文章编号:1001 - 9081(2006)02 - 0282 - 03

基于 Chord 协议的网格服务管理方法

郑明华,李　青
(上海大学 计算机工程与科学学院 ,上海 200072)
(Zmh5566@126.com)

摘　要:介绍一种利用 Chord 的 P2P 协议来管理网格服务的方法,可以克服当前通常采用 UDDI 来负责网格服务的管理而导致容易造成性能瓶颈以及单点故障的难题。实验证明该办法可方便快速、高效地部署、发布以及查找网格服务。

关键词:网格;服务;Chord;对等网络
中图分类号: TP393　　**文献标识码**:A

Grid service management method based on Chord protocol

ZHENG Ming-hua, LI Qing
(*College of Computer Engineering & Science, Shanghai University, Shanghai* 200072, *China*)

Abstract: The method which uses a kind of P2P protocol named Chord to manage grid service was introduced. The method may conquer problems such as single point of failure and performance bottle-neck resulted from adopting UDDI to supervise grid service. It is testified by trial that the method can deploy, publish and search grid services rapidly and efficiently.

Key words: grid; service; Chord; P2P

目前,在网格中,当我们需要请求网格服务时,首先联系通用描述、发现和集成(Universal Description, Discovery, and Integration,UDDI)注册中心,获得所要求的服务的 GSR(Grid Service Reference)[1],然后才能调用相应的服务。但是当用户很多的时候,UDDI 中心容易形成性能瓶颈,甚至可能发生单点故障问题。因此本文提出一种基于 P2P 分布式查找协议即 Chord 协议的网格服务管理方法。

1　Chord 协议

Chord 是一种分布式查找协议,即使在网络中节点很多的时候,它可以通过少数几步的路由(在 Chord 中称为 hop)来找到所需要的相关服务或结点。Chord 实现的主要功能是:给定一个 Key,根据该 Key 映射到某个结点(node)。在 Chord 协议中,给每个描述数据项的关键字、主机赋予一个 m 位(二进制位)的标识符(ID),此标识符可以通过 hash 关键字产生。

Chord 协议采用 SHA-1[2] 作为相容函数(consistent hash)来给每个结点和每一关键字赋 m - 位的标识符。哈希函数通过哈希关键字来计算该关键字的标识符,结点标识符是通过哈希 IP 地址得到的。关键字和节点标识符组成一个大小为 2^m 的一个环(见图 1)。

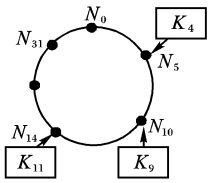

图 1　系统的 Chord 环

这个环就是 Chord 环,Chord 环上的标识符从 0 到 $2^m - 1$。关键字存储到一个结点,当且仅当这个结点的标识符大于或等于该关键字的标识符,这一结点就叫做该关键字的后继,用 successor(key)表示。当 m 足够大的时候,不同的关键字或者结点 IP 哈希到同一个标识符的可能性就可以忽略不计[3]。如图 1 所示,关键字 K_4 存储在节点 N_5,K_9 存储到 N_{10}, …。Chord 有一个叫做 Finger table 的路由表,它保存不超过 m 项纪录,每一项指向一个结点。关于路由表的具体定义见参考文献[3]。定位 ID 所在的结点时,先查找本机的路由表,找出大于或等于 ID 的第一个记录所对应的结点,再到这一结点查找,直到找到 ID 的后继为止。

2　基于 Chord 协议的网格服务管理

为了提高服务定位的速度,我们对 Chord 的路由表进行扩展,加入了 service repository 表,该表维护了 m(在本系统中等于 160)位 ID 与网格服务注册信息之间的映射。扩展后的路由表如表 1 所示。

表 1　service repository **定义**

符号	说明
ID	结点或关键字的标识符
IP	ID 对应的结点的 IP 地址,假如是网格服务,就是服务宿主的结点 IP
Info	在地址为 IP 的结点中服务关键字标识符等于 ID 的服务的描述(目前是 XML 文件)

路由表中还有其他的字段,与 Chord 的 Finger 表[3] 相同,在此不再赘述。

在系统中,将所有网格服务关键字标识符和结点标识符组成一个大小为 2 的 m 次方的 Chord 环,其中,为了系统的可扩展性以及减少两个结点或两个关键字经过 hash 后出现相同标识符(ID)的可能性, m 的值要取尽量大,在本系统中取 $m = 160$。对于每个结点,将其 IP 地址直接哈希为一个关键字。如图 1 所示,该图是一个 $m = 5$ 的 Chord 环。IP 地址为

收稿日期:2005 - 08 - 03;**修订日期**:2005 - 10 - 27　　**基金项目**:国家自然科学基金资助项目(60373071)
作者简介:郑明华(1973 -),男,海南儋州人,硕士研究生,主要研究方向:计算机网格、分布式计算;　李青(1962 -),男,湖北嘉鱼人,教授,博士生导师,主要研究方向:高性能计算.

