第 37 卷 第 24 期
Vol.37 No.24

计 算 机 工 程
Computer Engineering

2011 年 12 月
December 2011

·软件技术与数据库·　文章编号：1000—3428(2011)24—0038—04　文献标识码：A　中图分类号：TP311

跨域服务注册中心的数据同步研究

伍　枫，谷建华，赵天海

(西北工业大学高性能计算研究与发展中心，西安 710072)

摘　要：为减少 Web 服务注册中心的用户响应时间，根据 UDDIV3 规范在分布式环境下建立跨域服务注册中心系统。将系统分为域间和域内两层，域间使用基于订阅的方式进行数据异步复制，域内使用动态主从复制方式进行数据同步复制，并优化 UDDI 的数据结构。分析优化前后的数据量，对该系统和节点间完全复制的系统进行比较。实验结果表明，该系统能够有效地提高服务注册中心的性能。

关键词：Web 服务；统一描述、发现和集成规范；跨域服务；数据同步

Research on Data Synchronization for Cross-domain Service Registry Center

WU Feng, GU Jian-hua, ZHAO Tian-hai

(Research and Develop Center of High Performance Computing, Northwestern Polytechnical University, Xi'an 710072, China)

【Abstract】In order to reduce the user response time of Web service registry center, on the basis of the characteristics of Universal Description Discovery and Integration(UDDI) standard, this paper divides the system into two layer of inter-domain and internal-domain, and establishes cross-domain system. In this system, inter-domain uses subscription-based approach for asynchronous data replication, internal-domain uses dynamic master-slave replication for synchronization data replication. It optimizes the UDDI data structure. The amount of data before and after optimization is analyzed. The system and complete duplicate systems between nodes are compared. Experimental results show that the system can effectively improve the performance of the service registration center.

【Key words】Web service; Universal Description Discovery and Integration(UDDI) standard; cross-domain service; data synchronization

DOI: 10.3969/j.issn.1000-3428.2011.24.013

1　概述

Web 服务注册中心根据统一描述、发现和集成(Universal Description Discovery and Integration, UDDI)协议建立，目标是对 Web 服务进行统一的描述、发现、集成，需要服务注册中心必须具有高吞吐量、低响应时间、高可靠性和高准确性。而 UDDIV3 规范设计的主要思路是面向以后全球统一的 UDDI 架构的，是一种大型的分布式系统[1]。每个 UDDI 节点之间不需要完全的数据复制，需要将整个系统环境根据地理位置、经济状况、客户和商业群的不同分成多个域，域内需要建立多个节点以提高系统的可扩展性。因此，域服务注册中心的节点间数据同步问题是研究的关键。

目前的数据复制方法主要分为同步复制和异步复制：(1)同步复制要求事务必须等到与此相关的所有备份都完成操作后才能提交，保证了在任何时刻数据库整体的一致性。同步复制方法包括 2PC[2](两段提交协议)、3PC[2](两段提交协议)、主从拷贝复制[3]等。同步复制虽然保证数据的实时性，但也带来易死锁、通信量大、节点规模受限制及事务响应时间较长等问题。(2)异步复制可以在事务提交后再将数据的修改传播到其他节点上，即某个数据被更新后并不马上更新所有的结点备份，因此，允许不同的节点包含不同的数据，这样大大降低了对网络的要求。异步复制方法包括虚拟日志法[4]、消息队列法[5]等。但异步复制存在数据不一致的时间段可能有潜在的数据冲突。

本文在分布式环境下提出跨域的两层系统结构，分别使用数据同步复制和异步复制方式，同时优化了 UDDI 的数据结构，从而实现整个系统的数据同步。

2　系统拓扑结构

本文采用符合 UDDIV3 规范的面向全球架构的分域系统拓扑结构，传统的分布式系统中需要通过各种方法保证所有节点保持数据一致，但往往一个域中的用户由于各种原因很少或几乎不会用到域外的服务，其原因如下：

