《网络存储技术》课程作业一

NAS 技术概述

付容天 学号 2020211616

班级 2020211310

计算机科学与技术系

计算机学院（国家示范性软件学院）

2022 年 10 月 9 日

目录

1 引言

NAS（Network Attached Storage，网络附加存储）技术是一种将分布、独立的数据整合为大型、集中化管理的数据中心、从而便于不同主机和应用对服务器进行访问的文件级技术。简单来讲，NSA 就是一种“将文件数据存储在网络上供主机进行存取和处理”的技术。

NAS 可以被定义为一种特殊的专用数据存储服务器，其最重要的功能就是跨平台文件共享功能。NAS 通常在一个 LAN 上占有自己的节点，无需其他应用服务器的干预，并允许用户从该网络上进行数据的存取、更新、删除等操作。在这种配置中，NAS 集中管理和处理所在局域网上的所有数据，有效降低数据管理和共享的成本。

2 NAS 概要

2.1 专用目的

NAS 通常占用所在 LAN 上的一个专用节点，从而对其他其他主机提供数据服务。从此角度来看，不妨认为 NAS 服务器是一种专用的数据服务器，它可以授权网络用户和异质客户端从集中位置存储和检索数据。显然，NAS 具有灵活和易于横向扩展的特点。

NAS 的专用目的决定了其硬件和软件基础也通常是专用的，其硬件、软件和配置共同构成其颇具特色的文件服务。NAS 的实现基础通常包含一个或多个通常排列成逻辑存储器、冗余存储器或 RAID 存储驱动器的网络设备。通常来讲，用于 NAS 的硬盘驱动器在功能上与其他驱动器相似，但在固件、振动耐受性或功耗上有不同之处，以使其更适合在 RAID 阵列中使用。

2.2 文件级存储

NAS 技术是一种“文件级存储”的技术，也就是说文件存储的用户是自然人。数据按照不同应用要求的结构方式组成不同类型的文件（往往通过不同后缀来指代不同的类型），并且，每一组文件放在用一个目录（文件夹）下，所有的文件和目录共同构成一个树状结构。

这种文件级的存储方式大大方便了 NAS 技术中的文件共享，如果采用块级存储，那么对于不同主机而言，不同文件系统间的文件是无法进行共享的，这是因为不同的文件系统需要对数据文件进行不同的格式化后才能得到数据；而文件级存储方式本身具有文件的管理功能，因此不需要主机系统再对文件进行格式化，相当于引入了一种“中间件”来进行文件的统一管理并提供对外的统一接口，从而方便了文件的共享。

2.3 NAS 与 DAS 的比较

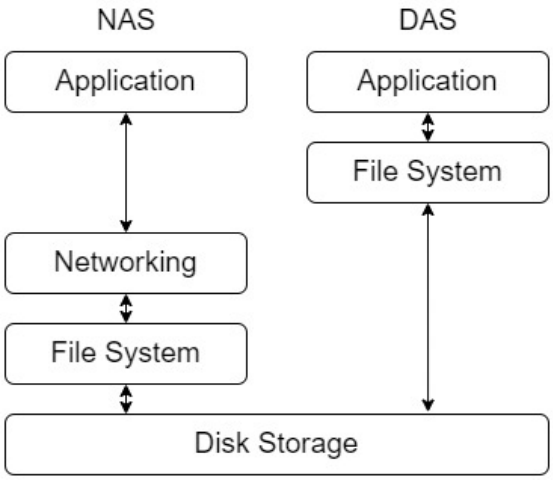

图 1: NAS 和 DAS 的比较

DAS（Direct Attached Storage，直连式存储）技术是一种将数据直接存储在本地（未接入网络）的存储设备上的存储方式，其通常由机械硬盘、固态硬盘、光盘等存储单元构成。一个典型的 DAS 设备通常通过 HBA（Host Bus Adapter，主机总线适配器）直接连接到计算机上，通常作为已有存储设备的拓展，这决定了 DAS 技术不具有易于在不同主机之间直接进行数据共享的特点，因为在不同主机之间没有类似于 hub、switch、router 等网络设备进行连接。

和 DAS 相比，NAS 具有的优点包括：

- 通过网络连接，便于进行不同主机之间的数据共享
- 当网络情况稳定时，NAS 提供的服务往往具有更好的性能，因为 NAS 设备可以针对文件服务进行精确调整

而 NAS 相比 DAS 的缺点包括：

- NAS 提供的文件服务不如 DAS 稳定，NAS 提供的服务往往和当前网络的速度和拥塞程度有极大的关系，而 DAS 服务的质量则仅取决于本地计算机的硬件配置，其服务质量是可以在一定程度上得到保证的
- DAS 可以提供更加具有特色的存储服务，往往支持对硬件和低级软件的定制；而 NAS 受限于网络连接要求，其硬件基础往往是不支持定制的

2.4 NAS 和 SAN 的比较

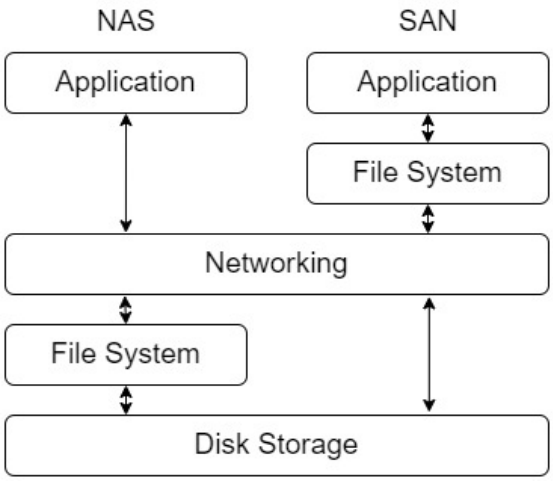

图 2: NAS 和 SAN 的比较

SAN（Storage Area Network，存储区域网络）技术提供对整合的块级数据存储的访问，主要用于从服务器访问数据存储设备，以便这些设备在操作系统看来是直连式存储。和 NAS 不同，SAN 通常是无法通过 LAN 直接访问的专用存储网络，SAN 仅提供块级访问，但构建在 SAN 之上的文件系统则提供文件级访问，这个文件系统被称为共享磁盘文件系统。SAN 是最简单的数据存储专用网络，它源于以数据为中心的大型机架构，互相连接的存储设备确保了存储设备之间的数据传输发生在服务器后面，并且对一般用户来讲是透明的。

SAN 支持的协议包括光纤通道协议（FCP）、Internet 小型计算机系统接口协议（iSCSI）、以太网光纤通道协议（FCoE）、基于光纤通道的非易失性内存标志协议（FC-NVMe）等协议，其中影响力最大的是 FCP 协议，其使用具有嵌入式 SCSI 命令的光纤通道传输协议，部署量占据 SAN 市场总额的 70

和 SAN 相比，NAS 具有如下优点：

- 造价较低，在 NAS 架构中，数据通过成熟的 TCP/IP 协议进行传输，但在 SAN 中则必须使用专有协议例如 Fibre Channel、iSCSI 和 Infiniband 等，因此 SAN 通常有自己的网络和存储设备，必须单独购买、安装和配置。这使得 SAN 技术的安装和使用比 NAS 技术昂贵
- NAS 提供统一的文件系统，方便文件的存储和共享，而 SAN 仅提供基于块的存储，并将文件系统问题遗留给了用户端

而 NAS 相比 SAN 的缺点包括：

- NAS 架构大量使用以太网和 TCP/IP 协议进行内存传输，这样做不但增加了大量的 CPU 指令周期，而且使用了低速传输介质；但 SAN 中大部分操作都由适配卡上的硬件完成，CPU 开销的增加有限，因此 NAS 架构的速度一般无法超越 SAN 架构

