第 32 卷　第 17 期　　计　算　机　工　程　　2006 年 9 月
Vol.32　№ 17　　Computer Engineering　　September 2006
·安全技术·　文章编号：1000—3428(2006)17—0219—03　文献标识码：A　中图分类号：TP393.02

基于 P2P 的证书库系统设计与实现

陈仕权，熊选东
(信息工程大学电子技术学院，郑州 450004)

摘　要：采用 P2P 技术设计了一个证书库系统，与传统的采用目录服务器设计的证书库系统相比，该系统具有良好的可扩展性、健壮性、负载平衡和容灾容错等特性，解决了传统目录服务系统不易配置，出现访问瓶颈的问题。
关键词：P2P；证书库；目录服务系统

Design and Implementation of Certificate Storage Systems Based on P2P

CHEN Shiquan, XIONG Xuandong
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】This paper designs a certificate storage system using P2P technology. Comparing to traditional certificate storage system which designed with directory service system, the certificate storage system is scalability, robust, load balance, fault tolerance. And it solves some problem of traditional directory services such as hard to configure, accessing bottleneck and so on.
【Key words】P2P; Certificate Storage; Directory service system

当前的证书库系统通常是采用目录服务系统构建的，它们遵循轻量级目录访问协议(Light Directory Access Protocol，LDAP)标准，客户端通过标准的接口访问目录服务器。

使用目录服务器构建证书库系统，通常有两种模式：

(1)单机模式。即所有证书信息存放在一台目录服务器上。这种模式通常是在证书量不是很大、证书发布点(即 CA)单一的情况下采用。

(2)多个分布的目录服务器。在证书发布点很多，证书发布量很大的情况下，目录服务器之间通过引用，构成一个逻辑上完整的树状结构。客户端应用程序在访问目录服务器群时，首先定位到一台目录服务器，如果不能找到需要查询的证书信息，通过向上一级的引用提交查询请求，直到找到需要查找的证书为止。

尽管 LDAP 已经非常成熟，在构建证书库系统中也得到了广泛应用，但是，使用目录服务器构建证书库系统也存在一定的问题：

(1)目录服务器的配置比较麻烦；

(2)尽管目录服务器是分布结构，但由于客户端对服务器的访问仍是 C/S 模式，仍然可能出现访问瓶颈及遭到拒绝服务攻击；

(3)尽管目录服务器有数据复制功能，能够提供数据备份服务，但备份机制不够灵活，在出现单点故障的情况下，不能提供可靠的不间断服务。

本文实现了一种基于 P2P 技术构建证书库系统的方案，本系统提供了证书的添加、删除、更新、查找等功能，解决了上述目录服务系统存在的问题，系统在性能方面也能够满足用户的需求。

1 系统体系结构

系统中所有服务器结点组成服务器组，服务器组存放了系统中所有的证书信息，并在组内完成证书的添加、删除、更新、查询等操作。

系统设计分为客户端和服务器端两部分，如图 1 所示，浅色代表服务器结点，深色代表客户端结点。

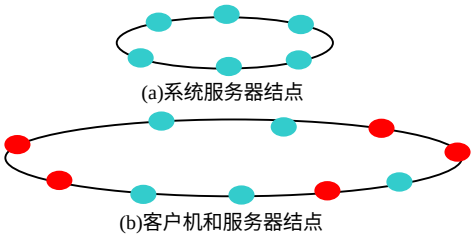


图 1　系统结构

考虑到服务器结点数量不会很多(通常不会超过 100 台服务器)，服务器结点的路由表采用退化的Chord[1]算法，即每个服务器结点维护的路由表包含所有其它服务器结点的地址信息，其逻辑拓扑结构如图 2 所示。

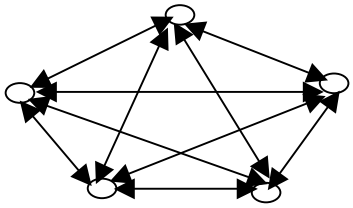


图 2　服务器节点路由

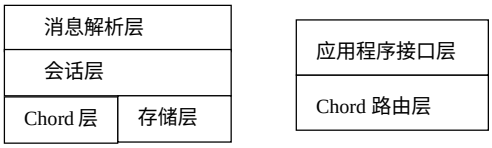
从每个服务器结点出发，都可以直接到达任意一个其它服务器结点。

系统客户端结点发起查询请求，并完成消息路由功能。客户端消息路由采用基于 DHT(Distributed Hash Table)的 Chord 数据查找算法，每个客户端结点通过维护一个路由表

