云存储的底层关键技术有哪些?

云计算 云存储 阿里云 七牛云 视界云

关注者 671 被浏览 25,684

云存储的底层关键技术有哪些?

分布式文件系统？只能想到这个，这个是不是还是比较宽泛？

关注问题 写回答 添加评论 分享 邀请回答 ···

9 个回答 默认排序

皮皮

73 人赞同了该回答

总的来说,分布式存储的底层所运用的技术随着系统设计目标的不同有各种各样的实现,总体上有上层架构设计时决定,但是总得来说我个人觉得有比较基本的几类,我印象中Dynamo的一篇论文概括得相当清晰,尽管已经是10年前的文章了,但是精髓还在,这里也特地去扒了下这张图:

Table 1: Summary of techniques used in Dynamo and their advantages.

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

我们看到,这里的技术基本都衍生于课本上的基础的数据结构,但是如何把这些技术应用于工程实践中得到一个稳定系统,需要结合很多实践的思考,是真正闪耀工程智慧的地方.

所以我认为并不是技术决定了架构,而是架构决定了该做什么样的技术选型.下面是我对存储系统设计的一些要点的浅显思考.

首先要定义什么才是云存储,我个人粗浅定义的云存储是:

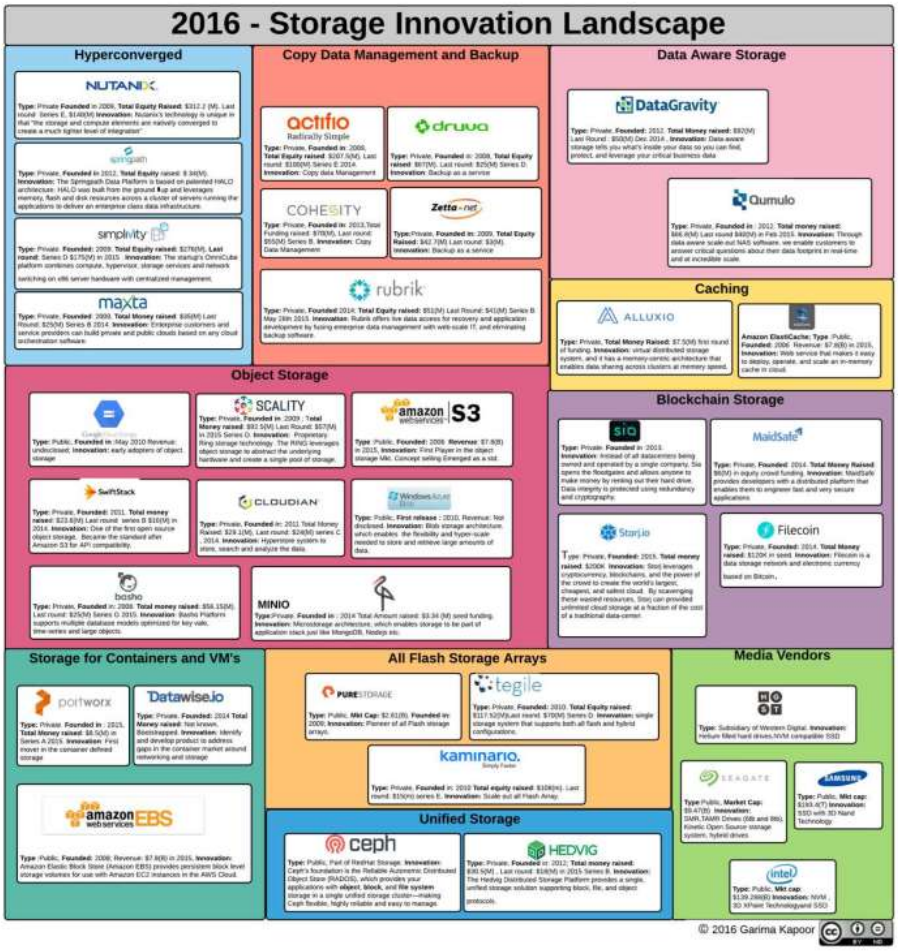
> 一个无法感知物理上的拓扑分布,从客户端看来支持无限水平拓展,按照一组定义好的接口通过网络进行非本地数据交互的存储系统,都可以称之为云存储系统(分布式存储).

可见,最基本的几个特点是:

- 抽象底层,屏蔽掉物理分布.
- 可支持无限或近于无限的水平扩容.
- 特定的访问接口

▲ 赞同 73 ▼ 1 条评论 分享 收藏 感谢

目前市面上除了网盘类的云存储之外.还有很多存储系统,他们大致分为通用存储和特定场景的存储系统.下面这张图基本概括了业界已知的一些存储产品(但有不少私有云产品没有出现,比如华为FusionStorage等等).