- 目前 NAS 的根本瓶颈是底层链路的速度，在底层链路速度较低或不稳定时，NAS 架构提供服务的质量显著低于 SAN 架构

需要注意的是，尽管存在差异，但 SAN 和 NAS 并不相互排斥，并且可以组合为 SAN-NAS 混合体，从而提供来自同一系统的文件级协议（NAS）和块级协议（SAN）。

3 NAS 技术介绍

3.1 AFS

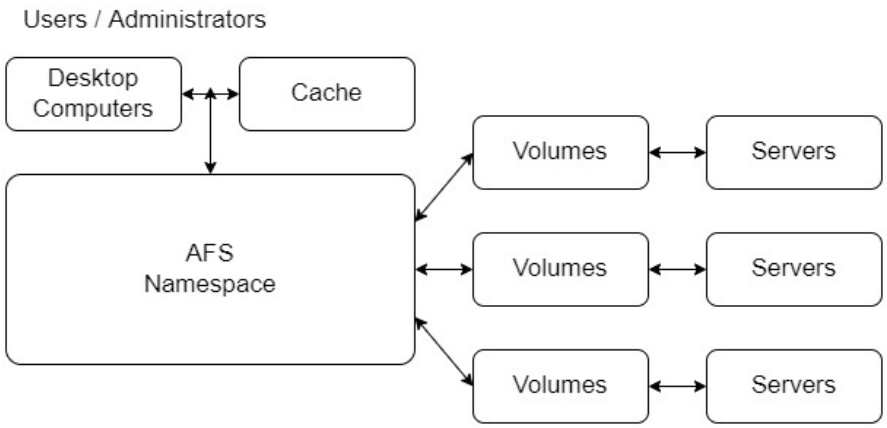

图 3: AFS 概念图

Andrew File System（AFS）是一种分布式的文件系统，它使用一组服务器向所有客户端提供概念上完全相同的存储空间，其主要用途是分布式计算。AFS 的最大特点是安全性和可扩展性：

- 安全性：AFS 使用 Kerberos 协议对用户身份进行验证，该协议允许通过非安全网络进行通信的节点以安全的方式验证彼此的身份（相互验证），该协议要求一个受信任的第三方来辅助对称密钥加密技术，在 AFS 中，被选择的那一组服务器必须是受信任的，从而使 Kerberos 协议可以正常工作
- 可扩展性：AFS 能够很容易地支持数百个节点，甚至数千个节点的分布式环境。同时，在大规模的分布式文件系统中，AFS 利用本地存储作为分布式文件的缓存，在远程文件无法访问时，依然可以部分工作，提高了系统可用性

AFS 的一个重要特点就是卷、文件树、子目录和 AFS 挂载点等概念的出现和实现。卷是由管理员创建并且链接到 AFS 单元中的特定命名路径，卷的物理位置对文件系统的普通用户是不重要的（不可见的），用户可以正常创建目录和

文件。一个卷可能有一个配额，以限制其空间消耗量，管理员可以改变这个配额、甚至可以改变卷的存储位置。

对于存储空间而言，AFS 系统将其划分为共享名称空间和本地名称空间。共享名称空间在所有工作站上都是相同的，而本地名称空间则对于每个工作站而言是唯一的。

在修改文件时，AFS 则采用了弱一致模型：用户在打开文件后，将文件缓存在本地文件系统上，对打开文件的更新操作仅对本地副本进行修改，当修改后的文件关闭时，修改的部分才会被复制回文件服务器。

AFS 可以对不同用户进行以下授权：

- Lookup(l)：允许用户列出 AFS 系统的文件目录、访问子目录和进行检索
- Insert(i)：允许用户向存储系统中添加新文件或创建新的子目录
- Delete(d)：允许用户删除文件或子目录
- Administrator(a)：给一个用户在对应目录下授予管理员权限
- Read(r)：允许用户查看目录中文件的内容并列出子目录中的文件
- Write(w)：允许用户修改目录中的文件

3.2 AFP

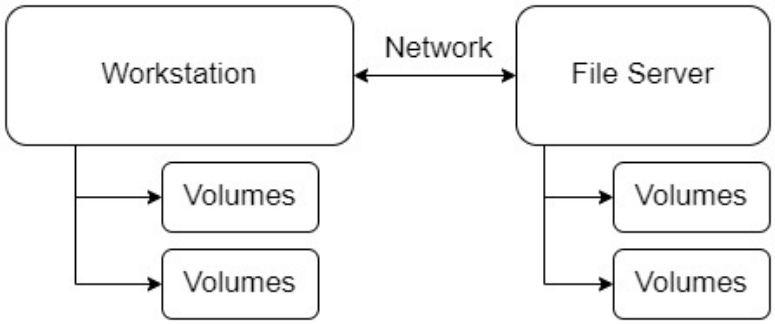

图 4: AFP 概念图

Apple Filing Protocol（AFP）是一种专有网络协议，是 Apple 文件服务的一部分，主要用于控制访问远程文件系统。该协议的典型应用程序是 AppleShare，它主要为各种计算机用户提供文件共享服务。

在访问文件时，通过本地计算机上运行的程序向本地文件系统发送命令。本地存储器中的数据结构可以指示某些卷是由本机文件系统还是由外部文件系统管理，本地文件系统通过查看此数据结构来发现请求的文件是在本地还是远程访问。如果数据结构指示外部文件系统，则本地文件系统将该命令路由到 AFP 转换器（AFP 3.0 及更高版本完全依赖 TCP/IP 经由端口 548 建立通信）。

AFP 主要由三个系统组件构成：

- 文件系统结构：其由可通过网络寻址的资源（诸如文件服务器、卷、目录、文件和分支等）组成，这些资源称为 AFP 文件系统可见实体，AFP 可以规定这些实体之间的关系，例如，一个目录可以是另一个目录的父目录
- AFP 命令：这是本地计算机用于操作 AFP 文件系统结构的命令，支持以文件服务器的形式发送文件系统命令，或者在本地计算机上运行的应用程序直接进行 AFP 命令
- 与命令相关联的算法：用于指定 AFP 命令执行的操作

3.3 SMB

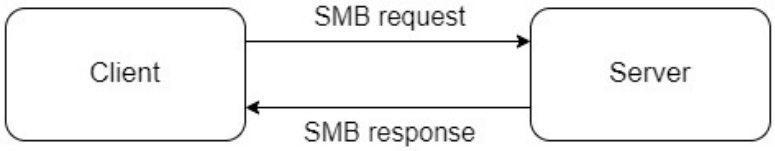

图 5: SMB 概念图

Server Message Block（SMB）是一种采用 TCP/IP 协议作为底层传输协议的通信协议，旨在提供经管身份验证的 IPC 机制并支持计算机网络上的文件共享、打印机共享、网络浏览和进程间管道通信等功能。SMB 是 Microsoft 分布式文件系统实现的基础。

3.3.1 实现方案

在传输实现方面，SMB 常有两种实现方式：

- 直接运行在 TCP 端口 445 上
- 最初运行在支持 IEEE 802.2 和 IPX/SPX 协议的 NetBIOS 上，后来运行在支持 TCP/IP 协议的 NetBIOS 上（NetBT），TCP/IP 协议的引入使复杂互联网络（尤其是 Internet）上的数据传输成为可能。NetBT 上 SMB 的服务器组件使用三个 TCP 或 UDP 端口：
  - 137：NetBIOS 名称服务
  - 138：NetBIOS 数据报服务
  - 139：NetBIOS 段服务

