在 Windows 上实现 NFS 的用户资源管理机制*

王晓亮[1]，朱珍民[1,2]，李　颖[2]

(1. 湘潭大学 信息工程学院，湖南 湘潭 411105; 2. 中国科学院 计算技术研究所，北京 100080)

摘　要: 针对 Linux NFS 系统只能基于主机的访问控制, 提出了一种在 Windows 上实现 NFS 的用户资源管理机制的方法。

关键词: NFS; Windows 服务器; 用户资源管理

中图法分类号: TP393. 07　　**文献标识码**: A　　**文章编号**: 1001-3695(2005) 01-0197-03

Realize NFS Service Based on User Resource Management with Windows OS

WANG Xiao-liang[1], ZHU Zhen-min[1,2], LI Ying[2]

(1. College of Information & Engineering, Xiangtan University, Xiangtan Hunan 411105, China; 2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: Compared with Linux NFS's access control basd on host, this paper puts forward a way to realize NFS service based on user resource management with Windows OS.

Key words: NFS; Windows Server; User Resource Management

1　引言

网络计算机(NC) 采用了具有本地操作系统的瘦客户机模式, 系统能够实现客户机的零管理, 非常适合于网络教育、电子政务等应用环境。传统的计算机系统运行环境都保存在本地的文件系统中, 在 NC 环境下, 系统的运行环境需要从网络计算机本地和网络服务器的多处数据获取, 特别是网络服务器存放了绝大部分的数据。

Linux 的 NFS 网络文件系统为网络存储提供了良好的机制。NFS 是由 SUN 公司发展, 并于 1984 年推出的技术。NFS 是一个 RPC 服务, 它的设计是为了在不同的系统间达到文件的共享。

网络计算机的用户多, 当用户想用远端文件时访问量大, 因此需要开发出一套基于用户的适合 NC 应用的网络文件系统。而在目前所有的 Linux NFS 服务中, NFS Server 所导出的文件或目录都记录在一个 Export 的文件中, 虽然 Linux 下的 NFS 可以在 Export 文件中设置导出文件和目录的用户读写权限, 但存在安全问题。来自客户端 NFS 请求的用户认证, 由用户的 UID 和所属组的 GID 组成, 这种文件访问的安全验证对于没有开放 NFS 服务的系统是安全的。但是在网上, 其他机器的 Root 完全有权在自己的机器上设置这样一个 UID, 而 NFS 服务器无论这个 UID 是否在自己的机器上, 只要 UID 符合, 就赋予这个用户对这个文件的操作权。比如, 目录 /home/frank 只能由 UID 为 501 的用户打开读写, 而这个目录可以被远程机器 Mount, 那么, 这台机器的 Root 用户新增一个 UID 为 501 的用户, 然后用这个用户登录并 Mount 该目录, 就可以获得相当于 NFS Server 上的 UID 为 501 用户操作权限, 从而读写 /home/frank。要解决这个问题, 只能通过配置 Export, 限制客户的主机地址, 这就导致了 NFS 的资源管理机制基于主机, 当不同的用户在同一客户机下 Mount NFS 文件系统的时候, 导出的文件与目录都是一样的。

但是, 从实际应用角度考虑, 希望相异用户在不同客户机利用 NFS 访问服务器资源时, 服务器导出的都是该用户权限允许目录以及公共目录所虚拟合成的用户专有目录。所以, 如果能在传统的 NFS 系统上进行扩展, 进一步深化用户的概念, 实现基于用户的 NFS 资源管理, 非常有实际应用价值。

收稿日期: 2004-03-05; 修返日期: 2004-04-19

基金项目: 国家"863"计划资助项目(2001AA114060) ; 北京市科技重大基金资助项目(20037060)