有这么多的产品出现,这个现象本身就说明了云存储系统的设计上是各有千秋,各有各的使用场景.所以每个存储系统底层涉及的技术也是有所不通.通常意义上从大方向上我们可以分为这么几类:

- 对象存储
- 块存储
- 文件存储

这里除了对象存储外,其他两个概念都是从一种单机的模式经过一定的架构之后成为了一个分布式系统.

分布式文件系统属于比较熟悉的一类的分布式系统,比如**HDFS**,**GlusterFS**等等.从GFS开始,为业界树立了很多分布式设计的准则,当然同时期国内很多公司都有自己类似的分布式系统.这类系统和后来的对象存储有很多共同的概念,但是依然保有文件和目录等层级结构的.而且或多或少弱化了很多POSIX语意,比如HDFS.这样对实现和性能上有很大提升.

块设备很长一段时间内都比较难做到水平扩容,都是通过SAN等臃肿的技术方案来解决.但是近些年来也出现了很多商用解决方案.比如UCloud的**UDisk**,华为的FusionStorage,SmartX的ZBS等等.都是通过把廉价的X86服务器做成存储池来为上层虚拟机提供海量的块设备存储服务.他们的标准接口通常都是SCSI和iSCSI.

鉴于工作的原因,我关注比较多的是对象存储(比如AWS的**S3**,openstack的**ceph**),这类系统近年来受到的关注比较大,也比较易于使用.他们的接口偏向于应用层上层,
web2.0的发展趋势.这类系统通常被设计成一个K-V系统.由于设

文件系统那样维护开销很大的inode信息,寻址速度也不会随着集群文件数量增加而下降.做到了真正的线性拓展.我个人经验认为做好一个存储系统很重要的几点包括(从应用层到底层):

- 设计目的和使用场景要清晰

每个系统都有自己要解决的问题,每个解决方法都有自己的trade off.这时候的关键就要均衡架构设计和问题可解性.也只有在详细了解了使用场景和需求之后才能真正保留最关键的功能,去除不必要的复杂度.让系统在未来具备可拓展.可以这么说,初期考虑得越少,加的功能越多,后期越无法收敛,系统将越来越庞大而丑陋.

- 索引和存储两端都要可拓展.易运维

对于每一个存储系统来说,面对的都是海量的数据,每份数据都会带来存储和索引的开销.这时两部分的抽象非常重要.原则就是要尽可能把实际的物理介质(索引存储的数据库,数据存储的磁盘)抽象出来,在上层具有一个可拓展,可迁移的逻辑单元.当然对象存储系统之间差异也很大,从潮流上看,基本都摒弃了索引的中心化存储方案.在寻址方面也各有各的花招,比如swift使用的一致性哈希虚节点技术,ceph使用的rados算法.当然这里也会有很多细节问题,比如如何防热点/冷热均衡,如何支持异构的设备,大文件/小文件的I/O优化等等.

- 数据一致性/持久性/安全性

作为一个存储系统,数据是最重要的资产.数据的一致性,因为受制于CAP理论,我们会牺牲掉一致性(因为网络系统的分区性可以认为是必然的),通常对象存储系统采用的是最终一致性,这在一定程度上降低了实现复杂度,同时也提升了系统吞吐量,在一定程度上为某些功能提供了可实现性(比如跨地域副本同步).数据持久性目前比较通用的机制是多副本,这是很经典的设计,可以提升数据吞吐量等优势,但是缺点是成本很高.所以很多解决方案根绝数据的冷热程度,会对冷数据做EC处理,但EC的一个劣势是在数据恢复时会降低故障节点的数据吞吐,加大网络拥塞(facebook最近公开的一项研究对这个问题有了不错的解决方案).

- 系统高可用

作为一个商用的系统,在牺牲掉一些对一致性要求非常高的场景(银行交易)后,对可用性的高要求是必然的.通常对高可用的设计是在架构中去除所有的单点可能,做到关键节点的冗余.做好系统容量负载监控和计划,从接入层往后,对内和对外都做好过载保护,负载均衡,尽可能消除带状态节点.随时能做水平扩容准备.

分布式存储的水太深,深感自己还没入门,在此抛砖引玉,希望对大家有所帮助.如果你对分布式系统也很感兴趣,希望真正接触到它,欢迎广大有志青年来UCloud,和我们一起构建顶级的公有云服务:-).