在 Windows 中，ID 为 LanmanServer 的服务和 ID 为 LanmanWorkstation 的服务共同实现了 SMB，其中，前者负责服务共享资源，后者负责维护计算机名称并帮助访问其他计算机上的共享资源。在安全方面，SMB 通常采用 Kerberos 协议来对用户身份进行核对（但在简单对等网络上往往使用 NTLM 协议进行身份验证）。

3.3.2 Opportunistic Locking

SMB 的一大特点就是支持 Opportunistic Locking。Opportunistic Locking 旨在通过控制客户端对网络文件的缓存来提高性能。在传统意义下，对于文件的控制可以通过文件锁来实现，但是 OpLock 并不是严格意义上的文件锁因为它并不是被用来提供文件访问的严格互斥特性的，具体来讲，OpLock 常有下面四种类型：

- Batch Lock：最初是为了支持 DOS 批处理文件执行操作的特定行为而创建的，当某文件在短时间内打开和关闭多次时，性能消耗过大。为了解决这个问题，可以创建一个 Batch Lock，从而使客户端延迟发送文件关闭请求，如果后续存在文件打开请求，则两个请求相互取消
- Level-1 OpLock / Exclusive Lock：当应用程序以“共享模式”打开存储在 SMB 服务器、但未被任何其他进程（或其他客户端）打开的文件时，客户端会从服务器接收一个 Exclusive Lock。这意味着客户端现在可以假设它是唯一可以访问此特定文件的进程，并且客户端可以缓存对文件的所有更新，并在提交文件到服务器时再进行更新。通过 Exclusive Lock，文件读取和写入文件所需的往返次数更少。如果另一个客户端或进程试图打开同一个文件，服务器便会向客户端发送一条消息（称为 break 或 revocation），这会使先前提供给客户端的 Exclusive Lock 无效，然后客户端将刷新对文件的所有更改
- Level-2 OpLock：对于共享文件而言，如果客户端持有 Exclusive Lock，但是第三方打开该文件，则客户端必须放弃其 Exclusive Lock 以允许其他客户端的读写访问。这时，客户端可能会从服务器接收一个 Level-2 OpLock，该锁允许缓存读取请求，但不允许写入缓存
- Filter OpLock：该锁是在 Windows NT 4.0 中新加入的锁，类似于 Level-2 OpLock，但可以有效防止文件打开和锁定接收之间的共享模式冲突

3.3.3 性能表现

SMB 协议的实际性能表现通常与网络上广播流量有强联系，广播流量的增加通常会导致问题。注意，虽然 SMB 协议本身并不使用广播来实现服务，但是 Windows NT 4.0 服务器常常使用 NetBIOS 协议来定期广播特定主机上可用的服务来发挥作用。在主机数量较少的网络中这不会导致大的问题，但随着网络上主机数量的不断增加，一些由过多的广播流量导致的问题便会出现。

此外，网络设计人员还发现延迟对 SMB 1.0 协议的性能有重大影响，使得它的性能比 FTP 等其他协议更差。例如，Internet 上的 VPN 连接通常会引入网络延迟，这是因为 SMB 1.0 是块协议而非流协议，最初仅是为小型 LAN 设计的，它的块大小限制为 64K，SMB 签名会产生额外的开销，并且 TCP 窗口大小未针对 WAN 链接进行优化。为了解决这个问题，网络研究人员和设计人员逐渐发展出了 SMB 2.0 协议、TCP 窗口缩放等技术，各种网络供应商也开始提供支持优化后的 SMB 1.0 协议和 SMB 2.0 协议的 WAN 设备等。

3.3.4 发展历史

Barry Feigenbaum 最初于 1983 年初在 IBM 设计了 SMB，旨在将本地文件访问转变为网络文件系统，微软随后在 LAN Manager 操作系统中实现了 SMB 协议，这就是 SMB 1.0 协议。SMB 1.0 协议最初设计为在基于 IEEE 802.2 的 NetBIOS 帧上运行。从 Windows 2000 开始，SMB 使用 TCP 端口 445 在 TCP 上运行，该功能称为“直接主机 SMB”。

微软在 2006 年推出了新版本的协议，即 SMB 2.0 协议，该协议用于 Windows Vista 和 Windows Server 2008。SMB 2.0 通过将命令和子命令的数量从一百多个减少到十九个，从而大幅优化了性能。并且，它支持流水线机制，从而提高了高延迟链路的性能。SMB 2.0 还包括对符号链接的支持。其他改进包括文件属性的缓存、使用 HMAC SHA-256 散列算法改进的消息签名以及通过增加每台服务器的用户、共享和打开文件的数量等来实现更好的可扩展性。SMB 2.0 使用 32 位或 64 位宽的存储字段，并在文件句柄的情况下允许使用 128 位，从而消除了 SMB 1.0 对块大小的限制，提高了通过快速网络传输大文件的性能。

SMB 2.1 随 Windows 7 和 Windows Server 2008 R2 发布，通过支持弹性句柄和多协议版本的支持协商机制引入了较小的性能增强。

SMB 3.0（以前称为 SMB 2.2）是随 Windows 8 和 Windows Server 2012 发布的，它带来了一些非常重要的新特性，主要包括：

- SMB 透明故障转移：让管理员可执行群集文件服务器中节点的硬件或软件维护，且不会中断将数据存储在这些文件共享上的服务器应用程序。此外，如果群集节点出现硬件或软件故障，SMB 客户端将以透明方式重新连接到其他群集节点，且不会中断将数据存储在这些文件共享上的服务器应用程序。即客户端能够持续、稳定的对远程文件服务器进行通讯，用户不会感受到单点服务器故障所带来的性能影响
- SMB 横向扩展：可构建横向扩展文件服务器（Scale-Out File Server），在使用群集共享卷（CSV）时，管理员可以通过文件服务器群集中所有节点，创建可供同时访问含直接 I/O 的数据文件的文件共享，这可以更好地利用文件服务器客户端的网络带宽和负载平衡，以及优化服务器应用程序的性能
- SMB 多通道：如果在 SMB 3.0 客户端及服务器之间提供多条路径，则支持网络带宽和网络容错的聚合，提升了网络可用性及文件服务器的稳定性，

并让服务器应用程序可以充分利用可用网络带宽，以及在发生网络故障时快速恢复

- SMB 直接访问（SMB over Remote Direct Memory Access，即 RDMA）：支持使用具有 RDMA 功能且可全速运行的网络适配器，其中延迟非常低且 CPU 利用率极少，对于 Hyper-V 或 Microsoft SQL Server 等实现工作负载，这让远程服务器如同本地存储一般
- 用于服务器应用程序的性能计数器：全新 SMB 性能计数器提供有关吞吐量、延迟和 IOPS 的按共享列出的详细信息，从而让管理员可以分析用于存储数据的 SMB 3.0 文件共享的性能，这些计数器是为了将文件存储在远程文件共享上的服务器应用程序而设计的，如 Hyper-V 和 SQL Server
- 性能优化：SMB 3.0 客户端和 SMB 3.0 服务器均已针对小型随机读、写和 I/O 操作进行了优化，此外，默认情况下支持大型最大传输单元（MTU），这将大幅提高大型连续传输性能，如 SQL Server 数据仓库、数据库备份或还原、部署或复制虚拟硬盘
- SMB 加密：提供 SMB 数据的端对端加密并防止数据在未受信任网络中遭受窃听。无需新部署成本，且无需 Internet 协议安全性（IPsec）、专用硬件或 WAN 加速器，使用 AES CCM 128 位加密算法，它可按共享配置，也可针对整个文件服务器配置，并且可针对数据遍历未受信任网络的各种方案启动

