

38 | 文件系统与对象存储

2019-09-03 许式伟

许式伟的架构课

[进入课程 >](#)



讲述：姚迪迈

时长 16:06 大小 14.76M



你好，我是七牛云许式伟。

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。

异常处理才是存储的业务逻辑

存储为什么会复杂，要从什么是存储谈起。

让我们简单回顾一下 [“36 | 业务状态与存储中间件”](#) 的核心逻辑。

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。

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靠、高可用？

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。

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”。

所以，数据库这样的存储中间件出现基本上是历史必然。

从文件系统谈起

但尽管数据库很通用，它决不会是唯一的存储中间件。

比如，在服务端开发中，我们业务用到的多媒体（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多媒体数据的吗？

不，文件系统需要改变，因为：

第一，伸缩性问题。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。

第二，性能瓶颈。单机文件系统通常在文件数目达到临界点后，性能快速下降。在 10TB 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。

第三，可靠性，更严谨来说是持久性（Durability）问题。单机文件系统通常只是单副本的方案。但是，今天单副本的存储早已经无法满足业务的持久性要求。

数据需要有冗余（比较经典的做法是 3 副本），以便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。

第四，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？但是我们得考虑两个问题。

第一，你的集群规模会变大。如果你仍然沿用这个土方法，比如你现在有 100 台这样的机器，那么它就会变成 1 年就丢失一次数据。

第二，你采购的磁盘容量会变大。如果实际数据修复时间没那么理想，比如变成 3 天，那么单机的可靠性就直降至 4 年丢失一次数据。100 台这样的机器就会是 15 天就丢失一次数据。

这个数字显然无法让人接受。

所以服务端存储只要规模够大，就会使得很多看起来是小概率的事件，变成必然事件。

什么样的数据会有最大的存储规模？

答案是非结构化数据。这类数据的组织形式通常以用户体验友好为目标，而不是机器友好为目标。所以数据本身也自然不是以机器易于理解的结构化形式来组织。

图片、音视频、Office 文档等多媒体文件，就是比较典型的非结构化数据。互联网上 90% 以上传输的数据量都是非结构化数据。

移动互联网、人工智能与物联网技术的发展，进一步加快了非结构化数据的产生。从读图时代，到视频与实时互动，以及未来的 AR/VR 技术，人们正在一步步把物理世界映射到数字世界，通过数字世界实现更随时随地的、更自然的沟通体验，通过数字世界更好地理解和管理我们的物理世界。

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：

第一，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几百 K 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。

第二，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（一般单个日志文件大小在 1GB 级别），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。

第三，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。

对象存储

非结构化数据的存储方式，最理想的绝对不是分布式文件系统。

文件系统只是桌面操作系统为了方便用户手工管理数据而设计的产物。服务端操作系统发展的初期，人们简单沿用了桌面操作系统的整套体系框架。

但从非结构化数据的存储开始，出现了分叉路口。对服务端体系架构来说，文件系统其实是一个过时的东西。

非结构化数据最佳的存储方式，还是键值存储（KV Storage）。用于存储非结构化数据的键值存储，有一个特殊的名字，叫对象存储（Object Storage）。它和结构化数据的键值存储，实现机制上往往有极大的差异。

对象存储的 Key，看起来也像一个文件系统的路径（Path），但仅仅是像而已。对于对象存储来说，Key 中出现的 “/” 字符，只是一个普通字符。

在对象存储中，并不存在目录（Directory）这样的概念。

既然对象存储是一个键值存储，就意味着我们可以通过对 Key 做 Hash，或者对 Key 按 Key Range 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，它们更多的不是去 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），而是对象存储（Object Storage）。蛮多七牛云的新手会问，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。

第一个大家公认的对象存储是 AWS S3，你可以把它理解为一个非常简单的非结构化数据存储，它最基本的访问接口如下：

 复制代码

```
1 func PutObject(bucket, key string, object io.Reader) (err error)
2 func GetObject(bucket, key string) (object io.ReadCloser, err error)
```

七牛云存储并不仅仅是简单的分布式存储，它需要额外考虑以下这些问题。

第一，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 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几个 TB）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 技术来做下载加速。