UCloud – 专业云计算服务商

编辑于 2016-07-25

int32bit
懂点OpenStack

49 人赞同了该回答

我认为需要理清这些关键技术，最好的方式是理解这些关键技术因何而生的，是为了解决什么问题。换句话说，如果要你自己手动实现一个分布式存储系统会如何设计。这里我们暂且不关心是什么块存储、对象存储还是文件存储。我们从设计一个分布式存储系统开始，逐步介绍涉及的关键技术。

分布式存储系统，通俗地讲就是要把文件存储到多个机器中。那么需要解决的第一个问题便是这些文件如何知道该保存到哪台服务器中。这里有两个思路，其中一个是设计一个控制服务器，由这个控制服务器负责统一调度，客户端请求存储一个文件时，首先与控制服务器交互，控制服务器返回需要保存到服务器的地址，读取文件时也需要与控制服务器交互，获取存储位置信息，其中HDFS、GFS等分布式存储使用此种技术，**namenode**就类似于控制服务器角色。另外一个思路是，不需要控制服务器，客户端自己计算需要存储到哪里，最简单的方式是直接取hash，比如有8台存储服务器，只需要把文件内容或者文件名取hash模8即可计
但有个问题是，当服务器数量增减时，hash就失效了，几乎需要

实现水平扩展，这在分布式系统中是无法忍受的。为了避免出现这种情况，引入了一致性hash算法，又称为环哈希，关于该算法可参考深入云存储系统Swift核心组件：Ring实现原理剖析 - 牛皮糖NewPtone - 博客园，其中OpenStack Swift、华为FusionStorage就是使用的该方法。除了环hash，当然还有其他的类hash算法，比如**CRUSH算法**，关于CRUSH算法介绍可参考大话Ceph--CRUSH那点事儿，其中开源分布式存储系统Ceph就是使用的该方法。需要注意的是虽然基于hash的文件分布映射方法不需要控制节点计算需要存储的位置，但仍然需要控制服务器保存一些集群元数据，比如集群的成员信息、映射规则、监控等等，如Ceph的mon服务。如果只有一个控制服务，则存在单点故障，挂掉了就会导致服务不可用。为了避免单点故障，具备高可用特点，必然需要同时启动多个控制服务，有多个控制服务就必须区分谁是leader，谁是slave，因此需要**分布式一致性来协调**选主，可以基于现有的分布式协调系统实现，如**Zookeeper、Etcd服务**等，也可以直接基于**Paxos、Raft**算法实现，Paxos相对复杂，Google的分布式锁服务Chubby就是基于**Paxos**实现的，Raft相对比较简单容易理解，可参看Raft-Understandable Distributed Consensus动态展示了该算法的执行流程，更多关于Raft的信息可参看官方文档主页Raft Consensus Algorithm。

文件直接映射到物理主机或者物理硬盘，粒度太粗略，容易导致数据分布不均匀。如果踢掉一台服务器或者一块硬盘，需要把这台服务器的数据迁移到重映射的另一台主机，迁移数据的IO都集中在这两台主机之间，其它主机帮不上忙。于是引入了**虚拟主机概念**，OpenStack Swift中叫做partition以及**Ceph中PG**等都是类似的概念。原理就是在物理主机上面加一层逻辑主机，比如有8台物理主机，可以创建128个虚拟主机，然后把这8台物理主机映射到这128台逻辑主机上，这样相当于每一台主机都虚拟成16台虚拟主机，当然实际上不一定是按照平均分，可以根据磁盘容量映射，磁盘空间大可以映射较多的虚拟主机。当然虚拟主机数量通常都会设置成2的幂，这样很多计算都可以使用位运算进行优化，比如取模运算等。这样文件块会先根据虚拟主机计算存储位置，然后再从表中查找虚拟主机映射的物理主机，文件块分布更均匀，当踢掉一台主机时会重映射到多台主机中，数据迁移效率提升。

解决了文件**如何分布的问题**，自然会遇到的问题是如果文件很大怎么办，可能在一台服务器根本存不下，即使存下了，也会导致各个服务器的磁盘利用率不均衡，甚至可能出现大量存储碎片。于是我们自然想到的是把文件分块，然后基于块存储，比如按照64MB大小分块，如果存储一个2GB的文件，则需要把文件分割成32个块，然后逐块存储，存储位置仍然使用前面提到的hash算法。分块是存储密度更大、更紧凑，几乎所有的分布式存储系统都会使用**分块技术**。

接下来，将考虑存储系统的**数据可靠性**（数据不丢）以及**可用性**（数据可访问）问题，如果其中一个服务器坏了怎么办？显然可能出现一个文件的某些块不能访问了，文件读取失败。为了解决这个问题，最容易想到的方法是使用**冗余技术**，即每一个块，我都存储多份，并分布到不同的服务器中，这样即使其中一个服务器宕机了，也能从其他服务器中读取块，这个和**RAID 1**技术原理是一样的。存储多少份呢，这个需要权衡成本以及数据可靠性要求，通常来说存储三份就够了。有人会说，存储三份，相当于使用了三倍的存储空间，这样存储资源是不是有点太浪费了，而又不想牺牲数据可靠性。我们学习算法时经常使用时间换空间的思想，计算换存储，这个仍然可以从RAID实现中获取灵感，以**RAID 5**为例，通过**奇偶校验**恢复数据，存储利用率为(n-1)/n，相比RAID 1的1/2提高了存储利用率，并且具有RAID 1一样的可靠性，但需要耗费CPU计算奇偶位。奇偶校验只能缺一位，自然可以想到进一步泛化，于是引入**纠删码技术，**原理其实就是类似解线性方程，关于纠删码技术介绍可以参考Erasure Code - EC纠删码原理。几乎所有的分布式存储系统都使用了冗余副本技术，大多数都会支持纠删码，比如Ceph、Swift。