SMB 3.0.2 是随 Windows 8.1 和 Windows Server 2012 R2 发布的，在该版本和更高版本中，可以选择禁用早期的 SMB 1.0 以提高安全性。并且，该版本的 SMB 支持客户端直读直写请求，且对 RDMA 进行了性能改进。

SMB 3.1.1 是随 Windows 10 和 Windows Server 2016 发布的。该版本除了支持 SMB 3.0 中添加的 AES CCM 128 加密外，还支持 AES GCM 128 加密，并使用 SHA-512 哈希实现预认证完整性检查。当连接到使用 SMB 2.0 及更高版本的客户端时，SMB 3.1.1 还强制进行安全协商。

SMB 具有一些第三方的实现，包括 Samba、Netsmb、NQ、MoSMB、Fusion File Share by Tuxera、Likewise 和 CIFSD 等。

3.4 FTP 和 SFTP

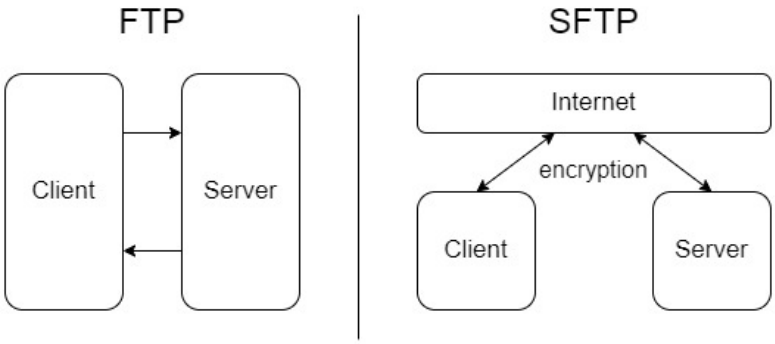

图 6: FTP 和 SFTP 概念图

3.4.1 FTP 概要

File Transfer Protocol（FTP）是一种标准通信协议，它工作在 OSI 模型的第七层，TCP 模型的第四层，用于将计算机文件从服务器传输到计算机网络上的客户端。FTP 建立在客户端-服务器（C/S）模型架构之上，在客户端和服务器之间使用单独的控制和数据连接，且在建立连接前要经过一个“三次握手”的过程，保证客户与服务器之间的连接是可靠的，而且是面向连接，为数据传输提供可靠保证。FTP 用户可以使用明文登录协议对自己进行身份验证，通常采用用户名和密码的形式，但如果服务器配置允许，则可以匿名连接。为了保护用户名和密码并加密内容的安全传输，FTP 通常使用 SSL/TLS（FTPS）或替换为 SSH 文件传输协议（即 SFTP）。

FTP 允许用户以文件操作的方式（如文件的增、删、改、查、传送等）与另一主机相互通信。然而，用户并不真正登录到自己想要存取的计算机上面而成为完全用户，可用 FTP 程序访问远程资源，实现用户往返传输文件、目录管理以及访问电子邮件等等，即使双方计算机可能配有不同的操作系统和文件存储方式。

FTP 采用 Internet 标准文件传输协议 FTP 的用户界面，向用户提供了一组用来管理计算机之间文件传输的应用程序。FTP 是基于 C/S 模型而设计的，在客户端与 FTP 服务器之间建立两个连接。开发任何基于 FTP 的客户端软件都必须遵循 FTP 的工作原理，FTP 的独特的优势同时也是与其它客户服务器程序最大的不同点就在于它在两台通信的主机之间使用了两条 TCP 连接，一条是数据连接，用于数据传送；另一条是控制连接，用于传送控制信息（命令和响应），这种将命令和数据分开传送的思想大大提高了 FTP 的效率，而其它客户服务器应用程序一般只有一条 TCP 连接。在 FTP 中，客户有三个构件：用户接口、客户控制进程和客户数据传送进程。服务器有两个构件：服务器控制进程和服务器数据传送进程。在整个交互的 FTP 会话中，控制连接始终是处于连接状态的，数据连接则在每一次文件传送时先打开后关闭。

3.4.2 FTP 使用模式

FTP 有两种使用模式：

- 主动模式（又称 Standard 方式、PORT 方式）：要求客户端和服务器端同时打开并且监听一个端口以创建连接，在这种情况下，客户端由于安装了防火墙所以可能会产生一些问题
- 被动模式（又称 Passive 方式、PASV 方式）：只要求服务器端产生一个监听相应端口的进程，这样就可以绕过客户端安装了防火墙的问题

在主动模式下，FTP 连接创建需要遵循以下步骤：

1. 客户端打开一个随机的端口（端口号大于 1024，比如 x），同时一个 FTP 进程连接至服务器的 21 号命令端口。此时，该 TCP 连接的来源地端口为客户端指定的随机端口 x，目的地端口（远程端口）为服务器上的 21 号端口
2. 客户端开始监听端口（x+1），同时向服务器发送一个端口命令（通过服务器的 21 号命令端口），此命令告诉服务器客户端正在监听的端口号并且已准备好从此端口接收数据。这个端口就是我们所知的数据端口
3. 服务器打开 20 号源端口并且创建和客户端数据端口的连接。此时，来源地的端口为 20，远程数据（目的地）端口为（x+1）
4. 客户端通过本地的数据端口创建一个和服务器 20 号端口的连接，然后向服务器发送一个应答，告诉服务器它已经创建好了一个连接

3.4.3 FTP 传输方式

FTP 的传输有以下两种方式：

- ASCII 传输方式：假定用户正在拷贝的文件包含的简单 ASCII 码文本，如果在远程机器上运行的不是 UNIX，当文件传输时，FTP 通常会自动地调整文件的内容以便于把文件解释成另外那台计算机存储文本文件的格式。但是如果用户正在传输的文件包含的不是文本文件，而是程序，数据库，字处理文件或压缩文件等文件的时候，用户需要在拷贝任何非文本文件之前，用 binary 命令告诉 FTP 逐字拷贝
- 二进制传输模式：在二进制传输中，保存文件的位序，以便原始和拷贝的是逐位一一对应的，即使目的地机器上包含位序列的文件是没意义的

3.4.4 FTP 支持命令

FTP 支持的命令如下表所示：

命令	RFC	描述
ABOR		Abort an active file transfer
ACCT		Account information
ADAT	RFC 2228	Authentication/Security Data
ALLO		Allocate sufficient disk space to receive a file
APPE		Append (with create)
AUTH	RFC 2228	Authentication/Security Mechanism
AVBL		Streamlined FTP Command Extensions Get the available space
CCC	RFC 2228	Clear Command Channel
CDUP		Change to Parent Directory
CONF	RFC 2228	Confidentiality Protection Command
CSID		Streamlined FTP Command Extensions C/S Identification
CWD	RFC 697	Change working directory
DELE		Delete file
DSIZ		Streamlined FTP Command Extensions Get the directory size
ENC	RFC 2228	Privacy Protected Channel
EPRT	RFC 2428	Specifies an extended address and port to which the server should connect
EPSV	RFC 2428	Enter extended passive mode
FEAT	RFC 2389	Get the feature list implemented by the server
HELP		Returns usage documentation on a command if specified, else a general help document is returned
HOST	RFC 7151	Identify desired virtual host on server, by name
LANG	RFC 2640	Language Negotiation
LIST		Returns information of a file or directory if specified, else information of the current working directory is returned
LPRT	RFC 1639	Specifies a long address and port to which the server should connect
LPSV	RFC 1639	Enter long passive mode
MDTM	RFC 3659	Return the last-modified time of a specified file
MFCT	The 'MFMT', 'MFCT', and 'MFF' Command Extensions for FTP	Modify the creation time of a file
MFF	The 'MFMT', 'MFCT', and 'MFF' Command Extensions for FTP	Modify fact (the last modification time, creation time, UNIX group/owner/mode of a file)