作者简介：陈仕权(1979—)，男，硕士，主研方向：计算机网络安全，对等计算；熊选东，副研究员
收稿日期：2005-11-28　**E-mail：**csqsky@126.com

进行消息路由，本路由表包含了客户端结点和服务器结点。限于篇幅，这里不再详细介绍 Chord 数据查找算法。

系统软件软件如图 3 所示。



(a)服务器端软件结构　　(b)客户端软件结构

图 3 系统软件结构

系统服务器端设计主要分为消息解析层、会话层、Chord 层和存储层 4 层结构。

消息解析层主要完成 2 个功能：

(1)接收从 CA 服务器和客户端应用程序发出的命令(对 CA 服务器端主要是添加、删除、更新等操作命令，对客户端应用程序主要是查询操作命令)，解析后交给下面的会话层进行处理。

(2)接收其它服务器发送来的数据包(主要是系统维护数据包和对存储层进行操作的数据包)，解析后交会话层处理。

会话层的主要功能是根据消息解析层接收到的消息，或重新对消息进行封装交给 Chord 层转发，或向存储层进行读，写、删除、查询等操作，或向 Chord 层发出路由表维护命令。

Chord 层的主要功能是负责 P2P 路由状态的维护，主要是根据结点的加入或退出，维护正确的路由表状态，为上层提供正确的路由信息。

存储层负责证书信息的存储，实现数据条目的查找，加入、删除、更新等操作功能中，并将处理结果交给会话层处理。在本系统中，存储层是由所连接的数据库系统实现的。

2 关键技术

本系统涉及数据分配技术和复制技术 2 个关键技术。

2.1 数据分配

在Chord算法中，数据是通过相容哈希[2]的方法进行分配的。通过对数据条目的某个唯一属性值进行哈希运算，得到该数据条目的一个唯一标识符，通过该标识符与系统中结点 ID(系统结点ID通常是对结点IP地址进行哈希运算得到的，它和数据条目的唯一标识符在相同的名字空间)进行比较，找到结点ID大于数据条目标识符的第一个结点，把该数据条目存放在该结点。

采用这种方法进行数据信息的存储具有两方面的好处：

(1)由于数据条目标识符和结点 ID 都是通过哈希运算(比如 SHA-1)得到的，数据条目的分配可以有很好的负载平衡；

(2)当系统中有新结点加入或退出时，涉及迁移的数据条目较少，只需把后继结点的数据迁移到新加入结点或把失效结点的数据迁移到后继结点即可。

但是，采用 Chord 算法中的数据分配策略存在一个问题，即数据是不可控制的，证书是无法管理的。这对于证书库系统来说是不可接受的。

本文提出了以下方案进行证书信息的分配，该分配方案解决了证书信息管理问题。在介绍分配方案前，首先介绍结点命名问题。

在本系统中，服务器结点名字由两部分组成，前半部分由管理员根据证书的 DN(Distinguish Name)规则命名，比如将一台服务器结点命名为 o=ZhengZhou，c=cn 或 ou=XinXiDaXue,o=ZhengZhou，c=cn 等。DN 规则必须和证书中 DN 规则的部分或全部后缀一致。服务器结点后半部分的名字由结点 IP 地址的哈希值决定。

数据分配规则如下：

规则 1 证书信息存放在和证书具有最大相同 DN 规则的服务器结点上。

假设系统中有以下几个服务器结点，其名字前半部分分别为：

(1)结点 A：o=ZhengZhou,c=cn；

(2)结点 B：ou=XinXiDaXue,o=ZhengZhou,c=cn；

(3)结点 C：ou=DianYuan, ou=XinXiDaXue,o=ZhengZhou,c=cn。

现有一张证书，其 DN 规则为 cn=ZhangSan, ou=XinXi DaXue,o=ZhengZhou,c=cn。根据数据分配规则 1，该证书和服务器结点 B 有最大相同 DN 规则，因此，该证书存放在服务器结点 B。

