# 存储分类

根据服务器类型分为：封闭系统的存储和开放系统的存储。

封闭系统主要指大型机，开发系统指基于Windows、UNIX、Linux等操作系统的服务器；

开发系统的存储分为：内置存储和外挂存储。

外挂存储根据连接的方式分为：直连式存储（Direct-Attached Storage，简称DAS）和网络化存储（Fabric-Attached Storage，简称FAS）。网络化存储根据传输协议又分为：网络接入存储（Network-Attached Storage，简称NAS）和存储区域网络（Storage Area Network，简称SAN）。

# DAS

## 概述

存储设备是通过电缆（通常是SCSI接口电缆）直接到服务器。I/O请求直接发送到存储设备。这种方式是连接单独的或两台小型集群的服务器。

## 特点

DAS的特点是初始费用可能比较低。可是这种连接方式下，对于多个服务器或多台PC的环境，每台PC或服务器单独拥有自己的存储磁盘，容量的再分配困难；对于整个环境下的存储系统管理，工作烦琐而重复，没有集中管理解决方案。

所以整体的管理成本较高。

# NAS

NAS：网络附加存储，network attached storage，带有集中式文件系统功能的磁阵。

1. 文件系统的（逻辑）位置

一般磁阵可以划分出多个LUN供使用者使用，每个使用者必须有自己的文件系统，但是也可以把文件系统的功能从使用者外迁到磁阵上，对外提供统一的用户接口，使用者不用再记录文件和卷上扇区/蔟块的对应关系，该工作由磁阵上的集中式文件系统模块处理。

1. 使用者与磁阵集中式文件系统的交互

底层传输网络TCP/IP协议，上层的应用逻辑为VIFS/NFS，也称为网络文件系统。网络文件系统的磁盘或卷在远程节点（磁阵或远程主机上），且文件系统功能也搬到了远程节点（磁阵或远程主机上），但是SAN只是磁盘或者卷在远程节点上，文件系统功能在本机上。

1. 磁阵和主机都可以是NAS，只要主机有磁盘和文件系统，并且对外提供访问其文件系统的接口（NFA、CIFS）。

NAS的两个物理条件：1）NAS必须可以访问卷或者物理磁盘；

2）NAS必须具有接入网络的能力

|  |  |
| --- | --- |
| 网络文件系统与本地文件系统的区别 | 唯一的区别就是传输方式从主板上的总线变为网络（一旦用户挂载一个网络文件目录到本地，就可以像本地文件系统一样使用网络文件系统） |
| FTP HTTP | 这些文件服务不属于网络文件系统（网络文件系统可以直接访问远端文件，而FTP需要将所有文件复制到本地才可以） |

## NFS

## CIFS

微软定义了一套自己的网络文件系统的规范，CIFS即Internet范围的FS，Linux和UNIX系统使用另外一种方式NFS（Network File System），这些上层协议都是利用TCP/IP协议进行传输的。

网络文件系统的文件系统逻辑不是在本地运行，而是在网络上的其他节点运行，使用者通过外部网络将读写文件的信息传递给运行在远端的文件系统，也就是调用远程的文件系统模块，而不是在本地内存中使用文件系统的API进行。所以网络文件系统又叫做远程调用式文件系统，也就是RPC FS.

相比较于SAN，这种网络文件系统不仅磁盘或者卷在远端节点上，连文件系统也搬到了远程的节点。本地文件系统可以通过主板上的导线访问内存调用其功能，而网络文件系统只能通过网络适配器上链接的网线而不是主板的导线访问远端的文件系统。

CIFS是一个开销非常大的NAS协议

基于NAS的数据访问，客户端并不关心文件存放在磁盘的那些扇区，这些逻辑全部由NAS服务端狐狸，客户端向NAS设备发送的只有各种文件操作请求以及实际的文件流式数据。这种带有集中式文件系统功能的盘阵，叫做网络附加存储（Network Attached Storage）