命令	RFC	描述
MFMT	The 'MFMT', 'MFCT', and 'MFF' Command Extensions for FTP	Modify the last modification time of a file
MIC	RFC 2228	Integrity Protected Command
MKD		Make directory
MLSD	RFC 3659	Lists the contents of a directory in a standardized machine-readable format
MLST	RFC 3659	Provides data about exactly the object named on its command line in a standardized machine-readable format
MODE		Sets the transfer mode (Stream, Block, or Compressed)
NLST		Returns a list of file names in a specified directory
NOOP		No operation (dummy packet; used mostly on keepalives)
OPTS	RFC 2389	Select options for a feature (for example OPTS UTF8 ON)
PASS		Authentication password
PASV		Enter passive mode
PBSZ	RFC 2228	Protection Buffer Size
PORT		Specifies an address and port to which the server should connect
PROT	RFC 2228	Data Channel Protection Level
PWD		Print working directory. Returns the current directory of the host
QUIT		Disconnect
REIN		Re initializes the connection
REST	RFC 3659	Restart transfer from the specified point
RETR		Retrieve a copy of the file
RMD		Remove a directory
RMDA	Streamlined FTP Command Extensions	Remove a directory tree
RNFR		Rename from
RNTO		Rename to
SITE		Sends site specific commands to remote server (like SITE IDLE 60 or SITE UMASK 002). Inspect SITE HELP output for complete list of supported commands
SIZE	RFC 3659	Return the size of a file
SMNT		Mount file structure

命令	RFC	描述
SPSV	FTP Extension Allowing IP Forwarding (NATs)	Use single port passive mode (only one TCP port number for both control connections and passive-mode data connections)
STAT		Returns information on the server status, including the status of the current connection
STOR		Accept the data and to store the data as a file at the server site
STOU		Store file uniquely
STRU		Set file transfer structure
SYST		Return system type
THMB	Streamlined FTP Command Extensions	Get a thumbnail of a remote image file
TYPE		Sets the transfer mode (ASCII/Binary)
USER		Authentication username
XCUP	RFC 775	Change to the parent of the current working directory
XMKD	RFC 775	Make a directory
XPWD	RFC 775	Print the current working directory
XRCP	RFC 743	
XRMD	RFC 775	Remove the directory
XRSQ	RFC 743	
XSEM	RFC 737	Send, mail if cannot
XSEN	RFC 737	Send to terminal

3.4.5 SFTP

SSH File Transfer Protocol（SFTP）是一种网络协议，该协议可通过任何可靠的数据流提供文件访问、文件传输和文件管理。该协议假定在安全通道（例如SSH）上运行，服务器已经对客户端进行了身份验证，并且客户端用户的身份可用于协议。

与早期只允许文件传输的 SCP 协议相比，SFTP 协议允许对远程文件进行一系列操作，使其更像远程文件系统协议。SFTP 客户端的额外功能包括恢复中断的传输、目录列表和远程文件删除。SFTP 尝试比 SCP 更独立于平台，例如，SCP 客户端指定的通配符扩展由服务器决定，而 SFTP 的设计则避免了这个问题。而且 SCP 通常在 UNIX 平台上实现，SFTP 服务器则在大多数平台上都可用。但是 SFTP 协议的来回特性，与 SFTP 协议相比，SCP 中的文件传输速度更快。与 FTP 相比，通过 SFTP 上传的文件可能与其基本属性相关联，例如时间戳，这是 SFTP 优于普通 FTP 协议的优势。

现在来简要介绍 SFTP 技术的发展历程。1995 年，芬兰赫尔辛基科技大学的研究员 Tatu 设计了第一个版本的协议（现称为 SSH-1）。SSH 的目标是取代早期

既不提供强认证，也不保证机密性的 rlogin、TELNET、FTP 和 RSH 协议。1995 年 7 月时 Tatu 以免费软件的方式发布了该版本的协议，受到了广泛欢迎，到 1995 年底，SSH 用户群已在 50 个国家普及，并增加到 20000 个用户。

1995 年 12 月，Tatu 创建了 SSH Communications Security 以推广和开发 SSH。SSH 软件的原始版本使用了各种免费软件，例如 GNU libgmp，但 SSH Communications Security 发布的更新版本逐渐发展成为专有的软件。

“Secsh” 是负责 SSH 协议版本 2 的 IETF 工作组的官方 Internet 工程任务组（IETF）名称。2006 年，该协议的修订版本 SSH-2 被采纳为标准。此版本与 SSH-1 不兼容。SSH-2 具有 SSH-1 的安全性，并且有其他功能改进。例如，更好的安全性来自 Diffie-Hellman 密钥交换和通过消息认证码进行的强完整性检查。SSH-2 的新功能包括通过单个 SSH 连接运行任意数量的 shell 会话的功能。由于 SSH-2 在 SSH-1 上的优越性和普及性，一些实现如 libssh 等仅支持 SSH-2 协议。

但是，随着开发工作的进展，Secsh 文件传输项目的范围扩大到包括文件访问和文件管理。最终，由于一些委员会成员开始将 SFTP 视为文件系统，而不仅仅是文件访问或文件传输协议，这显然超出了工作组的权限，开发工作也因此中断。2013 年，在中断了七年之后，小组开始尝试以版本 3 草案的编制和修订作为主要任务而重新启动 SFTP 工作，并陆续发布了版本 3、版本 4、版本 5 和版本 6 的协议。

3.5 HTTP 和 WebDAV

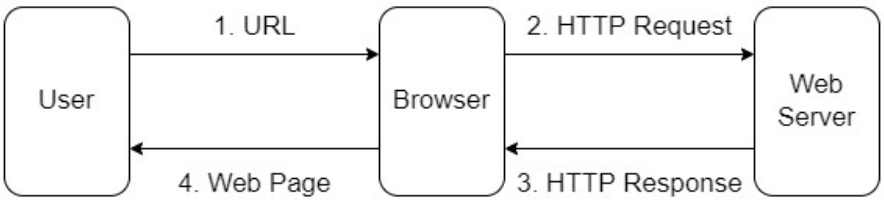

图 7: HTTP 概念图

3.5.1 HTTP 概要

Hypertext Transfer Protocol（HTTP）是一个简单的请求-响应协议，它作为 Internet 协议套件模型中的一个应用层协议，主要用于分布式、协作、超媒体信息系统，且通常运行在 TCP 上。它指定了客户端可能发送给服务器什么样的消息以及得到什么样的响应。请求和响应消息的头以 ASCII 形式给出；而消息内容则具有一个类似 MIME 的格式。这个简单模型使开发和部署 Web 应用变得非常简单，极大促进了早期 Web 的发展。

HTTP 的开发由 CERN 于 1989 年开始，并在一份最初的文档中进行了总结，该文档描述了使用第一个名为 0.9 的 HTTP 协议版本的客户端和服务器的行为。在 1990 年，HTTP 成为了 WWW 的支撑协议。当时 WWW 之父 Berners-Lee 倡

议成立 WWW 联盟，并组织了 IETF 小组进一步完善和发布 HTTP。HTTP 是应用层协议，是为了实现某一类具体应用的协议，并由某一运行在用户空间的应用程序来实现其功能。