(1)域外的服务由于地理位置相对较远导致网络传输速度慢或经常发生错误，而域内的服务速度较快。

(2)用户一般对域内的注册商比较认可，而注册商之间通过竞争将服务放到用户需求量较大的域中。

(3)访问域外的服务时可能会由于安全和用户权限的问题使访问变得非常复杂[6]。

跨域服务注册中心整体框架如图 1 所示，整个系统分为 2 层，第 1 层代表系统由多个域组成，第 2 层由域内多个 UDDI 节点组成，每个域内有一个主节点和多个普通节点，主节点数据库分为域内和域外 2 个部分，而普通节点的数据库只包含域内信息。用户的数据查询请求基于一个原则，即将查询等级分为域内和域外，在多数情况下用户只需要在域内进行查询就可以得到理想的结果。在系统的第 1 层中，因为用户对域外的服务信息的实时性和准确性要求不高，所以只需每个域设定 1 个头节点负责与其他域的头节点采用域间

基金项目：国家“863”计划基金资助项目“面向应用可定制的跨域协同服务支撑平台”(2009AA01Z142)

作者简介：伍　枫(1985－)，男，硕士研究生，主研方向：分布式计算；谷建华，教授、博士生导师；赵天海，讲师

收稿日期：2011-06-20　**E-mail：**wufeng888@126.com

异步数据复制并将域外的信息写入数据库的域外部分；在系统的第 2 层中，因为用户需要实时查询域内的服务信息，所以域内所有节点之间采用域内同步数据复制，域内的普通节点将数据写入数据库，而头节点将数据写入数据库的域内部分。在整个系统结构中，头节点更新所有域的数据，普通节点只需要更新本域内的信息，节省了存储空间，减少了网络通信量。

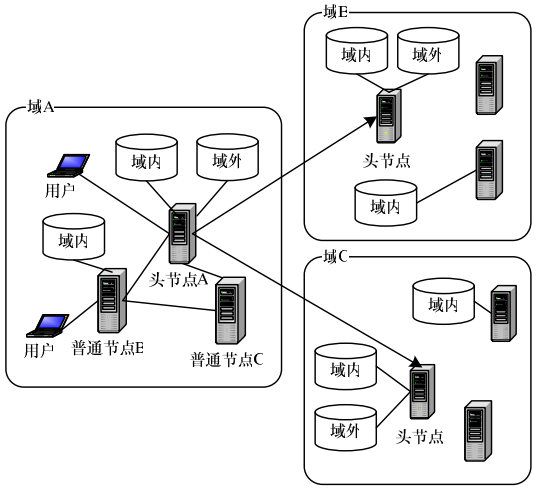

图 1　跨域服务注册中心整体框架

当有用户发出请求时整个系统的运行情况如下：(1)用户对头节点 A 进行数据查询请求，首先在域内的数据库进行查询，随后可以自主选择是否在域外的数据库中进行查询。(2)用户对头节点 A 进行数据更新请求，同时进行域内数据同步和域间数据同步。(3)用户对普通节点 B 进行数据查询请求，首先对本节点的数据库进行查询，随后可以自主选择是否发送查询信息给该域内的头节点 A 并在其域外的数据库中查询。(4)用户对普通节点 B 进行数据更新请求，首先进行域内数据同步，再通过头节点 A 进行域间数据同步。

3　域间数据同步

3.1　数据同步方法

域间采用订阅方式进行数据异步复制，即每个头节点订阅外域头节点数据库中域内部分的数据信息，当某个头节点内有信息变化时，将变化的结果通知其他头节点并使其他节点进行相应的更新。

这需要首先为每个 UDDI 实体对象增加 modified 和 modifiedIncludingChildren 属性，代表在 UDDI 中的主要实体对象包括 businessEntity、businessService、bindingTemplate 和 tModel 的更新时间。同时为每个头节点建立一个包含 UDDI 中主要实体对象相应 Key 值的订阅对象和一个监听服务。随后每个头节点创建一个线程，定时查询订阅对象中的 Key，得到订阅结果对象(实体更新时间在上次通知结束时间和本次通知开始时间之间的实体对象集合，即表示在这段时间内这些实体对象发生变化)，最后访问外域头节点的监听服务，通知其他域内的头节点进行数据更新。