2　NFS 基本原理和关键技术

2.1　NFS 基本原理

NFS 是由客户端和服务器共同实现的: 客户端通过一些核心函数调用来使用远程文件系统; 而服务器端, 由 NFS 服务器监听进程来提供文件数据的操作。一般来说, 最主要是两个监听进程 Mountd 和 Nfsd, 其中 Mountd 用来监听客户的安装请求, 并发送相应的应答信息, 如客户端地址和服务器地址; 而 Nfsd 进程用来监听客户端的读写文件请求, 并返回相应的文件数据。具体地说, Client 方面主要负责处理用户对远程文件的操作请求, 并把请求的内容按一定的包格式从网络发送给文件所在的 Server 方面; 而 Server 方面则接收 Client 方面的请求, 调用本机的系统函数进行文件的实际操作, 并把结果按一定格式返回给 Client 方面, Client 方面得到 Server 的返回结果后, 把它返回给用户。图 1 为 NFS 网络文件系统的基本体系结构。

2.2　NFS 的文件和目录操作

NFS 的主要优点是可以将占用大量磁盘空间的或用户共

享的数据只保存在一个 NFS 服务器上。当其他客户机要访问这些数据时,只需通过 NFS 将其安装到本地目录进行透明的访问。所谓透明的访问,是指访问这些文件与访问本地的一般文件的用户界面是一致的,并不需要额外的命令。所以 NFS 文件的访问对客户来说是完全透明的,可以跨越各种服务器和主机平台进行。它与其他文件系统的不同主要在于底层的访问上,其他文件系统通过本地磁盘直接访问文件,而 NFS 在底层通过 RPC(远程过程调用)协议访问远程服务器上的文件和目录(图 2)。

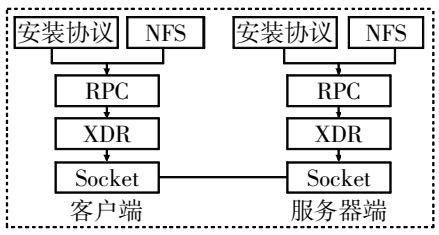

图 1 NFS 网络文件系统的基本体系结构

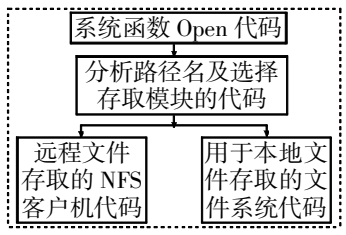

图 2 本地文件和远程文件的操作

在 NFS 中,每次通过对远程文件系统上一个称之为“文件句柄”的数据结构来实现对远程机器上目标文件的操纵。首先对文件名在本地进行解析,每次解析全路径名的一部分。它从分层结构的根及路径的开始出发,重复地从路径中取出下一部分,并找出一个具有该名字的文件或子目录。

因此在 NFS 系统中,一个目的文件句柄的获得是分多个步骤实现的。首先必须得到服务器为目录和文件提供的最初句柄,这是由 Mount 安装协议取得该 NFS 服务器上的分层文件结构信息,并取得相应文件系统的根句柄实现的。在得到一个远程文件系统的根句柄后,结合本地对文件名字解析的结果,调用 NFS 的远程过程,在当前远程文件系统根句柄下取出各个子目录的文件句柄返回,检查返回的文件句柄,得到最后所要访问的文件句柄。以后对该文件的各种操作,就通过该文件句柄来实现。

图 3 展现了当客户端在服务器的中查找路径为/a/b/c 的文件时,客户端和服务器进行的句柄交流。

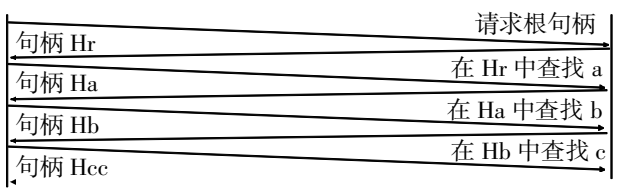

图 3 NFS 的句柄交流

Mount 安装协议主要是根据服务器端的 Export 文件加载目录的。一个典型的 Export 文件一般如下:

```
# exports file for nfs
/home/wxl    wxl(rw)
/usr/x386    192.168.0.1
/home/ftp
```