无论使用纯副本技术还是结合纠删码，必然还是需要把一个块复制多份存储，写入多份，这里假设副本数为3份。这些副本如何写入呢？当拿到三个副本的位置后，客户端可以同时写入三个副本，这种方式称为**直接复制(direct replication)**，这样的问题是客户端会同时占用3倍的业务网络带宽，吞吐量也只有1/3，glusterfs采用的是这种复制策略。另一种方式是客户端只选择其中一个主节点写入数据，当写完第一个节点的数据后，由第一个节点复制到第二个节点，再由第二个节点复制到第三个节点，以此类推直到写完所有的副本，这种方式称为**链式复制(chain replication)**，Ceph、HDFS都是采用的该种策略，这样由于客户端其实只是写了一份数据，不占用额外的业务网络，而存储节点之间的复制可以是一个专门的存储网，不影响业务网络。关于chain replication可以参考论文Chain Replication for Supporting High Throughput and Availability。

写入多份数据，如何保证这些副本数据都是一样的，如何保证三个数据同步呢，万一哪台服务器挂了写不进去怎么办。于是引入了**一致性策略**。最简单的方法，就是等所有的副本都完成时才返回结果，这样保证写入的三个副本肯定没有问题，这就是强一致性，其中Ceph就是使用的**强一致性模型**，强一致性能够保证多副本完全一致，并且不会读取脏数据，

巨慢则会拖垮整个集群，典型的木桶效应，因此强一致性天生难以支持跨区域部署，因为跨区域的远端时延太长了，导致存储系统性能低。为了避免这种情况，我们可以适当放宽条件，即只要保证一半以上的服务器写入成功即返回，这样即使其中有少数服务器拖后腿也没有关系，不用等，让他自个慢慢同步，最终一致即可。这就是典型的**最终一致性模型**，OpenStack Swift即采用该种策略，这种模型能够提高读写性能，但可能读取脏数据，比如刚好读到还没有来得及同步的服务器的数据块。事实上高性能和强一致性是两者不可兼得的，这就是著名的**CAP理论**，这里的C代表一致性，A代表可用性（在一定时间内，用户的请求都会得到正确的应答），P代表分区容错。正常情况下，存储系统的所有节点都是互通的，处在一个网络连通区域中，如果有些节点之间不连通了（节点挂了或者网络故障），这就相当于把一个网络连通区域割裂了几个区域，彼此不能通信了，因此叫做分区。分布式存储系统要系统出现分区时数据不丢（可靠性），数据可访问（可用性），避免脑裂，因此P是100%需要满足的，否则稍微一个网络抖动，数据就损坏了。剩下的就是C和P之间的权衡，这个就看你要设计成什么存储系统了，如果一致性不那么重要，比如对象存储，上传了一个新文件，即使马上读不到数据也无所谓，但是可能需要支持大规模的对象写入，因此更关注A，设计为**AP存储系统**。而对于一些实时性要求高的系统，必须保证写入后数据一定能够读到正确的数据（而不是脏数据），就必须牺牲吞吐量，因此设计为**CP存储系统**。

为了节省存储空间，可能会用到**压缩技术**，压缩大家都很熟悉了，这里不多介绍。

如果是一个海量分布式存储系统，尤其是提供公有云服务，比如网盘服务，肯定会有用户上传一模一样的文件，为了节省成本，自然想到避免存储重复的文件，这就是**重删技术（**Data deduplication），可参考int32bit：百度云的「极速秒传」使用的是什么技术？，简单理解就是客户端上传文件时，先在本地计算下hash指纹，然后上传到服务器比对，如果指纹一样，说明文件已经存在，此时不需要上传文件内容，直接链接下即可，不仅节省了存储空间(比压缩更省），还节省了上传时间，实现秒传。我了解的Fusion Storage是实现了重删技术，OpenStack Swift、Ceph貌似都没有。

另一个问题是，如果集群彻底瘫了，数据就彻底没了，这可不能忍。为了解决这个问题，你自然会想到使用复制手段，即**备份技术，**把文件复制存储到其它廉价存储服务器中，比如S3。当用户执行save操作时，复制这个文件并重命名为xxx-20180312233020(时间戳），这样非常容易就能恢复到备份的任意版本，由于每次都要拷贝整个文件，因此称为全量备份(full backup)。每次都复制显然耗时耗空间，自然想到只复制上一次备份后改变的内容，这样就可以节省存储空间，即增量(incremental backup)备份。注意，备份一定要拷贝到其它存储系统，如果仅仅是拷贝到当前存储系统，不叫备份，只能叫副本，集群瘫了，数据仍然不能恢复。