3.2　数据同步流程

域间数据同步的流程如图 2 所示，具体如下：(1)一个域内的头节点定时获得本节点的订阅结果对象。(2)如果发现订阅结果对象中包含数据，则将订阅结果对象封装成 SOAP[3](简单对象访问协议)消息。(3)通知域外所有的头节点并发送 SOAP 请求。(4)如果头节点得到外域所有头节点的确认消息，则更新完成。(5)如果域外节点由于某种原因没有返回消息，则过一段时间后再次向此节点发送 SOAP 请求。

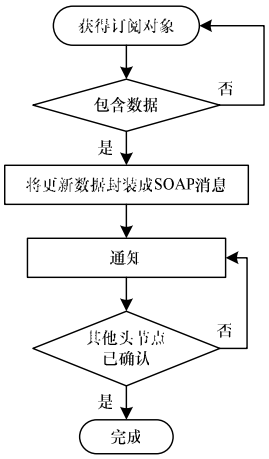

图 2　域间数据复制流程

3.3　数据同步特点

因为用户很少或几乎不会用到外域的服务信息，所以域间的数据传输不需要很高的实时性，采用订阅的数据异步复制方式提高了每个域的独立性，降低了域间网络性能的要求。另外，域间只在头节点进行数据复制，减少了整个系统的网络通信量。

4　域内数据同步

4.1　数据同步方法

首先规定域内进行数据同步复制时，接收用户请求的节点为协调者，其他需要与协调者进行数据同步的节点为参与者。主从拷贝技术首先在域内随机设定一个主节点，使每一个需要进行数据更新的协调者首先与主节点进行通信，再由主节点将信息广播到所有的参与者。这种方法保证了域内数据复制的同步性。另外，可能产生的数据冲突问题集中到主节点上发生，避免了协调者和所有参与者之间因冲突而产生的大量网络通信量，降低用户的响应时间。但这种技术的主要问题是可扩展性差，当主节点出现失效时，其他节点需要通过选举产生新的主节点。本文提出一种动态主从拷贝技术，当域内的主节点失效时，可以将负载较低的 UDDI 节点选举为新的主节点。

4.1.1　节点优先级的设定

每个节点在数据库中建立一张权值表，表中记录了域内所有节点的 IP 地址和权值 Value。Value 越小，节点优先级越高，在主节点失效的情况下，则其他节点选举优先级最高的节点为新的主节点。

4.1.2　权值的获得

首先为 UDDI 中的实体对象增加 IP 和 Value 2 个属性。当用户向协调者发送数据请求后，协调者根据用户的请求从数据库获得相应数据，并从权值表中获得本地的 IP 和 Value 值并封装进 SOAP 请求消息中。如果此次请求在主节点上发生事务冲突，则主节点需要将请求对应 IP 的 Value 加 1，再将此信息发送至域内所有节点并更新权值表的信息。当主节点失效时，其他节点可以根据权值表中的 IP 和 Value 选举优先级最高的节点为新的主节点。当权值表中 2 个 IP 对应的 Value 相同时，将权值表中 IP 值转换成十进制数，最终将 IP 值最大的节点设为主节点。

4.2　数据同步流程

假设域内有一个主节点，当前有两个用户同时向 2 个域内的协调者发出数据更新请求，主节点对冲突事务利用先到

先执行的方式解决。则此次跨域服务注册中心的域内数据同步流程如图 3 所示，具体如下：

(1)2 个协调者分别将更新数据封装成 SOAP 请求发送至主节点。

(2)主节点对 2 个协调者的请求事务进行冲突检测，如果发生冲突且协调者 1 的事务先到，则协调者 2 的事务进入等待状态。

(3)主节点将协调者 1 的 SOAP 请求中对应协调者 1 的 Value 值加 1 并将 SOAP 请求进行广播。

(4)每一个参与者(此时协调者也是参与者)在提交事务的同时将日志写入数据库(将更新后的 Value 值也写入数据库中的权值表)。

(5)所有参与者完成数据复制并对主节点发送确认消息。

(6)主节点向本次数据的请求者协调者 1 发送返回消息。

(7)协调者 1 完成用户请求。

(8)由于协调者 1 的事务已经完成，根据冲突事务的顺序将协调者 2 的 SOAP 请求中对应协调者 2 的 Value 值加 1 并将 SOAP 请求进行广播。