数据分配规则 1 解决了数据控制问题，方便了数据的管理。数据分配规则 1 仍存在问题，即服务器结点的负载平衡问题。比如，假设某个证书认证中心(CA)产生了大量具有相同 DN 后缀的证书，假设证书 DN 后缀均为 ou=XinXiDaXue,o=ZhengZhou,c=cn，这样，大量的证书信息都存放在服务器结点 B，必然造成服务器结点 B 的过载。为解决这样的问题，可以添加一台名字前半部分为 ou=XinXiDaXue,o=ZhengZhou,c=cn 的服务器，并希望该服务器能够承担和服务器 B 大致相同的负载。

为实现上述设想，本文给出了数据分配规则 2。

规则 2 对于一条证书信息，假设有满足数据分配规则 1 的两个或两个以上的服务器结点(这些服务器结点名字的前半部分不一定完全相同)，那么该条信息存放在其中的一个服务器结点，该结点满足以下条件：

满足数据分配规则 1 的服务器结点为一组，根据名字的后半部分的值(结点 IP 地址的哈希值)，把它们按照由小到大的顺序，顺时针顺序组成一个圈。对证书序列号进行哈希运算(对证书序列号的哈希运算值与服务器结点名字后半部分值在相同名字空间内)，并根据该哈希值的大小把该值分配到上述服务器结点所组成的圈中，那么，从证书哈希值开始按顺时针方向遇到的第 1 个服务器结点负责存放该证书信息。

数据分配规则如图 4 所示，假设对于一条证书信息，满足数据分配规则 1 的有 3 个服务器结点 A，B，C，其名字的后半部分(服务器结点 ID 的哈希值)分别为 235、437、1 350。现通过对该证书序列号进行哈希运算，假设其哈值为 1 355，那么，根据数据分配规则 2，该证书信息存放在名字后半部分为 235 的服务器结点 A 上。

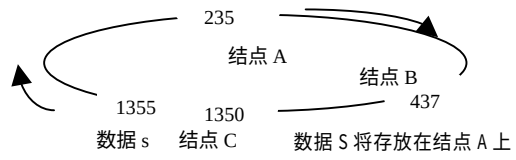


图 4 数据分配规则 2

由于数据的分配是根据哈希运算进行的，因此，该分配方案可以做到负载平衡。

在结点加入和退出时，该数据分配方案也满足数据迁移量不很大的要求。

2.2 数据复制/迁移算法

为了保证系统中存储的数据不被丢失或损坏，维护数据的可用性，通常要为数据在系统中保存一定数量的备份，这是通过数据复制/迁移算法实现的。

在介绍数据复制/迁移算法之前，首先定义以下两个

概念：

(1)原始数据：通过数据分配算法分配到某个结点上的数据称为该结点的原始数据。

(2)后继结点：在所有由服务器结点组成的服务器结点组内，根据结点的命名规则，按照下面的规则，所有服务器结点组成一个环。

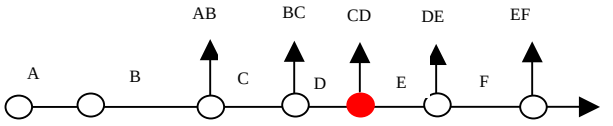
规则 3 沿顺时针方向，服务器结点名字中前半部分 DN 规则长的排在 DN 规则短的结点后面。

规则 4 在具有相同 DN 规则长度的服务器结点中，结点名字后半部分大的排在结点名字后半部分名字小的后面。

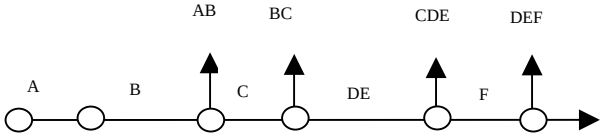
按照上述两个规则，所有服务器结点组成一个环，对于环中的任意一个结点，沿顺时针方向遇到的第一个结点称为该结点的后继结点。

数据复制/迁移算法是这样的：每个结点中的后继结点负责保存该结点中的原始数据。

图 5 举例说明了一个结点在失效前后数据迁移情况。图 5 所示为按照规则 3 和规则 4 组成的圆环的一部分，箭头方向为顺时针方向，圆圈代表服务器结点。A，B，C，D，E，F 代表分布在名字空间中的数据。