以上备份技术需要用户自己手动执行，如果没有实时备份，集群突然挂了，数据还是会丢。因此需要采取**容灾策略**，其中一个容灾策略就是异地同步技术，或者叫做复制技术(geo-replication/mirror)，这个类似于mysql的主从同步，即在异地建立一个一模一样的集群，这个集群正常情况下不向用户提供存储服务，仅仅同步本地的集群数据，当本地的集群挂了，能够自动切换到异地集群，服务依然可用。注意这个和副本之间完全同步不一样，复制技术通常采用异步策略，基于操作日志replay，mysql使用binlog，ceph使用journal日志。ceph的rbd mirror就是采用的此种技术，关于rbd mirro介绍参考Ceph Jewel Preview: Ceph RBD mirroring。

分布式存储系统不仅需要大量的磁盘IO，还需要网络IO，然而网络带宽必然是有限的，有限的资源就必然需要合理的分配。如果某个用户持续不断的读写，抢占大量的IO带宽，则必然导致其它用户性能下降，甚至出现饿死状况。因此需要公平控制用户的IO资源使用情况，于是引入了**QoS**。QoS的目标是要实现系统IOPS的调度分配，对单一客户端的IOPS、IO带宽进行限制，不能让某个客户端独占了整个系统的IOPS。QoS限制客户端能够使用的最大值称为**limit**，即上限。注意，QoS不是仅仅有limit就够了，为了避免某些客户端迟迟得不到IO调度而被饿死，QoS还包含一个下限控制，称为**reservation**。上限limit容易理解，这个下限就容易弄混，因为有人会想，如果我的这个客户端就是没有IO需求，那它的IOPS就是0，这个下限有什么意义。其实这个下限是一个承诺，当客户端有大于reservation的IOPS请求时，系统能够保证给予不小于reservation的IOPS，如果客户端本身就不需要大于的reservation的值，那自然不需要分配其IOPS。因此，这个reservation翻译为预留更合适，系统调度IOPS时，会优先满足reservation的值，多余的再根据实际情况分配。当然在同时满足了请求的上限和下限下，不同的请求IOPS仍然不同，优先级也有可能不一样，因此还需要一个控制指标，称为分配比例。系统会根据权重去分配，能者多得，这样才能真正发挥资源的最大价值。关于QoS的实现，有两种思路，一种是直接使用Linux系统的**cgroup**实现，QEMU对虚拟机的磁盘QoS控制就是使用的该原理，这种方式不依赖于存储系统本身的实现，另一种就是QoS由存储系统自己，对某个节点实现QoS可参考VMware在OSDI

throughput variability for hypervisor IO scheduling，而dmClock即分布式的mClock，实现分布式系统的QoS控制，目前是作为Ceph的一个子项目，ceph/dmclock，关于dmClock的介绍可参考虚拟化I/O QoS mClock算法介绍。

为了继续下文内容，需要先介绍下几个不同的存储服务（接口）：

- 块存储：即提供裸的块设备服务，裸设备什么都没有，需要用户自己创建分区、创建文件系统、挂载到操作系统才能用，挂一个块存储设备到操作系统，相当于插一个新U盘。只实现了read、write、ioctl等接口。SAN、LVM、Ceph RBD、OpenStack Cinder等都属于块存储服务。
- 文件存储：可以简单理解为分布式文件系统，通常实现了POSIX接口，不需要安装文件系统，直接像NFS一样挂载到操作系统就能用。典型的文件存储如NAS、HDFS、CephFS、GlusterFS、OpenStack Manila等。
- 对象存储：提供Web存储服务，通过HTTP协议访问，只需要Web浏览器即可使用，不需要挂载到本地操作系统，实现的接口如GET、POST、DELETE等，典型的对象存储如百度网盘、S3、OpenStack Swift、Ceph RGW等。

有些存储系统只提供以上某种接口，有些存储系统则能够同时支持以上三种存储服务接口，比如Ceph。

块存储最典型的使用场景是作为虚拟机的磁盘。虚拟机通常需要申请几十GB到几TB的虚拟硬盘，但虚拟机实际上并不是一下就真的会用那么多的存储，如果申请多少就分配多少，显然会造成磁盘空间利用率不高，不能实现超售。因此引入了精简配置(thin provision），这个其实不难理解，就是类似于Linux的稀疏文件(sparse file)，关于Linux稀疏文件介绍可参考int32bit: sparse文件处理与传输，简单理解就是当申请一个20GB的虚拟磁盘时并不会立即真正从硬盘中分配空间，而是用多少分配多少，因此可以创建总大小远大于实际物理磁盘空间大小的磁盘，实现磁盘超售。但需要注意控制超售率，避免虚拟磁盘写满时，物理磁盘空间不足导致数据损坏。