NAS不一定是盘阵，一台普通的主机也可以做成NAS，只要是有磁盘和文件系统即可。SAN是一个网络上的磁盘，NAS是一个网络上的文件系统。

NAS架构的路径在虚拟目录层和文件系统层通信的时候，用以太网和TCP/IP协议代替了内存，这样做不但增加了大量的CPU指令周期，而且使用低速介质传输。SAN架构中路径比NAS多了一次FC访问，但是FC逻辑发部分都是有是配置卡上的硬件完成，增加不了多少CPU开销，而且FC速度快。所以，如果后端磁盘没有瓶颈，那么除非NAS使用快于内存的网络访问方式和通信，导致其速度永远都不如SAN。

## GFS2

# SAN

SAN：Storage Area Network存储区域网络，这里特指IP-SAN和FC-SAN，也就是给主机提供远程磁盘。

SAN是一个网络上的磁盘，NAS是一个网络上的文件系统。

说明：对于SAN存储，远端磁盘需要做的就是提供一个磁盘就可以了，具体的划分LUN以及构建文件系统，形成可以访问的目录这个不是SAN存储的工作，对于NAS则不一样，它提供的是一个远程的文件系统，所以在LUN基础之上还需要构建文件系统，最后才提供给主机使用。

## IP-SAN

### 概述

IPSAN是在SAN后产生的，SAN默认指FCSAN，以光纤通道构建存储网络，IPSAN则以IP网络构建存储网络。由于FCSAN的高成本使得很多中小规模存储网络不能接受，一些人开始考虑构建基于以太网技术的存储网络。但是在SAN中，传输的指令是 SCSI的读写指令，不是IP数据包。

iSCSI（互联网小型计算机系统接口）是一种在TCP/IP上进行数据块传输的标准。它是由Cisco和IBM两家发起的，并且得到了各大存储厂商的大力支持。iSCSI可以实现在IP网络上运行SCSI协议，使其能够在诸如高速千兆以太网上进行快速的数据存取备份操作。为了与之前基于光纤技术的FCSAN区分开来，这种技术被称为IPSAN。iSCSI继承了两大最传统技术：SCSI和TCP/IP协议。这为iSCSI的发展奠定了坚实的基础。

将iSCSI为代表的以TCP/IP作为传输方式的网络存储系统称为IP-SAN，即基于IP的存储区域网络。这种方式是将服务器和存储设备通过专用的网络连接起来，服务器通过“BlockI/O”发送数据存取请求到存储设备。

最常用的是iSCSI技术，就是把SCSI命令包在TCP/IP 包中传输，即为SCSI over TCP/IP。

### 特点

优点：

利用无所不在的以太网络，一定程度上保护了现有投资。

IP存储超越了地理距离的限制，适合于对关键数据的远程备份。

IP网络技术成熟,不存在互操作性问题IP存储减少了配置、维护、管理的复杂度。IP网络已经被IT业界广泛认可－网络管理软件和服务产品可供使用千兆网的广泛使用大大提高了IP网络的性能万兆网络技术的发展，使IP存储在性能上可以超越FC存储。

## FC-SAN

### 概述

早期的SAN采用的是光纤通道（FC，Fibre Channel）技术，所以，以前的SAN多指采用光纤通道的存储局域网络，业内称为FCSAN。

### 特点

**优点：**

传输带宽高，目前有1,2,4和8Gb/s四种标准，主流的是4和8Gb/s；

性能稳定可靠，技术成熟，是关键应用领域和大规模存储网络的不二选择。

**缺点：**

成本极其高昂，需要光纤交换机和大量的光纤布线；

维护及配置复杂，需要培训完全不同于LAN管理员的专业FC网络管理员。

# 总结

性能上：FC性能最好，单端口可以达到2Gbps的带宽；NAS性能最差，即使采用千兆网络，通常只能30－40MBps；单千兆iSCSI可以达到60－70MBps；万兆网络下，单万兆口的速度可以超过500MBps。