(a) 红色结点失效前各结点数据分布



(b) 红色结点失效后情况各结点数据分布

图 5 数据迁移

由图 5 可以看出，当深色结点失效后，深色结点上的数据 C 和 D 分别迁移到它的第 1 个后继结点和第 2 个后继结点上。

根据以上的示例，当可以把系统中对每条数据的维护备份扩大到 N 的情况，如 N=6 时，对每个结点中的存储区分为 6 个区，第 1 个区存放第 5 个前驱结点中的原始数据，第 2 个区存放第 4 个前驱结点中的原始数据，……，第 6 个区存放本地结点中的原始数据。这样，当一个结点失效时，该结点把第 1 个区中的数据复制到第 1 个后继结点中的第 1 个存储区，把第 2 个区中的数据复制到第 2 个后继结点中的第 1 个存储区，把第 3 个区中的数据复制到第 3 个后继结点中的第 1 个区，……，把第 6 个区中的数据即本地结点的原始数据复制到第 6 个后继结点中的第 1 个区。

当系统中有结点加入时，数据迁移过程跟结点退出时相反。加入结点首先获得自身的原始数据，然后为其前趋结点备份数据，通知后继结点删除相关数据。

3 系统实现

按照本文提出的软件结构，本系统采用 C 语言进行了实现。在实现时，客户端 Chord 层采用公开源代码的 OpenDHT，OpenDHT 是一个开放源代码的基于 DHT 的 P2P 数据查找系统，它提供了可访问的分布式哈希表服务。

服务器端存储层由 SQL Server 数据库实现。SQL Server 提供了强大的数据存储能力和丰富的数据查询、添加、删除更新等功能。对于数据的备份，SQL Server 通过建立不同的数据表实现为其它结点的数据进行备份的功能。

4 结论

采用 P2P 技术构建的证书库系统是一个具有可扩展性、容灾容错特性和负载均衡的系统，该系统提供了基本的证书信息查询功能，能够满足大量用户同时对证书信息查询的需求。与采用 LDAP 目录服务系统构建的证书库系统相比，本系统在某些方面还不够完善，比如系统安全性方面没有被考虑等。因此，完善系统的安全性是下一步需要研究的工作。

参考文献

1 Stoica I, Morris R, Karger D, et al. Chord: A Scalable P2P Lookup Service for Internet Applications[C]. Proc. of the ACM SIGCOMM. ACM Press, 2001: 149-160.

2 Karger D, Lehman E, Leighton T, et al. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web[C]. Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997-05: 654-663.

3 Dabek F, Zhao B, Druschel P, et al. Towards a Common API for Structured P2P Overlays MIT Laboratory for Computer Science[C]. Proc. of the 2nd International Workshop on Peer-to-peer Systems, Berkeley, CA, 2003.

4 Dabek F M, Kaashoek F, Karger D, et al. Wide-area Cooperative storage with CFS.MIT Labora-tory for Computer Science[C]. Proceedings of the 18th ACM Symposium on Operating Systems Principles, 2001.

5 Karger D R, Ruhl M. Simple Efficient Load Balancing Algorithms for Peer-to-Peer systems[C]. Proc. of the 3rd International Workshop on P2P Systems, 2004.

（上接第 218 页）

参考文献

1 Mambo M, Usuda K, Okamoto E. Proxy Signatures: Delegation of the Power to Sign Message[J]. IEICE Transaction on Fundamentals, 1996, 79(9): 1338-1353.

2 Kim S, Park S, Won D. Proxy Signature, Revisited[C]. International Conference on Information and Communication Security, Berlin, 1997.

3 Sun H M, Lee N Y, Wang T H. Threshold Proxy Signature[J]. Proc. of the Comput. Digit. Technol. 1999, 146(5): 259-263.

4 Hsu C L, Wu T S, Wu T C. Improvement of Threshold Proxy Signature Scheme[J]. Applied Mathematics and Computation, 2003, 136(2/3): 315-321.

5 Wu T S, Wu T C. New Nonrepudiable Threshold Proxy Signature Scheme with Known Signers[J]. System Software, 1999, 22(8): 119-124.

6 Yang C Y, Tzeng S F, Hwnag M S. On the Efficiency of Nonrepudiable Threshold Proxy Signature Scheme with Known Signers[J]. System Software, 2004, 73 (3): 507-514.

7 Shamir A. How to share a Secret[J]. Commun., 1979, 22(11): 612-613.

8 Hsu C L, Wu T S. Self-certified Threshold Proxy Signature Schemes with Message Recovery, Nonrepudiation and Traceability[J]. Appl. Math. Comput., 2005, 164 (1): 201-225.