HTTP 是基于 B/S 架构进行通信的，而 HTTP 的服务器端实现程序有 httpd、nginx 等，其客户端的实现程序主要是各种 Web 浏览器。Web 服务是基于 TCP 的，因此为了能够随时响应客户端的请求，Web 服务器需要监听 TCP 端口 80。这样客户端浏览器和 Web 服务器之间就可以通过 HTTP 协议进行通信了。

3.5.2 HTTP 发展历程

1991 年，第一个文档化的 HTTP 官方版本正式发布，这个版本被命名为 HTTP/0.9。HTTP/0.9 仅支持 GET 方法，允许客户端仅从服务器检索 HTML 文档，不支持任何其他文件格式或信息上传。这个版本的 HTTP 协议无法进行内容的协商，并强制规定了双方发送的内容是什么，例如，图片是无法显示和处理的。

到了 1.0 协议阶段，Berners-Lee 提出了 HTTP/1.0。在此后的不断丰富和发展中，HTTP/1.0 成为最重要的面向事务的应用层协议。该协议对每一次请求或响应建立并拆除一次连接。其特点是简单、易于管理，并且解决了当时人们对于部署和使用 HTTP 协议的主要问题，所以它迅速得到了广泛的应用。1996 年 5 月，RFC 1945 作为 HTTP/1.0 的最终修订版发布。

在 1.0 协议中，双方规定了连接方式和连接类型，这极大扩展了 HTTP 的应用领域，但对于互联网最重要的速度和效率，则并没有太多的考虑。毕竟，作为协议的制定者，当时也没有想到 HTTP 会有那么快的普及速度。随着普及程度的不断加深，在 1997 年 1 月，万众瞩目的 RFC 2068 作为 HTTP/1.1 规范正式发布。

2012 年左右，受到谷歌公司开发的一种名为 SPDY 的新 HTTP 二进制协议的启发，HTTP 工作组宣布会考虑到 SPDY 的想法和成果，编修新的 HTTP/2 协议，并完成对 HTTP/1.1 规范的修订。2015 年 5 月，HTTP/2 发布为 RFC 7540，该版本的协议一经发布，便迅速被所有已经支持 SPDY 的 Web 浏览器所采用。

最近，HTTP/3 初稿已在 2020 年发布，主流网络浏览器和网络服务器开始采用它。并且，2022 年 6 月 6 日，IETF 将 HTTP/3 标准化为 RFC 9114。

3.5.3 HTTP 工作原理

HTTP 是基于客户/服务器模式，且面向连接的。典型的 HTTP 事务处理有如下的过程：

1. 客户与服务器建立连接：客户与服务器之间的 HTTP 连接是一种一次性连接，它限制每次连接只处理一个请求，当服务器返回本次请求的应答后便立即关闭连接，下次请求再重新建立连接。这种一次性连接主要考虑到 WWW 服务器面向的是 Internet 中成千上万个用户，且只能提供有限个连

接，故服务器不会让一个连接处于等待状态，及时地释放连接可以大大提高服务器的执行效率

2. 客户向服务器提出请求：HTTP 规范定义了 9 种请求方法（如下表所示），每种请求方法规定了客户和服务器之间不同的信息交换方式，服务器将根据客户请求完成相应操作，并以应答块形式返回给客户，最后关闭连接

Request method	**RFC**	**Request has payload body**	**Response has payload body**	**Safe**	**Idempotent**	**Cacheable**
GET	RFC 7231	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231	Optional	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	Optional	Yes	No	Yes	No
CONNECT	RFC 7231	Optional	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	No

3. 服务器接受请求，并根据请求返回相应的文件作为应答

4. 客户与服务器关闭连接

HTTP 是一种无状态协议，即服务器不保留与客户交易时的任何状态。这大大减轻了服务器记忆负担，从而保持较快的响应速度。HTTP 又是一种面向对象的协议，允许传送任意类型的数据对象。它通过数据类型和长度来标识所传送的数据内容和大小，并允许对数据进行压缩传送。当用户在一个 HTML 文档中定义了一个超文本链后，浏览器将通过 TCP/IP 协议与指定的服务器建立连接。

HTTP 支持持久连接，在 HTTP/0.9 和 1.0 中，连接在单个请求/响应对之后关闭。在 HTTP/1.1 中，引入了保持活动机制，其中连接可以重用于多个请求。这样的持久性连接可以明显减少请求延迟，因为在发送第一个请求之后，客户端不需要重新协商 TCP 连接。

从技术上讲是客户在一个特定的 TCP 端口（端口号一般为 80）上打开一个套接字。如果服务器一直在这个周知的端口上倾听连接，则该连接便会建立起来。然后客户通过该连接发送一个包含请求方法的请求块。接下来，服务器在收到请求之后便进行解析，之后将结果发回到请求该结果的客户机上。

3.5.4 HTTP 标头字段

HTTP 标头字段是客户端程序和服务器在每个 HTTP 请求和响应中发送和接收的字符串列表。这些标头通常对最终用户不可见，仅由服务器和客户端应用程

序处理或记录。它们定义了如何对通过连接发送/接收的信息进行编码、客户端的会话验证和识别或其匿名性、服务器应该如何处理数据、驻留时间和正在下载的文档等。

一般来讲，在标准请求字段中包括以下内容：

Name	Example	Status	Standard
A-IM	A-IM: feed	Permanent	RFC 3229
Accept	Accept: text/html	Permanent	RFC 2616, 7231
Accept-Charset	Accept-Charset: utf-8	Permanent	RFC 2616
Accept-Datetime	Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT	Provisional	RFC 7089
Accept-Encoding	Accept-Encoding: gzip, deflate	Permanent	RFC 2616, 7231
Accept-Language	Accept-Language: en-US	Permanent	RFC 2616, 4021, 7231
Access-Control-Request-Method, Access-Control-Request-Headers	Access-Control-Request-Method: GET	Permanent: standard	
Authorization	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==	Permanent	RFC 2616, 7123, 7235
Cache-Control	Cache-Control: no-cache	Permanent	RFC 2616, 7231, 7234
Connection	Connection: keep-alive	Permanent	RFC 2616, 7230
Content-Encoding	Content-Encoding: gzip	Permanent	RFC 2616, 7231
Content-Length	Content-Length: 348	Permanent	RFC 2616, 7230, 7231
Content-MD5	Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==	Obsolete	RFC 1544, 1864, 4021
Content-Type	Content-Type: application/x-www-form-urlencoded	Permanent	RFC 2616, 4021, 7231
Cookie	Cookie: $Version=1; Skin=new;	Permanent: standard	RFC 2965, 6265

Name	Example	Status	Standard
Date	Date: Tue, 15 Nov 1994 08:12:31 GMT	Permanent	RFC 2616, 5322, 5536, 7231
Expect	Expect: 100-continue	Permanent	RFC 2616, 7231
Forwarded	Forwarded: for=192.0.2.60;proto=http	Permanent	RFC 7239
From	From: user@example.com	Permanent	RFC 2616, 5322, 5536, 6854, 7231
Host	Host: en.wikipedia.org:8080		
HTTP2-Settings	HTTP2-Settings: token64	Permanent: standard	
If-Match	If-Match: ”737060cd8c284d8af79582d”	Permanent	RFC 2616, 7323
If-Modified-Since	If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT	Permanent	RFC 2616, 7323
If-None-Match	If-None-Match: ”737060cd8c2d3082f209582d”	Permanent	RFC 2616, 7232
If-Range	If-Range: ”737060cd8c284d8af709582d”	Permanent	RFC 2616, 7232, 7233
If-Unmodified-Since	If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT	Permanent	RFC 2616, 7232
Max-Forwards	Max-Forwards: 10	Permanent	RFC 2616, 7231
Origin	Origin: http://www.example-social-network.com	Permanent: standard	RFC 6454
Pragma	Pragma: no-cache	Permanent	RFC 2616, 7234
Prefer	Prefer: return=representation	Permanent	RFC 7240
Proxy-Authorization	Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==	Permanent	RFC 2616, 7235
Range	Range: bytes=500-999	Permanent	RFC 2616, 7233
Referer	Referer: http://en.wikipedia.org/wiki	Permanent	RFC 2636, 7231
TE	TE: trailers, deflate	Permanent	RFC 2616, 7230
Trailer	Trailer: Max-Forwards	Permanent	RFC 2616, 7230