第二，多媒体处理。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。

所以从用户使用的角度来看，七牛云存储是这样的：

七牛云存储 = 对象存储 + 上传下载加速 + 多媒体处理

存储成本与持久性

既然对象存储的存储规模最大，占据了 90% 以上存储需求，那么毫无疑问，它最关心的就是单位存储成本问题。通常我们用每 GB 每月花费多少钱来表示单位存储成本。

前面我们说了，GFS 这个经典的分布式文件系统，采用的是 3 副本的方式。这样做的好处是可靠，不容易发生数据丢失。

但是它的问题也很明显，就是存储成本非常高。如果我们排除不同公司的采购能力差异外，存储成本最大的关联因素是以下两个东西。

其一是存储密度。存储密度越高，单台机器的存储量越大，单位成本越低。存储密度取决于：单台机器能够插的硬盘数量、单块磁盘的容量。

其二是冗余度。GFS 采用的是 3 副本，也就是冗余度为 3。当前降低冗余度，通常采用的是纠删码（EC）这样的算术冗余方案。

比如，假设 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 份数据存储到 32 台不同的机器上。

这样做的好处是既便宜，又提升了持久性和可用性。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 38% ($32/28=1.14$, $1.14/3=38\%$)，经济效益相当好。

从持久性方面，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持久性。

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。

通过上面的分析可以看出，冗余度降低不一定会伤害集群的持久性和可用性，它们和冗余度不是正相关，而和集群的容错能力相关。

但是存储密度对系统的可用性和可靠性都会有一定的伤害。我们定性分析一下这里面的关系是什么样的。

我们重点考虑存储的核心指标：持久性（Durability）。它取决于以下两个关键指标。

一是单位修复时长，也就是一块磁盘损毁后，需要多久修复回来，假设这个修复时长为 T_0 。

二是容错能力，也就是集群允许同时有几块硬盘损坏，假设我们采用的纠删码是 $N + M$ 方案，那么我们的容错能力是可以接受同时损坏 M 块硬盘。

我们定性来看， T_0 时间内同时坏 M 块盘的概率，就是我们丢失数据的概率。因此，持久性 (Durability) 和 T_0 、 M 这两个参数相关。

存储密度对持久性的影响是什么？

假设集群总的容量规模不变，我们把单台机器的磁盘数量增加一倍，那么我们需要的机器数量减少一半。但由于集群的磁盘数量不变，我们的修复时长 T_0 也不变（假设网络 and CPU 计算力都不是瓶颈）。假设原本丢失数据的概率是 p ，那么现在丢失数据的概率还是 p 。

也就是说，在保证有足够的网络和计算力前提下，增加单台机器的磁盘数量，可能会降低可用性，但是对持久性几乎不会造成影响。

假设集群总的容量规模不变，但我们不是增加单台机器的磁盘数量，而是增加磁盘的密度。比如，我们把单盘容量增加一倍，那么我们集群的磁盘数也减少一半。这样我们的修复时长 T_0 会变成 $4T_0$ （修复时间和要修复的数据量成正比，和集群可用的磁盘数成反比）。

从这个角度看，提高磁盘密度对持久性的伤害还是比较大的。但是如果我们假设单块磁盘的坏盘概率和磁盘容量无关的话，由于磁盘数量减少了一半，这对集群整体的坏盘率又是一个正向的影响。

综合来说，假设原本丢失数据的概率是 p ，那么现在丢失数据的概率是：

$$1 - [(1-p)^{0.5}]^4 \approx 2p$$

即约等于 $2p$ 。

我们再来看一下集群的容量规模对持久性的影响。

假如我们将集群扩容一倍。那么我们的修复速度会快一倍，修复时长 T_0 会变成 $0.5T_0$ ，这是一个很正面的影响。但是由于磁盘数量增加了一倍，所以坏盘概率会增加，这又是一个负面的影响。

综合来说，一正一反两相抵消，集群规模对集群整体的持久性大体可以忽略。不过这只是一种非常粗略的估计方法。更严谨的演算表明，集群规模增加整体上对集群的持久性（Durability）是正向的影响。