扩展性上：FC和iSCSI采用SAN的架构，扩展性最好，在存储网络中，易于增加用户或增加存储模块。

应用成本上：FC的应用成本最高，需要配套的昂贵的光纤交换机（8端口光纤交换机大约3－4万元）；每个服务器需要配光纤通道卡（光纤通道卡5000－1万元）；以及采用光纤介质；相比之下，iSCSI只需要普通的以太网交换机（8端口千兆交换机1000元）；服务器本身带有千兆网口，即使增加一个网卡也就是500元，采用普通的超五类线就可以，价格可以忽略不计；市场定位：FC占据高端，iSCSI处于中高端，NAS和DAS处于低端。

# 块存储

## 简介

块存储主要是将裸磁盘空间整个映射给主机使用的，就是说例如磁阵里面有5块硬盘（假设每个1GB），可以通过划分逻辑卷、做RAID、或者LVM等种种方式逻辑划分出N个逻辑的硬盘（假设划分完的逻辑盘也是5个，每个也是1G，但是这5个1G的逻辑盘已经与原来的5个物理硬盘意义完全不同了，例如第一个逻辑硬盘A里面，可能第一个200MB是来自物理硬盘1，第二个200MB是来自物理硬盘2，所以逻辑硬盘A是由多个物理硬盘逻辑虚构出来的硬盘），接着块设备会采用映射的方式将这几个逻辑盘映射给主机，主机上面的操作系统会识别出硬盘，但是操作系统是区分不出到底是逻辑还是物理盘。在此种方式下，操作系统还需要对挂载的裸磁盘进行分区、格式化后才可以使用，与平常主机内置硬盘的方式没有差别。

块存储不仅仅是直接使用物理设备，还有间接使用物理设备的也叫块设备，比如虚机创建虚拟磁盘。VMware、VirtualBox都可以创建虚拟磁盘，能够造出这个东西，且构造的东西能被当做磁盘去使用，就叫做块存储。虚机创建的磁盘格式包括raw、qcow2等，这与主机使用的裸设备不一样，且有不同的应用场景。

对于IO要求高的场景使用裸设备（直接操作硬件，即裸LUN），对于CPU要求高的使用qcow2（存在于文件系统上的镜像，其实文件系统最终还是建立在物理硬件上，这个其实就是多了层包装）。

**注：块存储与LVM、ISCSI**

块设备可以直接供设备使用，即裸设备，也可以经过LVM划分后，提供给用户逻辑卷LV使用，对于多个裸设备，可以通过ISCSI协议做RAID后给用户直接使用或者划分LVM。

## 应用场景

一般用于主机的直接存储空间和数据库应用的存储分两种形式：

DAS：一台服务器一个存储，多机无法直接共享，需要借助操作系统的功能，如共享文件夹；

SAN：金融电信级别，高成本的存储方式，涉及到光纤和各类高端设备，可靠性和性能都很高，除了贵和运维成本高，基本都是好处。

云存储的块存储：具备SAN的优势，且成本低，不用自己运维，且提供弹性拓容，随意搭配不同等级的存储功能，存储介质可选普通硬盘和SSD。

## 接口

这种借口通常以QEMU Driver（虚机磁盘qcow2）或者Kernel Module（裸LUN）的方式存在，这种接口需要实现Linux的Block Device的接口或者QEMU提供的Block Driver接口，如sheepdog、AWS的EBS，青云的云硬盘和阿里云的盘古系统，还有ceph的RBD。

## 特点

### 优点

1. 通过了RAID与LVM等手段，对数据提供了保护；
2. 可以将多块廉价的硬盘组合起来，成为一个大容量的逻辑盘对外提供服务，提高了容量；
3. 写数据时，由于是多块磁盘组合成的逻辑盘，所以几块磁盘可以并行写入，提升了读写效率；
4. 很多时候块存储采用SAN架构组网，传输速率以及封装协议的原因，使得传输速度与读写速率得到提升。