Name	Example	Status	Standard
Transfer-Encoding	Transfer-Encoding: chunked	Permanent	RFC 2616, 7230
User-Agent	User-Agent: Mozilla/5.0	Permanent	RFC 2616, 5536, 7231
Upgrade	Upgrade: h2c, HTTPS/1.3, IRC/6.9, RTA/x11, websocket	Permanent	RFC 2616, 7230
Via	Via: 1.0 fred, 1.1 example.com	Permanent	RFC 2616, 7230
Warning	Warning: 199 Miscellaneous warning	Permanent	RFC 2616, 7234

而在响应字段中，通常包括：

Name	Example	Status	Standard
Accept-CH	Accept-CH: UA, Platform	Experimental	RFC 8942
Access-Control-Allow-Origin, etc.	Access-Control-Allow-Origin: *	Permanent: standard	RFC 2616, 7480
Accept-Patch	Accept-Patch: text/example; charset=utf-8	Permanent	RFC 5789
Accept-Ranges	Accept-Ranges: bytes	Permanent	RFC 2616, 7233
Age	Age: 12	Permanent	RFC 2616, 7234
Allow	Allow: GET, HEAD	Permanent	RFC 2616, 7231
Alt-Svc	Alt-Svc: http/1.1=”http2.example.com:8001”; ma=7200	Permanent: standard	
Cache-Control	Cache-Control: max-age=3600	Permanent	RFC 2616, 7231, 7234
Connection	Connection: close	Permanent	RFC 2616, 7230
Content-Disposition	Content-Disposition: attachment; file-name=”fname.ext”	Permanent	RFC 2616, 4021, 6266
Content-Encoding	Content-Encoding: gzip	Permanent	RFC 2616, 7231
Content-Language	Content-Language: da	Permanent	RFC 2616, 4021, 7231
Content-Length	Content-Length: 348	Permanent	RFC 2616, 7230, 7231

Name	Example	Status	Standard
Content-Location	Content-Location: /index.htm	Permanent	RFC 2616, 4021, 7231
Content-MD5	Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==	Obsolete	RFC 1544, 1864, 4021
Content-Range	Content-Range: bytes 21010-47021/47022	Permanent	RFC 2616, 7233
Content-Type	Content-Type: text/html; charset=utf-8	Permanent	RFC 2616, 4021, 7231
Date	Date: Tue, 15 Nov 1994 08:12:31 GMT	Permanent	RFC 2616, 5322, 5536, 7231
Delta-Base	Delta-Base: ”abc”	Permanent	RFC 3229
ETag	ETag: ”737060cd8c284d8af7ad3082f209582d”	Permanent	RFC 2616, 7232
Expires	Expires: Thu, 01 Dec 1994 16:00:00 GMT	Permanent: standard	RFC 2616, 4021, 5536, 7234
IM	IM: feed	Permanent	RFC 3229
Last-Modified	Last-Modified: Tue, 15 Nov 1994 12:45:26 GM	Permanent	RFC 2616, 7232
Link	Link: </feed>; rel=”alternate”	Permanent	RFC 5988
Location	Location: /pub/WWW/People.html	Permanent	RFC 2068, 2616, 7231
P3P	P3P: CP=”This is not a P3P policy!”	Permanent	
Pragma	Pragma: no-cache	Permanent	RFC 2616, 7234
Preference-Applied	Preference-Applied: return=representation	Permanent	RFC 7240
Proxy-Authenticate	Proxy-Authenticate: Basic	Permanent	RFC 2068, 2616, 7235
Public-Key-Pins	Public-Key-Pins: max-age=2592000	Permanent	RFC 7469
Retry-After	Retry-After: 120	Permanent	RFC 2616, 7231
Server	Server: Apache/2.4.1 (Unix)	Permanent	RFC 2068, 2616, 7231
Set-Cookie	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1	Permanent: standard	RFC 6265

Name	Example	Status	Standard
Strict-Transport-Security	Strict-Transport-Security: max-age=16070400; includeSubDomains	Permanent: standard	
Trailer	Trailer: Max-Forwards	Permanent	RFC 2616, 7230
Transfer-Encoding	Transfer-Encoding: chunked	Permanent	RFC 2616, 7230
Tk	Tk: ?	Permanent	
Upgrade	Upgrade: h2c, HTTPS/1.3, IRC/6.9, RTA/x11, websocket	Permanent	RFC 2616, 7230
Vary	Vary: Accept-Language	Permanent	RFC 2068, 2616, 7231
Via	Via: 1.0 fred, 1.1 example.com (Apache/1.1)	Permanent	RFC 2616, 7230
Warning	Warning: 199 Miscellaneous warning	Permanent	RFC 2616, 7234
WWW-Authenticate	WWW-Authenticate: Basic	Permanent	RFC 2068, 2616, 7235
X-Frame-Options	X-Frame-Options: deny	Obsolete	

除了以上说明的内容以外，在请求字段响应字段中还包括若干非标准字段。

3.5.5 WebDAV

Web Distributed Authoring and Versioning（WebDAV）是 HTTP 的一组扩展，在 GET、POST、HEAD 等几个 HTTP 标准方法以外添加了一些新的方法，从而允许用户代理直接在 HTTP Web 服务器中协作创作内容，从而将 Web 视为可写的、支持协作的媒介，而不仅仅是只读媒介。WebDAV 在 RFC 4918 中定义。

WebDAV 协议为用户在服务器上创建、更改和移动文档提供了一个框架，支持写文件锁定及解锁，还可以支持文件的版本控制等功能。最重要的功能包括维护有关作者或修改日期的属性、命名空间管理、集合和覆盖保护。属性维护包括文件信息的创建、删除和查询等。命名空间管理处理在服务器命名空间内复制和移动网页的能力。集合处理各种资源的创建、删除和列表。最后，覆盖保护处理与文件锁定相关的方面。它利用了现有技术，例如传输层安全、摘要访问身份验证或 XML 来满足这些要求。许多现代操作系统为 WebDAV 提供内置客户端支持。

WebDAV 扩展了请求方法允许的标准 HTTP 动词和标头集，增加的动词包括：

- COPY：将资源从一个统一资源标识符（URI）复制到另一个
- LOCK：对资源加锁，WebDAV 支持共享锁和排他锁
- MKCOL：创建集合（也称为目录）
- MOVE：将资源从一个 URI 移动到另一个
- PROPFIND：从 Web 资源中检索存储为 XML 的属性。它也被重载以允许检索远程系统的集合结构（也称为目录层次结构）
- PROPPATCH：在单个原子动作中更改和删除资源上的多个属性
- UNLOCK：从资源中移除锁