每行的第一字段定义一个导出目录,后面是允许加载该目录的主机名或 IP 地址。缺省的时候任何 IP 地址的主机都能加载。

下面就以一个具体的研究项目来说明如何在 NFS 系统中加入基于用户资源管理机制的基本思想。

3 基于用户资源管理的 Windows 平台 NFS 服务器

根据 NFS 系统的原理,结合加入用户资源管理的思想,设计了基于用户资源管理的 Windows 平台的 NFS 服务器。工作环境为 NC 端客户机(Linux),Windows 服务器。

3.1 Linux 客户端的安装协议加入用户信息

考虑到避免破坏其他文件系统的安装,故不在原有的 Mount 程序基础上改动。另外做了一个针对用户资源管理的 NFS Mount 程序(与原来的 Mount 程序也兼容),该程序使得不同的用户通过数字签名认证后加载 NFS 文件系统,并将自己的用户信息传送到服务器端。

3.2 Windows 服务器端的 NFS 服务器设计

Windows 服务器端我们仿照国外一款叫 Sossnt 软件的思想,设计了一个 Windows 平台的 NFS 服务器。传统 Linux 的 NFS 访问是通过 Inode 节点对磁盘文件进行访问的。为了不破坏 NFS 的结构和平台的可移植性,我们在 Windows 服务器上虚拟建立了一个 Inode Interface,通过它与 Windows 磁盘文件进行交流。服务端的流程设计如下:

服务主程序首先进行网络 Socket 的初始化,然后依次启动三个初始化进程,其中 pmap_init 负责 Port Mapper 的创建,mountd_init 负责 Mount Daemon 的创建,nfs_init 负责 NFS Server Daemon 的创建。

当服务器端的三个初始化进程成功后,主程序利用 svc_run 开始等待客户端发送来的服务请求。一旦接收到请求,svc_run 调用一个叫 svc_getreq 的函数进行循环,处理从网络端口传送过来的客户信息。该程序 svc_getreq 中有一个 sc_dispatch 函数指针针对不同的服务请求调用 pmap_dispatch 处理端口,mountd_dispatch 处理加载, nfs_dispatch 处理具体 NFS 服务。其中在 nfs_dispatch 中又可以根据具体的 NFS 服务分别调用 nfs_read, nfs_rename, nfs_setattr 等服务程序。

3.3 在 Windows 平台下 NFS 服务器加入用户资源管理机制

研究的目的是为了实现 NFS 基于用户的资源管理。通过上面对 NFS 基本原理和关键技术的介绍,可以看到 NFS 服务器,客户端的交流主要是通过两者间的句柄进行的。所以要加入用户资源的管理机制,实际上就是要在做到不同的用户 Mount 后返回他们各自相应的句柄。

因此,服务器接收到客户端的 mount 命令后,在 svc_run 模块中,根据用户信息通过服务器的 Export 文件开始构造 inode_interface。其中,inode_interface 是根据客户端传送过来的用户信息虚拟合成该用户的 Inodetable,里面既包括了该客户的私有目录 Inode,也包含了公共部分的目录 Inode,然后反馈给客户端一个相应合成目录的句柄。客户端通过这个句柄对服务器上的网络文件进行操作,这样就达到了 NFS 基于用户的资源管理模式。

具体的 inode_Interfce 构造为:首先,NFS 服务器的 Exports 文件导出客户机使用的所有资源目录;然后设计一个用户管理的模块,当不同的用户登录服务器时,根据用户名,NFS 服务端读取一个管理者配置的 Manager 文件。Manager 文件以记录项形式规定所有登录用户的私有目录和允许访问的一些公共目录。通过读取 Manager 文件中该用户的私有目录记录构造用户私有 Inodetree,通过读取公共目录记录构造公用部分 Inodetree,最后合成该用户的 Inodetable 数据结构,同时把 Inodetable 结构相关的文件根句柄传送给客户端。这样客户端就可以根据该句柄对服务器上的文件系统进行操作。