192.168.1.1 的节点 node 经过 SHA_1 哈希后为 11,那么它的后继结点就是 N_{14};根据 Chord 协议,node 的 ID、IP、Info 等信息就会保存到结点 N_{14} 的路由表(主要是指 service repository)中。而网格服务可以用元数据描述来当作关键字(Key),如元数据描述为"serviceName = test"的网格服务经过哈希后得 4;同样地,ID 4 与该网格服务所在结点 IP 以及服务的 Info 等信息就要保存到结点 N_5。经过这样的处理后,网格系统中提供网格服务的结点组成一个 Chord 环。

2.1　结点的关键组件

每一个网格服务发布节点除了安装网格容器外,还需安装一个我们项目组开发的容器插件 Container_Plugin(如图 2 所示)。

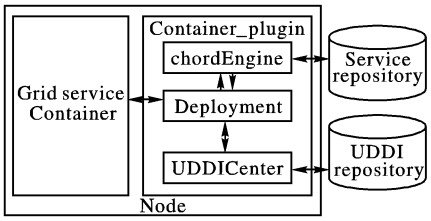

图 2　结点关键组件

它由 ChordEngine、UDDICenter 和 Deployment 三个模块以及 service repository 表和 UDDI repository 库组成,三个模块分别负责三个方面的工作:

1) ChordEngine,实现 Chord 协议,安装有该引擎的机器结点就是一个对等结点(peer)。ChordEngine 维护一个路由表(包括 service repository 表),这个路由表有 160 个路由项,每一项指向一个结点。当某个结点加入 Chord 环时,需要通过现在 Chord 环中的其他结点来初始化其路由表;另外该引擎还负责与其他的结点通讯,实现网格服务的发布、部署和查找功能。

2) UDDICenter,根据网格服务描述语言文件(WSDL),注册网格服务。

3) Deployment,将 Grid Service 部署到网格服务容器。Deployment 调用 chordEngine,查找网格服务(或结点)的后继并且把网格服务(或结点)的相关信息保存到它们的后继结点当中去。

表 Service repository,负责本地以及从远程路由过来的网格服务所需要发布的信息(信息的字段如表 1 所示)。库 UDDI repository 负责本地的网格服务的注册。

2.2　WSDL 文件的扩展

为了更好的利用现有的技术,我们对 WSDL(Web 服务描述语言)[1] 进行扩展,在网格服务的 WSDL 文件中加入如下格式的内容,这些内容写入 service repository,提高网格服务定位速度。

```
<Chord>
    <ID> 由哈希关键字后得到的标识符 </ID>
    <Info> 发布网格服务的描述 </Info>
    <IP> 网格服务的宿主结点 IP 地址 </IP>
</Chord>
```

把扩展后的 WSDL 文件叫做网格服务的 XWSDL 文件(发布文件),当用户提交该文件到 Container_Plugin 后,先把上面的这一段描述从 XWSDL 文件读取出来,再根据发布文件中其余的内容生成标准的 WSDL 文件。调用 Deployment 模块把这一个网格服务注册到 UDDI。ChordEngine 模块把上面的这段描述发送到该网格服务的后继结点,某个后继结点收到后,存入其 service repository。

2.3　服务部署、发布和查找

用户准备部署网格服务时,不是直接部署到网格服务容器,并注册到网格系统的 UDDI 中心。而是用户直接向 Container_Plugin 发送请求,提交其网格服务的发布文件;Container_Plugin 接收到该请求后,获取发布文件指定的关键字,假如只有一个关键字,就直接生成其 ID;假如是一个关键字集合,那么就计算出这一集合的子集,当然不包括空集,集合中的元素用"空格"格开,生成每一子集的 ID。然后调用 Chord 路由算法,查找这些 ID 的后继结点 successor(ID),将该服务的发布文件发送到 successor(ID),即 ID 的后继结点。后继结点将相应的信息(ID、IP 地址、服务描述元数据信息)写入其 service repository。