### 缺点

1. 采用SAN架构组网时，需要额外为主机购买光纤通道卡，还要买光纤交换机，造价成本高；
2. 主机之间的数据无法共享，在服务器不做集群的情况下，块存储裸盘映射给主机，在格式化使用后，对于主机来说就相当于本地盘，那么主机A的本地盘根本不能被主机B使用，无法共享数据（一旦格式化就是本地化的磁盘了，无法实现共享）。
3. 不利于不同操作系统主机间的数据共享，另外一个原因是因为操作系统使用不同的文件系统，格式化完成后，不同文件系统间的数据是无法共享的，例如一台安装了Win7，文件系统是FAT32/NTFS，而Linux是EXT4，EXT4是无法识别NTFS的文件系统的，就像是一直NTFS格式的U盘，插进Linux的笔记本，根本无法识别出来，所以不利于文件共享。

## 主流技术

Microsoft：Azure Block Storage

Google：Google Block Storage

Amazon：Elastic Block Storag（EBS）

OpenStack：Cinder

其他：Ceph RBD、sheepdog

# 文件存储

## 简介

为了克服上述文件无法共享的问题，所以就有了文件存储。

文件存储也有软硬一体化的设备，用一台普通服务器/笔记本，只要安装上合适的操作系统与软件，就可以架设FTP与NFS服务了，架上该类服务之后的服务器，就是文件存储的一种了。主机A可以直接对文件存储进行文件的上传下载，与块存储不同，主机A是不需要再对文件存储进行格式化了，因为文件管理功能已经将文件存储自己搞定了。

## 应用场景

与底层的块存储不同，上升到了应用层，一般指的是NAS，一套网络存储设备，通过TCP/IP进行访问，协议为NFSv3/v4由于通过网络，且采用上层协议，因此开销大，延时肯定比块存储高，一般用于多个云服务器共享数据，如服务器日志几种管理，办公文具共享。

## 接口

文件存储：通常意义是支持Posix接口，它跟传统的文件系统如EXT4是一个类型的，但区别在于分布式存储提供了并行化的能力，如ceph的CephFS，但是有时候又会把GFS，HDFS这种非POSIX接口的类文件存储接口引入此类。

## 特点

### 优点

1. 造价较低：随便一台机器就可以，另外普通以太网就可以，不需要专用的SAN网络；
2. 方便文件共享

### 缺点

1、读写速率低，传输速率慢：以太网，上传下载速度较慢，另外所有读写都要1台服务器里面的硬盘来承担，相比起磁盘阵列动不动就几十上百块磁盘同时读写，速率慢了许多。

## 主流技术

Microsoft；Windows Azure Blob

Google：Google FileStorage（GFS）

Amazon：Elastic File Storage(EFS)

OpenStack：Swift

其他：CephFS、HDFS、NFS、CIFS、Samba、FTP

# 对象存储

## 简介

对象存储最常用的方案，就是多台服务器内置大容量硬盘，再装上对象存储软件，然后再额外搞几台服务器作为管理节点，安装上对象存储管理软件，管理节点可以管理其他服务器对外提供读写访问功能。

之所以出现对象存储，是为了克服块存储与文件存储的缺点，发扬他俩各自的优点。简单地说，块存储读写块，不利于共享，文件存储读写慢，利于共享。

## 应用场景

具备块存储的高速以及文件存储的共享等特性，较为智能，有自己的CPU、内存、网络和磁盘，比块存储和文件存储更上层，云服务商一般提供用户文件上传下载读取的REST API，方便应用集成此类服务。

## 接口

对象存储也就是通常意义的键值存储，其接口就是简单的GET、PUT、DEL和其他拓展。

## 特点

### 优点

首先，一个文件包含了属性（术语叫metadata元数据，例如该文件的大小、修改时间、存储路径等）以及内容（具体数据）。

以前像FAT32这种文件系统，是直接将一份文件的数据与metadata一起存储的，存储过程先将文件按照文件系统的最小块大小打散（如4MB的文件，假设文件系统要求一个块大小4KB，那么就将文件打散成为100个小块），再写进硬盘里面，过程中没有区分数据metadata的。而每个块最后会告知你下一个要读取的块的地址，然后一直这样顺序地按图索骥，最后完成整份文件的所有块的读取，这种情况下读写速率很慢，因为就算你有100个机械手臂读写，但是由于你只有读取到第一块，才能知道下一个块在哪里，其实就相当于只能有1个机械手臂在实际工作。