也就是说，集群规模越大，存储的可靠性越高。当然，这一点的前提是前面我们的假设，修复速度和集群规模成正比成立。

结语

今天我们讨论了对象存储相关的核心话题。我们从文件系统谈起，介绍了非结构化数据的存储系统的演进历史。

对象存储的出现，是服务端体系架构和桌面操作系统分道扬镳的开始。后续两者的演进方向变得越来越大不相同。

由于承载了最大体量的数据规模，对象存储对单位存储成本极其敏感。我们定性探讨了成本与持久性（Durability）之间的平衡关系。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们将聊聊内存缓存。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

许式伟的架构课

从源头出发, 带你重新理解架构设计

许式伟
七牛云 CEO



新版升级: 点击「 请朋友读」, 20位好友免费读, 邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有, 未经许可不得传播售卖。页面已增加防盗追踪, 如有侵权极客邦将依法追究其法律责任。

上一篇 37 | 键值存储与数据库

下一篇 39 | 存储与缓存

精选留言 (16)

写留言



Geek_88604f

2019-09-03

非结构化和结构化数据的键值存储, 实现机制上往往有极大的差异。主要体现在哪些方面, 许老师?

作者回复: 主要还是因为值大小差异非常大, 导致两者优化目标不同。对象存储的第一优化目标是低成本, 结构化键值存储的优化目标是高iops, 方向完全不同。



2



Aaron Cheung

2019-09-03

存储这块是七牛的业务强项 从许老师这里学习了 38

展开 ▾



👍 2



JACK

2019-09-03

36.5是怎么算出来的?麻烦再讲解一下, 谢谢!

展开 ▾

作者回复: $32/28=1.14$

$1.14/3=36.5\%$



💬 1

👍 1



醉雪飘痕

2019-09-06

许老师好,

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）。

请问这个100年是怎么算出来的？

展开 ▾



歌在云端

2019-09-06

像老师请教一个问题，对象存储是怎么持久化到硬盘的？也是序列化之后存吗？请问一下跟文档型数据库类似MongoDB有什么区别

展开 ▾



安排

2019-09-06

对象存储底层实现时还会经过操作系统自带的文件系统吗？还是说对象存储操作的直接是裸盘？相当于自己重新实现了文件系统功能？

展开 ▾

作者回复: 对象存储有一些是基于裸盘，有一些基于文件系统。推荐是前者。



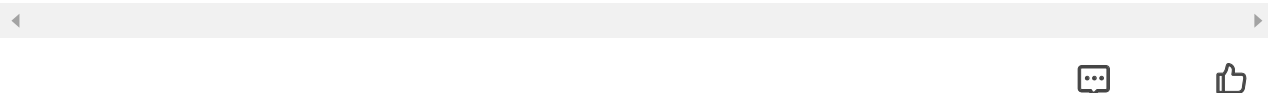


安排

2019-09-06

结构化数据和非结构化数据啥区别啊？表结构就是结构化数据吗？

作者回复: 主要是数据类型不一样。非结构化数据主要是图片、视频、应用日志等。表结构是结构化的。



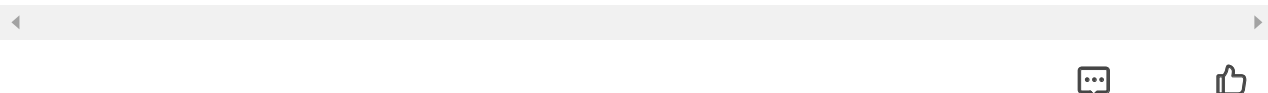
yga

2019-09-06

假设集群总的容量规模不变，但我们不是增加单台机器的磁盘数量，而是增加磁盘的密度。比如，我们把单盘容量增加一倍，那么我们集群的磁盘数也减少一半。这样我们的修复时长 T_0 会变成 $4T_0$ （修复时间和要修复的数据量成正比，和集群可用的磁盘数成反比）