常见的 WebDAV 客户端如下图所示：

Client	**Creator**	**Operating system support**	**License**	**Interface**
Cyderduck	David V. Kocher	Windows, OS X	GPL	GUI
davfs2	GNOME team	FUSE	GPL	VFS
davix	CERN	Windows, Linux, OS X	LGPL	CLI
GVfs	GNOME team	GNOME	GPL	VFS
KIO	KDE team	KDE	GPL	VFS
Konqueror	KDE team	KDE	GPL	GUI
GNOME Files	GNOME team	GNOME	GPL	GUI
SmartFTP	SmartSoft Ltd	Windows	Proprietary	GUI
WebDrive	South River Technologies	Windows, OS X, iOS, Droid	Proprietary	VFS
WinSCP	Martin Přikryl	Windows	GPL	VLI and GUI

3.6 NFS

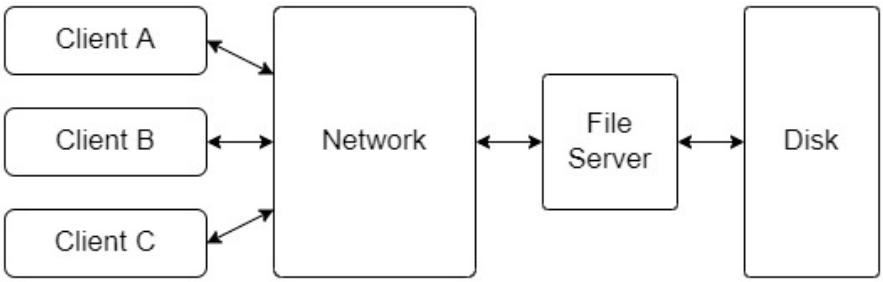

图 8: NFS 概念图

Network File System（NFS）是一种分布式文件系统协议，最初由 Sun 在 1984 年开发，允许客户端计算机上的用户通过计算机网络访问文件，就像访问本地存储一样。NFS 可以认为是文件系统上的一个网络抽象，来允许远程客户端以与本地文件系统类似的方式，来通过网络进行访问。NFS 允许在多个用户之间共享公共文件系统，并提供数据集中的优势，来最小化所需的存储空间。

NFS 具有下面的特点：

- 提供透明文件访问以及文件传输
- 容易扩充新的资源或软件，不需要改变现有的工作环境
- 高性能，可灵活配置

NFS 与许多其他协议一样，建立在开放网络计算远程过程调用（ONC RPC）系统上。其工作原理是使用 C/S 架构，由客户端程序和服务器程序组成。服务器程序向其他计算机提供对文件系统的访问，其过程称为输出。NFS 客户端程序对共享文件系统进行访问时，把它们从 NFS 服务器中“输出”出来。

具体来讲，一旦发现了为 NFS 所指定的需求，VFS 会将其传递给内核中的 NFS 实例。NFS 解释 I/O 请求并将其翻译为 NFS 程序（OPEN、ACCESS、CREATE、READ、CLOSE、REMOVE 等）。一旦从 I/O 请求中选择了程序，它会在远程程序调用层中执行。RPC 提供了在系统间执行程序调用的方法，它将封装 NFS 请求，并伴有参数，将它们发送到合适的远程对等级，然后管理并追踪响应，提供给合适的请求者。在服务器端，NFS 以相似的风格运行。需求到达网络协议栈，通过 RPC/XDR 然后到达 NFS 服务器。NFS 服务器负责满足需求。需求向上提交给 NFS 守护进程，它为需求标示出目标文件系统树，并且 VFS 再次用于在本地存储中获取文件系统。NFS 传输协议用于服务器和客户机之间文件访问和共享的通信，从而使客户机远程地访问保存在存储设备上的数据。

3.6.1 NFSv2

NFS 协议的第 2 版（在 1989 年 3 月的 RFC 1094 中定义）最初仅通过用户数据报协议 UDP 运行，它的设计者旨在保持服务器端无状态，并在核心协议之外实现锁定。虚拟文件系统接口允许模块化实现，在一个简单的协议中定义。由于 32 位限制，NFSv2 仅允许读取文件的前 2GB。

3.6.2 NFSv3

NFSv3 主要动机是减轻 NFSv2 中同步写入操作的性能问题，NFSv3（在 1995 年 6 月的 RFC 1813 中定义）添加了如下新特性：

- 支持 64 位文件大小和偏移量，以处理大于 2GB 的文件
- 支持服务器异步写入，提高写入性能

- 将 TCP 作为传输协议
- 许多回复中的附加文件属性，以避免重新获取它们的需要
- 支持 READDIRPLUS 操作，用于在扫描目录时获取文件句柄、属性和文件名

NFSv3 协议比以前的版本具有更好的可扩展性，其上述特点为文件系统在更广泛的网络中使用铺平了道路。

3.6.3 NFSv4

NFSv4（在 2000 年 12 月的 RFC 3010 中定义；在 2003 年 4 月的 RFC 3530 和 2015 年 3 月的 RFC 7530 中修订）受 AFS 和 SMB 的影响，进行了性能改进，并增加了对安全性的更好的支持，以及引入有状态的协议（NFS 之前的版本都是无状态的）。

NFSv4.1（在 2010 年 1 月的 RFC 5661 中定义；在 2020 年 8 月的 RFC 8881 中修订）旨在提供协议支持以利用集群服务器部署，包括提供对分布在多个服务器之间的文件的可扩展并行访问的能力（pNFS 扩展）。pNFS 将生态系统拆分为三个部分：客户端、服务器和存储。pNFS 将数据布局与数据本身拆分，允许双路径架构（一条路径用于数据，另一条路径用于控制）。当客户想要访问文件时，服务器以布局响应。布局描述了文件到存储设备的映射。当客户端具有布局时，它能够直接访问存储，而不必通过服务器（这实现了更大的灵活性和更优的性能)。当客户端完成文件操作时，它会提交数据（更新）和布局。如果需要，服务器能够请求从客户端返回布局。pNFS 实现了 LayoutGet 协议和 LayoutReturn 协议，从而分别从服务器获取和发布布局，而 LayoutCommit 则将来自客户端的数据提交到存储库，以便于其他用户使用。服务器也采用 LayoutRecall 协议从客户端回调布局。布局跨多个存储设备展开，来支持并行访问和更高的性能。

NFSv4.2（在 2016 年 11 月的 RFC7862 中定义）的新功能包括：服务器端克隆和复制、应用程序 I/O 建议、稀疏文件、空间预留、应用程序数据块（ADB）、以及 pNFS 的两个新操作（LAYOUTERROR 和 LAYOUTSTATS）。

在 NFSv4 之前，存在一定数量的辅助协议用于加载、锁定、和文件管理中的其他元素。NFSv4 将这一流程简化为一个协议，并将对 UDP 协议的支持作为传输协议移除。NFSv4 还集成支持 UNIX 和基于 Windows 的文件访问语义，将本地集成 NFS 扩展到其他操作系统中。并且，NFSv4 相对于其前身的一大优势是仅使用一个 UDP 或 TCP 端口 2049 来运行服务，这简化了跨防火墙协议的使用。

4 结语

NAS 技术是一种连接网络上从而提供数据存储功能的技术，其主要特点包括：文件级存储、以数据为中心、将存储设备与服务器分离、集中管理数据等。

NAS 技术有效地释放了带宽、提高了性能、降低了总拥有成本和后续投资，这使其成本远低于使用服务器存储，但效率却远远高于后者。

在本文中，我简要梳理了 NAS 技术的一些常见内容，感觉收获颇丰，同时又惊叹于计算机工业界实践成果之富，这值得我们持续不断地学习与探索！