(9)每一个参与者在提交事务的同时将日志写入数据库。

(10)所有参与者完成数据复制，对主节点发送确认消息。

(11)主节点向本次数据的请求者协调者 2 发送返回消息。

(12)协调者 2 完成用户请求。

最终协调者 1 和协调者 2 的数据更新请求按先来先出的事务处理原则提交完毕，而所有节点的权值表中的协调者 1 和协调者 2 的 Value 值加 1。

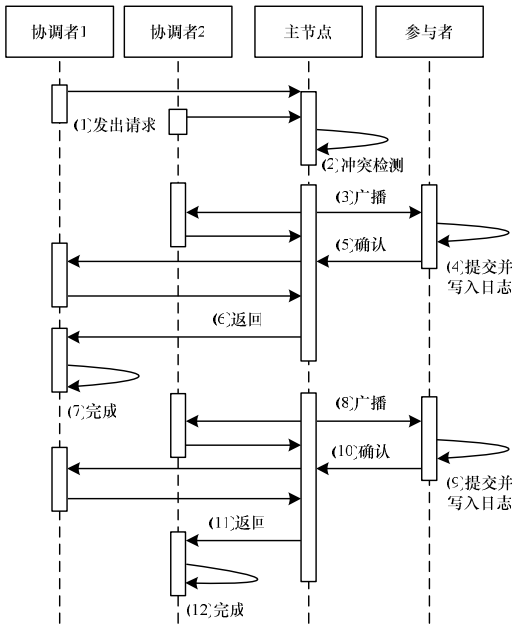

图 3 域内数据同步流程

4.3 数据同步特点

因为不同协调者发送到主节点的请求可能会发生冲突，从概率的角度上讲，发生冲突越少的协调者说明用户发送到此协调者上的请求越少，进而得知此协调者的负载相对较低。所以，优先级越高的节点相对负载越低。这样负载低的节点会优先替换失效的主节点，提高了系统中各个节点资源的利用率。另外，此方法需要根据数据冲突的情况实时得对所有数据库中的权值表中的数据进行更新，每个节点也需要增加存储开销，但由于 IP 和 Value 的数据量很小，对整个系统影响不大。

5 数据优化

在本文系统中，数据查询和更新请求通过 SOAP 消息的方式进行发送，需要将 UDDI 规定的完整数据结构封装成 SOAP 消息。本文提出一个简化的数据结构，减少了 SOAP 消息中元数据信息，降低了数据同步中的数据通信量。

在 UDDI 中定义了 businessEntity、businessService、bindingTemplate 和 tModel 等 4 个主要的数据结构完整的表达了 Web 服务在 UDDI 中注册的基本信息。然而完整的数据是比较复杂的，包含多层的子数据结构。因此，本文系统继续对对象的数据结构进行优化，将标准数据对象转化为数据传输对象 ReplicationMessage。可以看到 ReplicationMessage 对象主要有 3 层，第 1 层主要标示了此次数据请求的数据结构列表，第 2 层包括每一种数据结构的 Key 值和 Property，分别标示具体更新的对象及其属性列表，第 3 层标示了对象的属性名字和值。

图 4 为一个服务注册中心的数据层次结构。当有用户发出数据请求时，服务器会从数据库中获得数据，通过持久层对象，最终产生 UDDI 标准数据对象。本系统进一步将 UDDI 标准数据对象转化为数据传输对象 ReplicationMessage，最后将 ReplicationMessage 封装成 SOAP 消息进行发送。

