存储系统的那些事

—— 七牛新存储（v2）上线有感

许式伟 于 2014年6月5日

存储系统从其与生俱来的使命来说，就难以摆脱复杂系统的魔咒。无论是从单机时代的文件系统，还是后来C/S或B/S结构下数据库这样的存储中间件兴起，还是如今炙手可热的云存储服务来说，存储都很复杂，而且是越来越复杂。

存储为什么会复杂，要从什么是存储谈起。存储这个词非常平凡，存储 + 计算（操作）就构成了一个朴素的计算机模型。简单来说，存储就是负责维持计算系统的状态的单元。从维持状态的角度，我们会有最朴素的可靠性要求。比如单机时代的文件系统，机器断电、程序故障、系统重启等常规的异常，文件系统必须可以正确地应对，甚至对于磁盘扇区损坏，文件系统也需要考虑尽量将损失降到最低。对于大部分的业务程序而言，你只需要重点关注业务的正常分支流程就行，对于出乎意料的情况，通常只需抛出一个错误，告诉用户你不该这么玩。但是存储系统你需要花费绝大部分精力在各种异常情况的处理上，甚至你应该认为，这些庞杂的、多样的错误分支处理，才是存储系统的“正常业务逻辑”。

到了互联网时代，有了C/S或B/S结构，存储系统又有了新指标：可用性。为了保证服务质量，那些用户看不见的服务器程序必须时时保持在线，最好做到逻辑上是不宕机的（可用性100%）。服务器程序怎么才能做到高可用性？答案是存储中间件。没有存储中间件，意味着所有的业务程序，都必须考虑每做一步就对状态进行持久化，以便自己挂掉后另一台服务器（或者自己重启后），知道之前工作到哪里了，接下去应该做些什么。但是对状态持久化工作（也就是存储）非常繁琐，如果每个业务都自己实现，负担无疑非常沉重。但如果有了高可用的存储中间件，服务器端的业务程序就只需操作存储中间件来更新状态，通过同时启动多份业务程序的实例做互备和负载均衡，很容易实现业务逻辑上不宕机。

所以，数据库这样的存储中间件出现基本上是历史必然。但尽管数据库很通用，它决不会是唯一的存储中间件。比如业务中用到的富媒体（图片、音视频、Office文档等），我们很少会去存储到数据库中，更多的时候我们会把它们放在文件系统里。但是单机时代诞生的文件系统，真的是最适合存储这些富媒体数据的么？不，文件系统需要改变，因为：

1. 伸缩性。单机文件系统的第一个问题是单机容量有限，在存储规模超过一台机器可管理的时候，应该怎么办的问题。

2. 性能瓶颈。单机文件系统在通常在文件数目达到临界点后，性能快速下降。在4TB的大容量磁盘越来越普及的今天，这个临界点相当容易到达。
3. 可靠性要求。单机文件系统通常只是单副本的方案，但是今天单副本的存储已经早无法满足业务的可靠性要求。数据需要有冗余（比较经典的做法是3副本），并且在磁盘损坏时及早修复丢失的数据，以避免所有的副本损坏造成数据丢失。
4. 可用性要求。单机文件系统通常只是单副本的方案，在该机器宕机后，数据就不可读取，也不可写入。

在分布式存储系统出现前，有一些基于单机文件系统的改良版本被一些应用采纳。比如在单机文件系统上加 RAID5 做数据冗余，来解决单机文件系统的可靠性问题。假设 RAID5 的数据修复时间是1天（实际上往往做不到，尤其是业务系统本身压力比较大的情况下，留给 RAID 修复用的磁盘读写带宽很有限），这种方案单机的可靠性大概是100年丢失一次数据（即可靠性是2个9）。看起来尚可？但是你得小心两种情况。一种是你的集群规模变大，你仍然沿用这个土方法，比如你现在有 100 台这样的机器，那么就会变成1年就丢失一次数据。另一种情况是如果实际数据修复时间是 3 天，那么单机的可靠性就直降至4年丢失一次数据，100台就会是15天就丢失一次数据，这个数字显然无法让人接受。

Google GFS 是很多人阅读的第一份分布式存储的论文，这篇论文奠定了 3 副本在分布式存储系统里的地位。随后 Hadoop 参考此论文实现了开源版的 GFS —— HDFS。但关于 Hadoop 的 HDFS 实际上业界有不少误区。GFS 的设计有很强的业务背景特征，本身是用来做搜索引擎的。HDFS 更适合做日志存储和日志分析（数据挖掘），而不是存储海量的富媒体文件。因为：