分布式存储和本地存储一样，自然需要有版本控制，用户可以随时回滚到任意一个时间点的存储状态，或者基于某个时间点的版本修改，类似于git的checkout以及branch操作。当然你可以使用备份技术实现，只是太耗时耗空间。为了实现这个功能，引入**快照技术(snapshot)**，快照的功能形如其名，就是把当前的存储状态拍个照保存下来，以后可以随时回滚到某个快照时刻的状态。可以在快照的基础上，创建一个一样的磁盘卷，称为克隆（clone），注意和复制（copy）的不同，创建的新卷并没有拷贝源数据，也没有分配任何存储空间，只是画了一个指针指向原来的快照卷，秒级完成。OpenStack使用Ceph存储后端能够秒级创建虚拟机就是这个原因。当用户读取数据时，如果自己的卷没有找到，则需要在它parent中去找，如何写入则取决于采用何种快照方式。快照的实现有两种方式，一种是COW（Copy On Write，写时拷贝），快照是只读的，不允许修改，当用户有新的数据写入克隆的磁盘时，会首先从快照中拷贝一份数据写到另一个分配的空间，然后把修改的数据覆盖新分配的空间，相当于两次写操作。当磁盘创建快照，克隆，再创建快照，再克隆，形成一条很长的克隆链，当读取数据时，需要从当前卷开始查找，找不到查找其parent卷，直到查找到base镜像，写入数据也类似，当当前卷的块没有时，需要从其parent中依次查找，然后拷贝到自己的卷中，再写入新的数据。显然，当链越来越长时，卷的读写性能越来越差，因此需要控制链的长度。另一种快照技术是ROW(Redirect On Write)，这个和COW不同，当有新数据写入时，直接写入一个新分配的区，然后修改卷的指针指向新的区地址，只有一次写操作，关于COW与ROW介绍可以参考ROW/COW 快照技术原理解析。

虚拟机可能同时挂了多个虚拟硬盘，需要对这个虚拟机打快照，此时需要保证所有的卷的快照时一致的，而不能出现各个卷快照点不一致。于是引入了一致性快照技术(Consistency Snapshot Group)，参考存储专栏：深度解读高端存储的快照技术_存储在线。

以上，花了两个晚上粗略总结了分布式存储系统的一些关键技术，参考了很多博客文章，在原文中都有标明。需要强调的是真正实现分布式存储系统的技术远不止这些，这里仅仅作为抛砖引玉。其它的技术，诸如副本不一致时如何同步，新增或者减少节点时数据如何迁移，引入缓存提高性能等等，待有时间再补充。

编辑于 2018-03-29

丁凯
工程师，期望当一个坐台歌手，个人主页：d-kai.me

13 人赞同了该回答

快来邀请我啊，s3，ebs都做过，就差efs就可以召唤神兽草泥马了

==========华丽分割线==========

谢邀，是我臭不要脸的邀请作者来邀请我回答~

一切以客户的需求为出发点。传统存储以文件系统为典型代表，但是随着数据爆炸性增长，传统文件系统已经无法满足对存储系统的容量、性能等需求，因此，云存储应运而生。

云存储最大的特点是数据被集中存储在数据中心，公有云存储将客户数据存放在公有云服务商数据中心，而私有云存储则是将公有云存储能力私有化部署在客户自身的数据中心。

既然提到了数据中心，可想而知云存储最大的特点应该是海量：解决数以PB至EB的数据存储需求。所有云存储技术面对的通用问题有如下几个：

1. 扩展性：即容量可以通过横向增加服务器、磁盘等线性扩展，软件不应该成为限制扩展性的瓶颈；
2. 可靠性：如何保证数据不丢失，或者丢失概率极低；
3. 可用性：如何保证数据always online；
4. 性能：不同的客户的不同使用场景对云存储性能提出不同需求，如何保证存储系统满足性能需求
5. API：这是最容易被大家忽视的一个问题
6. 成本：花小钱办大事永远是商人追逐的目标

好，那接下来我们一个个剖析。

一：扩展性

好的扩展性取决于好的架构设计。主流云存储系统一般分为中心化和去中心化设计。中心化设计以GFS为典型代表，HDFS等也继承这种设计思想。顾名思义，中心化就是存在中心服务器会维护存储系统的关键元信息。维护这些元数据的关键作用是文件定位：即给定一个文件描述符，如何快速找到文件所在的存储位置（在哪个服务器的哪个磁盘上）。

当然，元信息的种类也有所不同，这种不同也体现了扩展性的差异，如GFS就在中心服务器中维护了文件属性等元数据，可想而知，随着文件数量的增长，必然达到一个瓶颈，于是更多的优化方案就出来了，例如元数据分割（静态分割、动态分割等）。

去中心化设计则力图避免该问题：数据的定位不再需要去中心服务器查询，而是通过特定的计算就可以找到文件的位置。可想而知，抛弃了元数据服务的束缚，整个系统就可以自由自在的翱翔了。当然，这也是有代价的，接下来就说。