Inodetable 的结构设计为

```
typedef struct DIRNODE {
    char name[ MAXFILENAMELEN] ; /* 文件和目录名 * /
    u_char fsid; /* Windows 磁盘驱动器号, 如 a: 为 1 * /
   u_short inode; /* inode 节点号 * /
   struct DIRNODE * next; /* 同目录的下一个文件和目录 * /
   struct DIRNODE * subdir; /* 指向子目录 * /
   struct DIRNODE * parent; /* 指向父目录 * /
   Acache_t * attr; /* 文件属性 * /
} Dir_t;
typedef Dir_t * Inode_t;
Inode_t * InodeTable;
```

文件句柄的结构设计为:

```
typedef struct {
    struct fhs f;       /* 文件和目录本身 * /
    struct fhs p; /* 父目录 * /
    char fh_pn[ FH_PN] ; /* 路径名 * /
} fhandle_t;
```

其中 struct fhs 表示句柄的状态。其结构如下:

```
struct fhs {
int fh_fsid; /* Windows 磁盘驱动器号, 如 a: 为 1* /
u_long fh_fno; /* 文件和目录的 inode 节点号 * /
} ;
```

4　结束语

关于 NFS 的产品现在有很多, 但不难发现, 所有的产品提供的依然是传统 NFS 系统的基于主机的访问控制机制, 并不满足网络计算机基于用户资源访问的性能要求, 没有实现动态的主动式的虚拟文件系统(即不同的用户显示不同的目录文件) 的需求。所以在 NFS 文件服务中加入用户资源管理, 改善了 NFS 认证机制只能基于主机访问的缺点, 进一步深化了用户的概念。这种全新的 NFS 资源管理机制是具有实用价值的, 它改变了传统的 NFS 单调的基于主机访问管理机制, 使得每一个用户在登录到 NFS 服务器后看到的是一个该用户主动式虚拟文件系统。也就是说, 该用户能授权访问的所有文件和目录以及公共目录, 一旦登录, 自会一目了然。

参考文献:

[1] 宋鹏, 王靖滨, 余可曼. Linux 网络文件系统(NFS) 分析[Z] . 杭州: 浙江大学计算机系, 1999.

[2] 毛德操, 胡希明. Linux 内核源代码情景分析(上册) [M] . 杭州: 浙江大学出版社, 2001.

[3] Douglas E Comer, David L Stevens . 用 TCP/IP 进行网际互连(第 1, 3 卷) [M] . 北京: 电子工业出版社, 1998.

[4] http: / /www. loa. espci. fr/winnt/sossnt4 /sossnt4. htm[EB/ OL] .

[5] http: / /www. ssc-corp. com/NFS/ds-br. asp[EB/OL] .

[6] http: / /www. lnu. edu. cn/nt/ d16 . html[EB/ OL] .

[7] http: / /it. rising. com. cn/ safety/ syjq/fhzq/020620nfs. htm [EB/ OL] .

作者简介:

王晓亮(1977-) , 男, 湖南湘潭人, 硕士研究生, 主要研究方向为计算机网络; 朱珍民(1962-) , 男(土家族) , 湖南慈利人, 教授, 硕士生导师, 主要研究方向为计算机仿真算法、并行算法、网络计算机; 李颖(1971-) , 男, 四川人, 博士研究生, 主要研究方向为计算机系统和信息安全、计算机接口技术。

(上接第 196 页) 和 C_2 在五个不同的区间内随机产生。对于 C_1, C_2 的每一种区域情况, 分别使用上述的四种算法进行 10 次实验(Chen 的算法中分别取 $x = 5, x = 15$) , 每次实验随机产生 1 000个连接请求, 最后得出 SR 的平均值, 如表 1 所示。