1. HDFS 的 block 大小为 64M，如果文件不足 64M 也会占用 64M。而富媒体文件大部分仍然很小，比如图片常规尺寸在 100K 左右。有人可能会说我可以调小 block 的尺寸来适应，但这是不正确的做法，HDFS 的架构为大文件而设计的，不可能简单通过调整 block 大小就可以满足海量小文件存储的需求。
2. HDFS 是单 Master 结构，这决定了它能够存储的元数据条目数有限，伸缩性存在问题。当然作为大文件日志型存储，这个瓶颈会非常晚才遇到；但是如果作为海量小文件的存储，这个瓶颈很快就会碰上。
3. HDFS 仍然沿用文件系统的 API 形式，比如它有目录这样的概念。在分布式系统中维护文件系统的目录树结构，会遭遇诸多难题。所以 HDFS 想把 Master 扩展为分布式的元数据集群并不容易。

分布式存储最容易处理的问题域还是单键值的存储，也就是所谓的 Key-Value 存储。只有一个 Key，就意味着我们可以通过对 Key 做 Hash，或者对 Key 做分区，都能够让请求快速定位到特定某一台存储机器上，从而转化为单机问题。这也是为什么在数据库之后，会冒出来那么多 NoSQL 数据库。因为数据库和文件系统一样，最早都是

单机的，在伸缩性、性能瓶颈（在单机数据量太大时）、可靠性、可用性上遇到了相同的麻烦。NoSQL 数据库的名字其实并不恰当，他们更多的不是去 SQL，而是去关系（我们知道数据库更完整的称呼是关系型数据库）。有关系意味着有多个索引，也就是有多个 Key，而这对数据库转为分布式存储系统来说非常不利。

七牛云存储的设计目标是针对海量小文件的存储，所以它对文件系统的第一个改变也是去关系，也就是去目录结构（有目录意味着有父子关系）。所以七牛云存储不是文件系统（File System），而是键值存储（Key-Value Storage），用时髦点的话说是对象存储（Object Storage）。不过七牛自己喜欢把它叫做资源存储（Resource Storage），因为它是用来存储静态资源文件的。蛮多七牛云存储的新手会问，为什么我在七牛的 API 中找不到创建目录这样的 API，根本原因还是受文件系统这个经典存储系统的影响。

七牛云存储的第一个实现版本，从技术上来说是经典的 3 副本的键值存储。它由元数据集群和数据块集群组成。每个文件被切成了 4M 为单位的一个个数据块，各个数据块按 3 副本做冗余。但是作为云存储，它并不仅仅是一个分布式存储集群，它需要额外考虑：

1. 网络问题，也就是文件的上传下载问题。文件上传方面，我们得考虑在相对比较差的网络条件下（比如2G/3G网络）如何确保文件能够上传成功，大文件（七牛云存储的单文件大小理论极限是1TB）如何能够上传成功，如何能够更快上传。文件下载加速方面，考虑到 CDN 已经发展了 10 多年的历史，非常成熟，我们决定基于 CDN 来做下载加速。
2. 数据处理。当用户文件托管到了七牛，那么针对文件内容的数据处理需求也会自然衍生。比如我们第一个客户就给我们提了图片缩略图相关的需求。在音视频内容越来越多的时候，自然就有了音视频转码的需求。可以预见在Office文档多了后，也就会有 Office 文档转换的需求。

所以从技术上来说，七牛云存储是这样的：

七牛云存储 = 分布式存储集群 + 上传加速(下载外包给CDN) + 数据处理集群

网络问题并不是七牛要解决的核心问题，只是我们要面对的现实困难。所以在这个问题上如果能够有足够专业的供应商，能够外包我们会尽可能外包。而分布式存储集群的演进和优化，才是我们最核心的事情。早在 2012 年 2 月，我们就启动了新一代基于纠删码算术冗余的存储系统的研发。新存储系统的关注焦点在：

1. 成本。经典的 3 副本存储系统虽然经典，但是代价也是高昂的，需要我们投入 3 倍的存储成本。那么有没有保证高可靠和高可用的前提下把成本做下来？

2. 可靠性。如何进一步提升存储系统的可靠性？答案是更高的容错能力（从允许同时损坏2块盘到允许同时损坏4块盘），更快的修复速度（从原先3小时修复一块坏盘到30分钟修复一块坏盘）。
3. 伸缩性。如何从系统设计容量、IO吞吐能力、网络拓扑结构等角度，让系统能够支持EB级别的数据存储规模？关于伸缩性这个话题，涉及的点是全方位的，本文不展开讨论，后面我们另外独立探讨这个话题（让我们把焦点放在成本和可靠性上）。

在经过了四个大的版本迭代，七牛新一代云存储（v2）终于上线。

新存储的第一大亮点是引入了纠删码（EC）这样的算术冗余方案，而不再是经典的 3 副本冗余方案。我们的 EC 采用的是 28 + 4，也就是把文件切分为 28 份，然后再根据这 28 份数据计算出 4 份冗余数据，最后把这 32 份数据存储在 32 台不同的机器上。这样做的好处是既便宜，又提升了可靠性和可用性。从成本角度，同样是要存储 1PB 的数据，要买的存储服务器只需 3 副本存储的 36.5%，经济效益相当好。从可靠性方面，以前 3 副本只能允许同时损坏2块盘，现在能够允许同时损坏4块盘，直观来说这大大改善了可靠性（后面讨论可靠性的时候我们给出具体的数据）。从可用性角度，以前能够接受 2 台服务器下线，现在能够同时允许 4 台服务器下线。