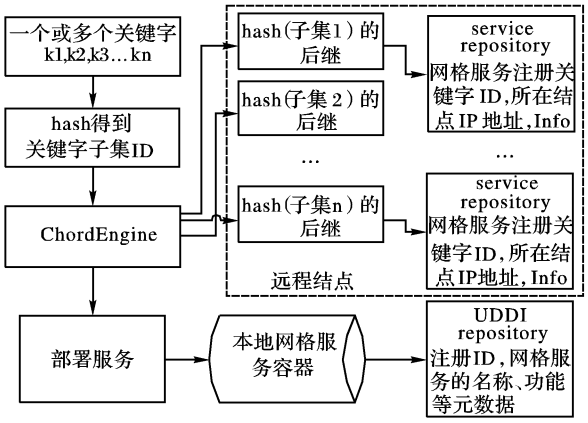

图 3　服务部署、发布时各个组件之间的信息流

另外发布网格服务的本地结点也要将该网格服务的发布信息保存到其 service repositoty,并且调用 Deployment 将服务部署到容器里。执行完这些步骤后,将该服务注册到 P2P 的 UDDI 中心。

2.3.1　网格服务的部署

与传统网格系统(如 Globus、欧洲数据网格等)不同,在我们的系统中,没有集中的服务注册中心 UDDI,每一个服务发布结点都有自己的 UDDI,负责部署本地网格服务。主要注册网格服务的名称、功能、调用方式、调用参数、返回结果和其他描述信息,也就是服务的元数据。

2.3.2　网格服务的发布与查找

Service repository 负责保存网格服务发布的信息,该服务可能是本地用户也可能是由远程服务器经过路由算法传递过来的。设网格服务的关键字为 key,当 successor(hash(key)) = hash(本机 IP),则该网格服务就发布在本地节点,发布信息同时写入 service repository。否则,经过 Chord 算法查找后继节点并发布到后继节点上。

本系统在发布服务前,先根据用户提供的网格服务的关键字(Key),哈希得到一个标识符(ID),把得到的 ID、本结点的 IP 地址、服务的 WSDL 文件组成一个新的发布文件(我们把这个文件叫做 WSDL 文件的一个扩展,前文已有描述)。根据 Chord 的路由算法,找到 ID 的后继节点将发布文件发送到该后继节点,完成服务的发布过程。

发布服务时,用户需要提交其服务的发布文件到 Container_Plugin, Container_Plugin 接受到请求后,提取发布文件的关键字。关键字可能是一个,也可能是多个,在生成 ID 时,本文采用关键字的一个任意子集,这样有利于服务的查找。比如有 k_1,k_2,k_3 三个关键字,那么可以生成的关键字为:$SHA\_1(k_1)$,

$SHA\_1(k_2)$,$SHA\_2(k_3)$,$SHA\_1(k_1\ \ k_2)$,$SHA\_1(k_1\ \ k_3)$,…,$SHA\_1(k_1\ k_2\ k_3)$ 共 7 个 ID,其中每个关键字之间是一个空格,并且按照这 7 个 ID,将网格服务发布到 7 个不同的结点上。假如其中两个或多个 ID 的后继结点是同一个,那么服务只是发布一次,见图 3。

这样处理后,可以提高服务的可找性以及快速、精确地找到所取的服务。再以上面地例子来说明,服务 α 的关键字是 k_1,k_2,k_3;服务 β 的关键字是 k_4,k_6,k_1。那么,通过 k_1 就可以查找到两个服务:α,β。为了精确的找到所需要的服务 α,可以输入 k_1 k_2 关键子组合,或者 k_1 k_2 k_3 等等。

2.4 服务的查找路由算法

用户或应用提交查询请求,那么执行以下的路由算法来获取服务的 GSR。

1) 从查询请求中提取服务的关键字 Key 集合,并哈希得到每个子集的标识符 ID。

2) 查找本地 service repository 表,如果发现其中的一个子集的 ID,则返回结果,查询结束。否则转 3) 。

3) 对每个子集的 ID,继续查找本地路由表(fingerTable)大于等于这些 ID 的所有标识符,记为 $ID_1 \cdots ID_n$,其中 n 等于子集的个数,将请求同时发送到 $ID_1 \cdots ID_n$ 所表示的节点。如果在这些结点中找到网格服务 ID 的后继,则返回结果,否则继续 2) 的查找,直到查找到或者失败。

如图 4 所示,黑圈表示计算机结点,白圈表示没有结点,或者是网格关键字经过 hash 后的标识符。在一个 m = 3 的 Chord 网络里,假如用户在 ID 为 0 的结点发出请求,要查找 ID 为 2 的结点。结点 0 的 service repository 里没有 ID 为 2 的网格服务,需要进一步查找后继结点。因为 ID 2 落在路由表的第二项里,所以,得出后继结点为 3;现在到了结点 3,在结点 3 的 service repository 里查到 ID 为 2 的服务,查找成功,返回 IP 和 Info 里的对应信息。该算法是针对本项目的需要,结合 Chord 的特点,加入并行计算功能的路由算法,与 Chord 协议原有的路由算法是有区别的。经过这样的修改后,可以更加容易的查找到发布的网格服务,缺点是增加了网络的信息流量。