去中心化固然好，但是由于存在节点变更时需要解决数据迁移问题，因此是一个头疼的问题，如何尽量减少数据迁移，又是一个大的课题，这里不展开细述。

因此，中心化的设计依然有其存在的价值，其简单的设计在分布式存储系统这个复杂的工程领域特别难得。

二：可靠性

有几天没来更新，请谅解，因为这几天在帝都吸雾霾，做贡献。今天就来聊聊可靠性吧，这应该是做存储的最关心的问题了，每天都在担心有没有给客户丢数据，做梦都能吓醒。

可靠性很好理解：数据别丢。那怎么保证数据不丢呢？

方案一：多副本

这很好理解，最简单粗暴的方法是将数据存在多个地方，这样，即使一个地方数据丢失了，还有其他的备份可用。这是最常见的做法，看似简单，实际非常复杂，需要考虑以下问题：

- 数据到底存储几备份才合适？
- 数据备份到底怎么存？
- 备份丢失时如何恢复？
- 多备份之间如何保证数据一致？
- 引入备份后对数据读写流程的侵入？

上面的这些问题既涉及工程经验问题，又包含了复杂的分布式理论知识（如多备份数据一致性），够吃好几壶了

方案二：纠删码

多副本好是好，就是代价太大了，需要多花费好几倍的成本来存储数据，资本家不会答应的。于是大家想起了通过计算的方式来为数据计算校验信息，在数据损坏时通过校验信息来恢复原始数据（参考RAID），同样很复杂，需要考虑的问题有：

- 选择何种算法，例如常见的Reed-Solomon，还有更高级的Cauchy-Reed-Solomon等等等等
- 何时进行纠删码计算？在线纠删码还是离线？Facebook还专门发表论文研究过这个问题；
- 数据错误如何检测以及如何恢复？

编辑于 2017-11-09

Rainbow chen
make yourself distinguished

11 人赞同了该回答

首先我们来说说什么是云存储?

从狭义上来说，云存储是指通过**虚拟化**、**分布式技术**、**集群应用**、**网格技术**、**负载均衡**等技术，将网络中大量的存储设备通过**软件集合**起来高效协同工作，共同对外提供**低成本**、**高扩展性**的数据存储服务。

从广义上来讲，云存储可以理解为**按需提供**的虚拟存储资源，如同云计算的Paas、Iaas服务一样，可称为数据存储即服务（Data Storage As a Service，DaaS），即基于指定的服务水平请求，通过网络提供适当的虚拟存储和相关数据服务。

云存储不是指某一个具体的设备，而是指一个由许许多多个存储设备和服务器所构成的集合体。使用者使用云存储，并不是使用某一个存储设备，而是使用整个云存储系统带来的一种数据访问服务。云存储的核心是应用软件与存储设备相结合，通过应用软件来实现存储设备向存储服务的转变。云存储就是将储存资源放到网络上供人存取的一种新兴方案。使用者可以在任何时间、任何地方，透过任何可连网的装置方便地存取数据。

综合定义，我们来看看云存储有哪些特点?

Ø **高可扩展性**：云存储系统可支持海量数据处理，资源可以实现按需扩展；

Ø **低成本**：云存储系统应具备高性价比的特点，低成本体现在两方面， 更低的建设成本和更低的运维成本；

Ø **无接入限制**：相比传统存储，云存储强调对用户存储的灵活支持， 服务域内存储资源可以随处接入，随时访问；

Ø **易管理**：少量管理员可以处理上千节点和PB级存储，更高效的支 撑大量上层应用对存储资源的快速部署需求。

了解了云存储的概念后，我们来看看云存储是什么架构?

传统的存储架构就是如下图所示的，存储、网络和主机都在同一个数据中心，客户通过局域网可以直接访问背后的存储。

而云存储是指通过虚拟化、分布式技术、集群应用、网格技术、负载均衡等技术，将分散在不同地方的大量的存储设备通过软件集合起来，客户通过公用访问接口、接入网和客户端程序等获取存储资源，客户并不知道所访问的存储资源处在什么地方。

Ø **存储层：**存储设备数量庞大且分布在不同地域，彼此通过广域网、互联网或光纤通道网络连接在一起。在存储设备之上是一个统一存储设备管理系统， 实现存储设备的逻辑虚拟化管理、多链路冗余管理，以及硬件设备的状态监控和故障维护。

Ø **基础管理层：**通过集群、分布式文件系统和网格计算等技术，实现云存储设备之间的协同工作，使多个的存储设备可以对外提供同一种服务， 并提供更大更强更好的数据访问性能。数据加密技术保证云存储中的数据不会被未授权的用户访问， 数据备份和容灾技术可以保证云存储中的数据不会丢失， 保证云存储自身的安全和稳定。

Ø **应用接口层：**不同的云存储运营商根据业务类型，开发不同的服务接口，提供不同的服务。例如视频监控、视频点播应用平台、网络硬盘，远程数据备份应用等。

Ø **访问层：** 授权用户可以通过标准的公用应用接口来登录云存储系统，享受云存储服务。

了解了基础架构之后，我们最后来看看云存储的关键技术有哪些?