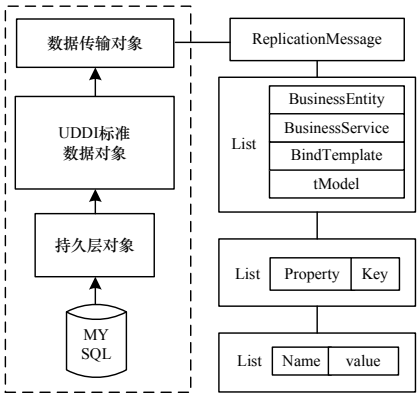

图 4 域服务注册中心数据结构

6 实验结果与分析

本文实验的测试环境如下：WinXP SP2, Inter core2 3.0 GHz, JRE 1.5.0_27, Apache-Tomcat-5.5.27, Mysql5.1, Apache Juddi 3.0.2。

(1)实验 1 使用 2 台 PC 机模拟 2 个 UDDI 节点，当一个用户向一个节点发送请求时，这个节点会将更新信息封装成 SOAP 消息发送到另一个节点。利用 TcpTrace 工具对 SOAP 消息进行截获并可以得到这次 SOAP 请求发送的数据量，如表 1 所示。其中，N 代表用户数据请求数(从对象的角度看就是请求或者更新的不同 UDDI 实体对象属性个数)，优化前表示按 UDDI 规范的数据结构组成的 SOAP 消息的数据量，优化后表示利用本文中介绍的方法组成的 SOAP 消息的数据量。

表 1 数据优化前后数据量比较

请求数 N	优化前/Byte	优化后/Byte
1	382 204	381 420
5	385 802	383 845
10	391 455	387 709
20	400 124	393 902

因为 SOAP 消息包括命名空间、SOAP 头、SOAP 体等必须的封装内容，而本实验只对信息进行小规模的修改，所以更新的数据量只占整个 SOAP 消息的小部分。可以看到，在

请求个数少的情况下，2 种方法的效果相近，而随着请求个数增加，优化后的效果越明显。

(2)实验 2 使用 6 台 PC 机作为服务器，每个服务器模拟一个 UDDI 节点。将系统 1 和系统 2 进行性能比较。系统 1 将整个系统分为 3 个域，域内分别包含 1 个、2 个和 3 个 UDDI 节点，将所有节点的数据库中的内容初始化成相同的内容。系统 2 采用基于 2PC 的传统分布式方法使这 6 个节点进行完全数据同步。

实验 2 利用 Apache JMeter 作为实验工具。首先利用 JMeter 的取样器组件建立 Web 服务的测试项目，并通过线程组元件控制 JMeter 执行测试计划时候使用的线程数(用户)。为每个 UDDI 节点创建 5 个线程，在 2 s 内依次启动这 5 个线程，所有线程同时向各自节点发送 SOAP 更新请求，请求的间隔时间为 500 ms，整个系统的测试时间为 5 min。利用 JMeter 的监听器组件进行一段运行时间内的数据监控，通过图表来说明结果。其中，测试持续时间表示整个系统运行时间，用户响应时间表示单个请求的平均响应时间。

不同系统平均用户响应时间如图 5 所示。

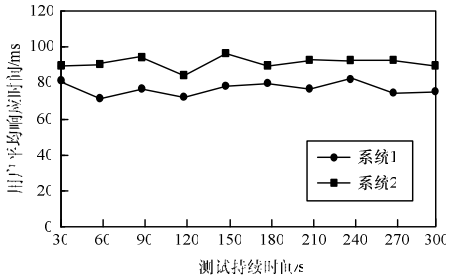

图 5　不同系统平均用户响应时间

从总体响应时间上来看，系统 1 比系统 2 的平均用户响应时间短。因为系统 1 采用两层结构，在域间只需要将请求定时发送域外的头节点上，在域内只需要将请求发送至域内节点，而系统 2 中则需要将请求发送到所有节点上。系统 1 中用户只需要等待少量的节点便可以获得数据的响应，相比系统 2 减少了大量的网络通信量。实验结果表明，本文提出的 2 层的系统结构能有效提高服务注册中心的性能。

7　结束语

UDDI 是 Web 服务技术中服务发布和发现的中心，也是实现电子商务全球化的关键。如何快速发现和定位所需要的服务是 UDDI 实现的目标。本文根据 UDDI 规范提出了跨域服务注册中心系统，并将整个系统分为域内和域间两层，为 UDDI 面向全球化提供了一个可行的系统结构。此外，本文提出的数据同步复制和异步复制相结合的方法也为该系统的数据同步问题提供了有效的解决方案。下一步的目标是增强本文提出的分层数据同步结构在出现意外情况时的处理，并增加服务器及时主动获取所有服务状态的能力。

参考文献

[1] Januszewski K, Mooney E. UDDI Version 3 Features List[EB/OL]. (2006-05-10). http://uddi.org/pubs/uddi_v3_features.htm.