新存储的第二大亮点是修复速度，我们把单盘修复时间从 3 小时提升到了 30 分钟以内。修复时间同样对提升可靠性有着重要意义（后面讨论可靠性的时候我们给出具体的数据）。这个原因是比较容易理解的。假设我们的存储允许同时坏 M 块盘而不丢失数据，那么集群可靠性，就是看在单位修复时间内，同时损坏 M+1 块盘的概率。例如，假设我们修复时间是 3 小时，那么 3 副本集群的可靠性就是看 3 小时内同时损坏 3 块盘的概率（也就是丢数据的概率）。

让我们回到存储系统最核心的指标 —— 可靠性。首先，可靠性和集群规模是相关的。假设我们有 1000 块磁盘的集群，对于 3 副本存储系统来说，这 1000 块盘同时坏 3 块就会发生数据丢失，这个概率显然比 3 块盘同时坏 3 块要高很多。基于这一点，有些人会想这样的土方法：那我要不把集群分为 3 块磁盘一组互为镜像，1000 块盘就是 333 组（不好意思多了1块，我们忽略这个细节），是不是可以提升可靠性？这些同学忽略了这样一些关键点：

1. 3 块盘同时坏 3 块盘（从而丢失数据）的概率为 p，那么 333 组这样的集群，丢失数据的概率是 1 - (1-p)^333 ≈ p * 333，而不是 p。
2. 互为镜像的麻烦之处是修复速度存在瓶颈。坏一块盘后你需要找一个新盘进行数据对拷，而一块大容量磁盘数据对拷的典型时间是 15 小时（我们后面将给出 15 小时同时坏 3 块盘的概率）。要想提升这个修复速度，第一步我们就需要打破镜像带来的束缚。

如果一个存储系统的修复时间是恒定的，那么这个存储集群在规模扩大的时候，必然伴随着可靠性的降低。所以最理想的情况是集群越大，修复速度越快。这样才能抵消因集群增大导致坏盘概率增加带来负面影响。计算表明，如果我们修复速度和集群规模成正比（线性关系），那么集群随着规模增大，可靠性会越来越高。

假设现在有一个 1000 块硬盘的存储集群，为了大家能够对容错度（M）、修复时间对存储系统的可靠性影响有个直观的认知，我们整理了如下表格：

	修复时间: 30分钟	**修复时间: 3小时**	**修复时间: 15小时**
3副本（M=2）	数据丢失概率：1e-8（8个9的可靠性）	数据丢失概率：1e-5（5个9的可靠性）	数据丢失概率：1e-2（2个9的可靠性）
28+4（M=4）	数据丢失概率：1e-16（16个9的可靠性）	数据丢失概率：1e-11（11个9的可靠性）	数据丢失概率：1e-7（7个9的可靠性）

也就是说，一个 1000 块硬盘、修复时间30分钟的存储集群，对于 3 副本方案，那么1年内出现数据丢失的概率 p(M=2) 为 1e-8（也就是8个9的可靠性）。对于 28 + 4 算术冗余方案，那么1年内出现数据丢失的概率 p(M=4) 为 1e-16（也就是16个9的可靠性）。其他依次类推。关于这个 p 具体是如何计算得来，需要非常长的推导过程，本文是存储技术的普及性文章，我们尽量用非存储从业人员能够听得懂的方式来描述存储的那些事，所以为了不干扰本文的主线，我们另外找机会讨论。

对我个人而言，七牛新一代云存储（v2）的完成，了了我多年的夙愿。但七牛不会就此停止脚步。我们在存储系统上又有了一些好玩的想法。从长远来说，单位存储的成本会越来越廉价（硬件和软件系统都会推动这个发展趋势）。而存储系统肯定会越来越复杂。例如，有赖于超高的容错能力，七牛对单块磁盘的可靠性要求降低了很多，这就为未来我们采用桌面硬盘而不是企业硬盘作为存储介质打下基础。但是单块磁盘可靠性的降低，则会进一步推动存储系统往复杂的方向发展。基于这个推理，我认为存储必然需要转为云服务，成为水电煤一样的基础设施。存储系统越来越复杂，越来越专业，这就导致自建存储的难度和成本越来越高，自建存储的必要性也越来越低。必然有那么一天，你会发现云存储的成本远低于自建存储的成本，到时自建存储就会是纯投入而无产出，也就没有多少人会去热衷于干这样的事情了。