)对于 Client 任务而言是完全透明的.这种服务调用语义

可以提高分布系统中服务的利用率，增强服务获取的透明性，提高分布系统的容错性.

运行时支撑部件由服务监控任务(Daemon)和一套低层的运行时刻库组成. 它们负责网络间 Client 与 Server 之间操作命令的交互，服务语义的选择和数据的传送，不同的数据格式转换，如 XDR 协议等. 运行时支撑部件可以采用不同的厂商不同的技术实现，如 DCE/RPC 或 SUN/RPC 等.

限于篇幅关系，本文不再详述 MSOM 提供的 API，具体的细节将另文给出.

4　总结与讨论

我们依据 MSOM 参考模型，在 SUN Sparc2 和 Wyse series 7000i(4 CPUs)硬件环境下，SUNOS4. 1. 2，SCO UNIX for MPX 和 PVM3. 37 软件环境下，实现了一个支持 MultiServer 的原型系统，下一步的工作是进一步完善原型系统，并对 MultiServer 系统做更进一步的理论探索和实践.

参考文献

1　Basic reference model & open distributed processing. ISO/IEC JTC/SC21, Part1～4, 1993.

2　Linington P F. Introduction to open distributed processing basic reference model. International IFIP Workshop on Open Distributed Processing, Berlin, Oct. 1991.

3　Herbert A. The challenge of ODP. International IFIP Workshop on Open Distributed Processing, Berlin, Oct. 1991.

4　Bearman M, Raymond K. Federating traders: an ODP adventure. International IFIP Workshop on Open Distributed Processing, Berlin, Oct. 1991.

5　Object Management Architecture Guide 2. 0. OMG TC Document, 1992.

6　Liu Yusheng, Hong D B. OSI RPC model and protocol. Computer Communication, 1994, **17**(1): 32～38.

7　Sun Microsystem. Networking on the Sun Workstation.

8　汲化，谢立，孙钟秀. 一个支持开放分布式处理的 DRPC 模型. 软件学报，1996，**7**(增刊)：73～77.

9　汲化，吴迎红，周笑波等. 一个支持服务动态连接的 RPC 模型的设计与实现. 95' 全国开放系统会议论文集，1995.

A MULTISERVER ORIENTED DISTRIBUTED COMPUTING MODEL

Ji Hua　Xie Li　Sun Zhongxiu

(*Department of Computer Science　Nanjing University　Nanjing*　210093)

Abstract　This paper presents some of the existing problems for ODP (open distributed processing), analyzes the traditional client/server model in the view of ODP's goal. A MultiServer oriented distributed computing model MSOM is proposed, through which the authors can get better distribution transparency, scalebility and fault tolerance.

Key words　Open distributed processing, MultiServer, service, service relation.