须指出的是,由于网格的动态、虚拟、共享特征,结点的状态可能随时出现变化,如有结点加入、退出或结点出现故障等。当系统出现这些问题时,将会影响到系统的性能,即使网格服务存在于某个结点也有可能找不到所要的这个网格服务。对于这些可能出现的问题,可以采用 Chord 协议稳定动能(stabilize)来解决[3]。

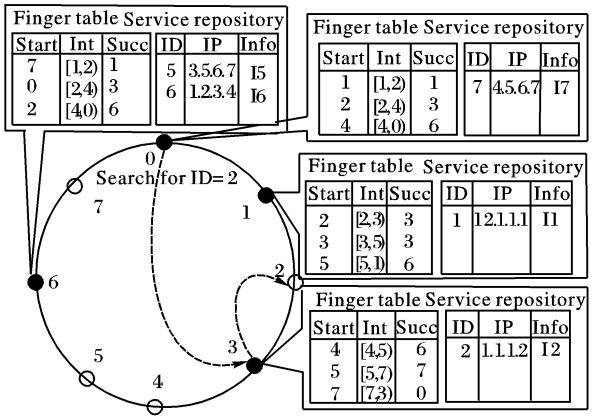

图 4 网格服务查找

3 结语

采用了 Chord 协议后,网格系统中可快速高效查找所需要的服务,减少了集中式 UDDI 所带来的性能瓶颈。下一步的工作主要是对网格服务容器做进一步的扩展,将现在的 Container_Plugin 的功能与服务容器集成在一起。这样,可以简化服务发布。

参考文献:

[1] FOSTER I, KESSELMAN C. The Anatomy of the Grid[J/OL]. U. S. Supercomputer Applications, 2001.

[2] Department of Commerce/ National Institute of Standards and Technology. Secure Hash Standard[J/OL]. U. S. FIPS, 1995, 180.

[3] STOICA I, MORRIS R, KARGER D, *et al*. Chord: A Scalable Peertopeer Lookup Service for Internet Applications[J/OL]. San Diego, California, USA: SIGCOMM, 2001 - 08

(上接第 281 页)

这也解决了与此相关的另一个问题:当两个通信实体都是移动终端,而且在同一时间发生移动并变换 IP 地址,若只是采用原方案,这时双方无法把自己的更新状态信息及时通知对方,从而导致通信中断。而采用 RVS 方案可以很好地解决这个问题,因为 RVS 可以转发其他用户告之的状态更新信息。如图 8 所示。

3) 可靠性问题。若移动终端的更新报文被丢失,则会导致其他方发起向移动终端的通信业务失败。这可以通过设定记录的生存期(Lifetime)以使网络定期主动发起要求终端更新注册。若记录的生存期未超时,则可以在移动终端最后注册的位置区及相邻位置区进行多向寻呼(omnipaging),即同一位置区的异构网络相互协同寻呼。

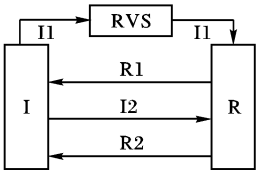

图 8 利用 Rendezvous Server 交换 HIP

综上所述,本论文所提出的移动性管理方案对于在异质网络间移动通信具有灵活性和普适性。而且基于 HIP 协议可以保证通信的简单、高效和安全。

参考文献:

[1] MOSKOWITZ R, NIKANDER P. Host Identity Protocol Architecture (IETF Internet-Draft: draft-ietf-hip-arch-02) [EB/OL]. http://www.ieee802.org/21/, 2004 - 01 - 11.

[2] MOSKOWITZ R, *et al*. Host Identity Protocol(IETF Internet-Draft: draft-ietf-hip-base-02) [EB/OL]. http://www.ieee802.org/21/, 2005 - 02 - 21.

[3] NIKANDER P, LAGANIER J. Host Identity Protocol (HIP) Domain Name System (DNS) Extensions(IETF Internet-Draft: draft-ietf-hip-dns-01) [EB/OL]. http://www.ieee802.org/21/, 2005 - 02 - 20.

[4] NIKANDER P, *et al*. End-Host Mobility and Multi-Homing with the Host Identity Protocol (IETF Internet-Draft: draft-ietf-hip-mm-01) [EB/OL]. http://www.ieee802.org/21/, 2005 - 02 - 20.

[5] LAGANIER J, EGGERT L. Host Identity Protocol (HIP) Rendezvous Extension (IETF Internet-Draft: draft-ietf-hip-rvs-01) [EB/OL]. http://www.ieee802.org/21/, 2005 - 02 - 18.