[2] 杜　凯, 缪嘉嘉, 杨树强, 等. 数据库复制技术研究进展[J]. 计算机工程与科学, 2008, 30(7): 118-122.

[3] Patino-Martínez M, Jimenez-Peris R, Kemme B, et al. MIDDLE-R: Consistent Database Replication at the Middleware Level[J]. ACM Trans. on Computer System, 2005, 23(4): 375-423.

[4] 张春玲, 吕震宇, 刘遵峰. 基于虚拟日志压缩的数据同步方案[J]. 计算机工程, 2010, 36(18): 67-69.

[5] 崔　伟, 汪诗林. 分布式系统中数据同步机制的研究与实现[J]. 计算机工程与设计, 2007, 28(10): 2259-2261.

[6] 黄　恺, 李　斓, 李建华. 分布式环境下行为感知的信任管理[J]. 计算机工程, 2011, 37(1): 139-141.

编辑　顾逸斐

(上接第 37 页)

4　实验结果与分析

为验证 FIMM 算法的有效性，通过实验将 FIMM 算法与 Apriori 算法进行比较。选取 T1L4D10K 作为事务数据库。T1L4D10K 是由 IBM ALMADEN 数据挖掘中心提供的数据生成工具生成的测试事务数据库文本。T1L4D10K 含有 1 000 个不同的项、10 000 条事务，事务的最大长度是 10，事务的平均长度是 4。

实验的硬件环境为 Celeron 1.4 GHz 的 CPU 和 768 MB 的内存，软件环境为 Windows XP Profession 操作系统，算法用 C#语言编写，在 Microsoft Visual Studio 2008 平台下开发。

2 种算法的运行时间比较如表 4 所示。

表 4　2 种算法的运行时间比较

序号	最小支持度	Apriori 算法运行时间/s	FIMM 算法运行时间/s	
			建立模型库	检索
1	2.000	20.18		0.01
2	1.000	28.15		0.01
3	0.500	57.37	94.85	0.01
4	0.250	208.99		0.01
5	0.125	881.05		0.01
平均时间		239.148	18.98	

随着支持度的降低，Apriori 算法的执行时间呈递增趋势。FIMM 算法中建立模型库的时间为 94.85 s，占整个执行时间的 99.9%，而检索时间固定在 0.01 s。从平均时间来看，FIMM 算法的执行时间小于 Apriori 算法的执行时间。由此可见，FIMM 算法在多个最小支持度挖掘过程中具有时间优势，挖掘效率较好。

5　结束语

本文针对传统频繁项目集挖掘算法的不足，提出基于多个最小支持度的频繁项目集挖掘算法 FIMM。FIMM 算法中模型库的建立是关键步骤，建立模型库之后，针对多个最小支持度的频繁项目集挖掘只需要很少的时间。下一步将扩展该算法，使算法适用于加权事务数据库和事务数据库的更新。

参考文献

[1] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules[C]//Proceedings of the 20th International Conference on Very Large Databases. Santiago, Chile: [s. n.], 1994: 478-499.

[2] Han Jiawei, Pei Jian, Yin Yiwen. Mining Frequent Patterns Without Candidate Generation[C]//Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press, 2000: 1-12.

[3] 吴绍函, 余昭平. 基于矩阵的关联规则挖掘算法[J]. 计算机工程, 2008, 34(23): 31-33.

[4] 陈　文. 基于位矩阵的加权频繁 k 项集生成算法[J]. 计算机工程, 2010, 36(5): 54-56.

[5] Han Jiawei, Kamber M. 数据挖掘: 概念与技术[M]. 范　明, 孟小峰, 译. 2 版. 北京: 机械工业出版社, 2007.

编辑　张正兴