T_0 怎么变成 $4T_0$ ？修复时间和数据量成正比，和集群可用磁盘数成反比。为什么呢，希望...
展开

作者回复: 因为磁盘容量变大一倍，要修复的数据量变大一倍，而系统的总磁盘数减半，所以最终修复时间需要4倍。

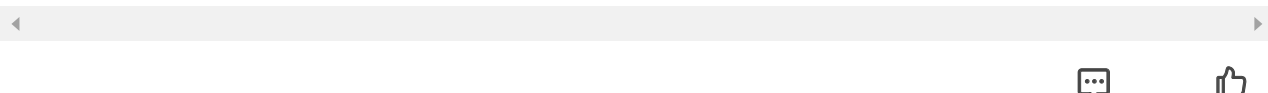


Geek_88604f

2019-09-05

说到hdfs的文件块，首先看看为什么用块来管理而不用文件。因为块有如下优势:单文件的大小不会受到单节点容量的限制，文件的不同的部分可以存储到不同的节点上；容易对数据进行备份提高容错能力，可以按块来备份；可以按照块来并发处理提高吞吐量。从整体上来看简化了存储系统的设计，一是按照块来设计可以很容易地估算出一个节点可以存出多少文件块；二是方便元数据管理，元数据可以和文件块分开存储，我理解桌面存储系...
展开

作者回复: 多谢补充



williamcai

2019-09-04

mc冗余码，数据切成28片 + 4份冗余，这4分冗余那些数据

作者回复: 是算术结果。 $Y_i = F_i(X_1, X_2, \dots, X_{28})$, 其中 $i = 1..4$ 。



coyang

2019-09-04

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。 -> GlusterFS 就是分布式的元数据集群，没有 master, active/standby 的概念。实现也很复杂。许老师能不能说一下未来分布式存储的趋势？个人感觉分布式文件系统还会长期保留来适应老的 application 迁移到分布式系统不用做任何改变。

展开 ∨

作者回复: 是的，老 application 不消亡，文件系统在服务端就会一直存在。



EidLeung

2019-09-03

“第一，HDFS 的 block 大小为 64M，如果文件不足 64M 也会占用 64M。”这句话有问题吧，HDFS 的问题是文件的元数据存储在内核中，因此不适合小文件。不足 64M（现在好像是 128M 了吧）的小文件在 HDFS 中存储只是在磁盘多了一些元数据信息而已，文件大小还是原来的大小啊。

展开 ∨

作者回复: 多谢补充，对 hdfs 的近况了解的确不多



靠人品去赢

2019-09-03

对象存储学到了，但是文件系统，路径，文档这些概念已经习惯有时候转不过来，突然你说这是个对象可能理解不了，觉得这不就是一个 IO 流写入的文件吗。

还有提高磁盘密度对持久化有影响，是不是我可以理解像机械硬盘叠瓦式（支持磁道可重叠，提高单盘容量），固态（一个单元储存的 byte 越来越多，QLC, PLC 都要出来了）是不是都是，不仅修复难而且掉盘率搞，读写速度感人，这些硬件的影响。

展开 ▾

作者回复: 1、对象存储这个名字仅仅是名字，不要太想对象两个字代表什么。
2、持久性的探讨只是定性分析，这块不是我们讨论的重点。

◀ ▶

💬 👍



有铭

2019-09-03

老师，你提到七牛云的对象存储是没有目录这个概念的；我记得阿里的oss是有的，也就是说，两者的实现原理并不一样？

展开 ▾

作者回复: 阿里也没有，目录是模拟出来的，这一点所有对象存储都一样。在七牛之间中也可以模拟出目录概念，但是它和真实的目录差别非常大。

◀ ▶

💬 2 👍



诗泽

2019-09-03

想到了Facebook 那篇关于小图片存储的论文📄

展开 ▾

💬 2 👍



Geek_88604f

2019-09-03

假设集群总的容量规模不变，我们把单台机器的磁盘数量增加一倍，那么我们需要的机器数量减少一半。但由于磁盘数量不变，我们的修复时长 T_0 也不变（假设网络和 CPU 计算力都不是瓶颈）。假设原本丢失数据的概率是 p ，那么现在丢失数据的概率还是 p 。——这个描述是不是有些小问题？前半段说磁盘数量增加一倍，后半段又说磁盘数量不变。

展开 ▾

作者回复: 后面描述过于含糊，应该是：但由于集群的磁盘数量不变

◀ ▶

💬 👍