表 1　各算法成功率情况比较

$C1$ 和 $C2$ 的取值范围	最优算法	Chen 算法		Korkmaz 算法	MHMCA
		$x = 5$	$x = 5$		
$C_1 \sim [50, 65]$ $C_2 \sim [200, 260]$	0.248	0.223	0.240	0.247	0.247
$C_1 \sim [75, 90]$ $C_2 \sim [300, 360]$	0.503	0.462	0.491	0.503	0.501
$C_1 \sim [100, 115]$ $C_2 \sim [400, 460]$	0.758	0.714	0.749	0.757	0.758
$C_1 \sim [125, 140]$ $C_2 \sim [500, 560]$	0.925	0.871	0.909	0.925	0.925
$C_1 \sim [150, 165]$ $C_2 \sim [600, 660]$	0.985	0.978	0.983	0.984	0.984

从表 1 可以看出: ①随着 C_1 和 C_2 取值范围的增大, 各算法的 SR 明显增加, 这是由于约束值越宽松, 找到可行路径的概率就越大; ②同时比较几种算法可以看到, MHMCA 算法和 Korkmaz 算法的成功率都非常接近最优算法的成功率, 而 Chen 算法当 x 取值较小时, 性能不是很好, 当 x 的值增加到 15 时, 其 SR 也比较接近最优算法了; ③虽然本文算法和 Korkmaz 算法的成功率几乎一致, 但是本文算法能找到所有满足约束的最小跳路径, 而 Korkmaz 算法仅能找到一条可行路径。

5　结论

多约束 QoS 路由问题是一个 NP 完全问题。本文提出了一种解决多约束路径问题的算法——多约束最小跳路径算法(MHMCA) 。利用 Bellman-Ford 算法首先将图中无用链路标记并删除, 再对简化图运用基于堆栈的深度优先搜索算法寻找满足约束条件的所有最小跳可行路径。MHMCA 算法具有如下特点: ①寻找满足约束条件的最小跳路径, 高效地利用了网络资源; ②最坏情况下, 本算法的时间复杂度为 $O(n^3)$, 在复杂性方面优于文献[1, 3, 5] 中的算法; ③本算法可以找到 MCP 中的所有最小跳可行路径; ④仿真实验结果表明, 本算法搜索最小跳可行路径的功率很高。

参考文献:

[1] Chen, *et al*. On Finding Multi-Constrained Paths[C] . International Conference on Communications, ICC '98, IEEE, 1998. 874-879.

[2] T H Cormen, *et al*. Introduction to Algorithms[M] . The MIT Press and McGraw-Hill Book Company, 2001.

[3] J M Jaffe. Algorithms for Finding Paths with Multiple Constraints[J] . Networks, 1984, 14: 95-116.

[4] T Korkmaz, M Krunz. A Randomized Algorithm for Finding a Path Subject to Multiple QoS Requirements[J] . Computer Networks: The International Journal of Computer and Telecommunications Networking, 2001, 36(2) : 251-268.

[5] H De Neve, *et al*. A Multiple Quality of Service Routing Algorithm for PNNI[C] . Proc. of the ATM Workshop, IEEE, 1998. 324-328.

[6] E Comer. 用 TCP/IP 进行网际互连, 第 1 卷: 原理、协议和体系结构[M] . 林瑶, 等. 北京: 电子工业出版社, 1998.

作者简介:

张琨(1977-) , 女, 讲师, 博士, 主要研究方向为计算机网络通信与安全技术; 王珩(1977-) , 男, 博士研究生, 主要研究方向为 QoS 路由及组播路由; 刘凤玉(1943-) , 女, 教授, 博士生导师, 主要研究方向为计算机网络与通信技术; 衷宜(1978-) , 女, 助教, 研究方向为网络通信技术。