而对象存储则将元数据独立出来，控制节点叫做元数据服务器（服务器+对象存储管理软件），里面主要负责存储对象的属性（主要是对象的数据被打散存放到了哪几台分布式服务器中的信息），而其他负责存储数据的分布式服务器叫做OSD，主要负责存储文件的数据部分。当用户访问对象，会先访问元数据服务器，元数据服务器只负责反馈对象存储在哪些OSD，假设返回文件A存储在B、C、D三台OSD，那么用户就会再次直接访问3台OSD服务器去读取数据。这时候由于是3台OSD同时对外传输数据，所以传输的速度就加快了。当OSD服务器数量越多，这种读写速度的提升就越大，通过这种方式，实现了读写快的目的。

另一方面，对象存储软件是具有专门的文件系统的，所以OSD对外又相当于文件服务器，那么就不存在文件共享方面的困难了，也解决了文件共享方面的问题。所以对象存储的出现，很好地结合了块存储与文件存储的优点。

### 缺点

为什么对象存储兼具块存储和文件存储的好处，那为什么还要块存储或者文件存储？

1. 有一类应用是需要存储直接裸盘映射的，例如数据库。因为数据库需要存储裸盘映射给自己后，再根据自己的数据库文件系统来对裸盘进行格式化的，所以是不能够采用其他已经被格式化为某种文件系统的存储的，此类应用更适合采用块存储。
2. 对象存储的成本比普通的文件存储还是较高，需要购买专门的对象存储软件以及大容量硬盘，例如对数据量要求不是海量，只是为了做文件共享的时候，直接使用文件存储的形式就好了，性价比高。

## 主流技术

Microsoft：Azure Storage

Google：Google Cloud Storage

Amazon：Simple Storage Service（S3）

OpenStack：Swift

其他：Ceph OSD，七牛云，又拍云

# 技术对比

块存储：和主机打交道，如插一块硬盘

文件存储：NAS，网络存储，用于多主机共享数据

对象存储：跟自己开发的应用程序打交道，如网盘

它们的层级是越来越高的，块存储更偏向底层

## 块存储 vs 文件存储

块级概念：块级是指以扇区为基础，一个或多个连续的扇区组成一个块，也叫物理块。它是在文件系统与块设备（如磁盘驱动器）之间。

文件级概念：文件级是指文件系统，单个文件可能由一个或多个逻辑块组成，且逻辑块之间不是连续分布的。逻辑块大于或等于物理块整数倍。

物理块与文件系统之间的关系图：映射关系：扇区🡪物理块🡪逻辑块🡪文件系统

文件级备份：文件级备份是指在指定某些文件进行备份时，首先会查找每个文件逻辑块，其次物理块，由于逻辑块是分散在物理块上，而物理块也是分散在不同扇区上。需要一层一层往下查找，最后才完成整个文件复制。

文件级备份比较费时间，效率不高，实时性不强，备份时间长，且增量备份时，单文件某一小部分修改，不会只备份修改部分，而是整个文件都备份。

块级备份：块级备份是指物理块复制，效率高，实时性强，备份时间短，且增量备份时，值备份修改过的物理块。

## 文件存储 vs 对象存储

对象存储：

1. 大多数对象存储的实现本质是键值对存储系统；
2. 采用扁平化的管理方式（根据键找到值）；
3. 值可以是任何东西，可以是小文件（小二进制片段），可以是大文件；
4. 对象存储一般不支持追加写和更新，面向的是一次写入，多次读取的需求场景；
5. 多采用Restful API。

文件存储：

1. 不考虑底层到底是怎么实现的（很多其实就是对象存储上套一层目录管理层）；
2. 采用目录结构管理数据；
3. 一般要尽可能兼容Posix文件系统API。