1.云存储中的存储虚拟化

通过存储虚拟化方法，把不同厂商、不同型号、不同通信技术、不同类型的存储设备互联起来，将系统中各种异构的存储设备映射为一个统一的存储资源池。存储虚拟化技术能够对存储资源进行统一分配管理，又可以屏蔽存储实体间的物理位置以及异构特性，实现了资源对用户的透明性，降低了构建、管理和维护资源的成本，从而提升云存储系统的资源利用率。

存储虚拟化技术虽然不同设备与厂商之间略有区别，但从总体来说，**可概括为基于主机虚拟化、基于存储设备虚拟化和基于存储网络虚拟化**三种技术。

2. 云存储中的分布式存储技术

分布式存储是通过网络使用服务商提供的各个存储设备上的存储空间，并将这些分散的存储资源构成一个虚拟的存储设备，数据分散的存储在各个存储设备上。目前比较流行的分布式存储技术为：**分布式块存储、分布式文件系统存储、分布式对象存储和分布式**

3. 云存储中的数据缩减技术

为应对数据存储的急剧膨胀，企业需要不断购置大量的存储设备来满足不断增长的存储需求。权威机构研究发现，企业购买了大量的存储设备，但是利用率往往不足50%，存储投资回报率水平较低。通过云存储技术不仅解决了存储中的高安全性、可靠性、可扩展、易管理等存储的基本要求，同时也利用云存储中的数据缩减技术，满足海量信息爆炸式增长趋势，一定程度上节约企业存储成本，提高效率。

比较流行的数据缩减技术包括：**自动精简配置、自动存储分层、重复数据删除、数据压缩**

4. 数据备份技术

在以数据为中心的时代，数据的重要性无可置否，如何保护数据是一个永恒的话题，即便是现在的云存储发展时代，数据备份技术也非常重要。数据备份技术是将数据本身或者其中的部分在某一时间的状态以特定的格式保存下来，以备原数据出现错误、被误删除、恶意加密等各种原因不可用时，可快速准确的将数据进行恢复的技术。数据备份是容灾的基础，是为防止突发事故而采取的一种数据保护措施，根本目的是数据资源重新利用和保护，核心的工作是数据恢复。

5.内容分发网络技术

内容分发网络是一种新型网络构建模式，主要是针对现有的Internet进行改造。基本思想是尽量避开互联网上由于网络带宽小、网点分布不均、用户访问量大等影响数据传输速度和稳定性的弊端，使数据传输的更快、更稳定。通过在网络各处放置节点服务器，在现有互联网的基础之上构成一层智能虚拟网络，实时地根据网络流量、各节点的连接和负载情况、响应时间、到用户的距离等信息将用户的请求重新导向离用户最近的服务节点上。

6.存储加密技术

存储加密是指当数据从前端服务器输出，或在写进存储设备之前通过系统为数据加密，以保证存放在存储设备上的数据只有授权用户才能读取。目前云存储中常用的存储加密技术有以下几种：全盘加密，全部存储数据都是以密文形式书写的;虚拟磁盘加密，存放数据之前建立加密的磁盘空间，并通过加密磁盘空间对数据进行加密;卷加密，所有用户和系统文件都被加密;文件/目录加密，对单个的文件或者目录进行加密。

发布于 2017-04-19

▲ 赞同 11 ▼ 添加评论 分享 收藏 感谢 收起

vvv张浩
利益相关：Tencent Cloud 员工

32 人赞同了该回答

1、IO路径复杂，时延跟本地SSD相比，相差十倍，如何攻克是难点(自己造的轮子含泪也要打完对么)

2、接入层、逻辑层，存储层。三层架构消耗大量设备，存储往往存多副本，成本高昂，吃力不讨好。小厂商做云计算，块设备一般都用本地块设备，无任何QoS，任租户在上面互相抢占，画面太美。

3、老美说唯Tax和Death是躲不过的。搞存储的脑子里永远举着CAP的天平。宇宙第一定律告诉你，追求强一致性，性能、服务必然减损（P不可舍弃）；若提供最终一致性，则块存储没法玩儿了。满足强一致性的前提下，还得考虑scale的问题

4、SATA搞不定小随机IO，SSD太贵，冷热数据如何均衡，cache、淘汰策略怎么设计？（TM 2016年底到现在闪存涨了几倍你知道么）

抛砖引玉，占个楼看大牛回帖，要来我司的简历给我。钱多，事情也多，没雾霾，来吧

编辑于 2017-03-06

▲ 赞同 32 ▼ 11 条评论 分享 收藏 感谢 收起

▲ 赞同 73 ▼ 1 条评论 分享 收藏 感谢