05丨文件系统原理：如何用1分钟遍历一个100TB的文件?

2019-11-27 李智慧

后端技术面试38讲 进入课程 >

讲述：李智慧
时长 13:25 大小 12.30M

文件及硬盘管理是计算机操作系统的重要组成部分，让微软走上成功之路的正是微软最早推出的个人电脑 PC 操作系统，这个操作系统就叫 DOS，即 Disk Operating System，硬盘操作系统。我们每天使用电脑都离不开硬盘，硬盘既有大小的限制，通常大一点的硬盘也不过几 T，又有速度限制，快一点的硬盘也不过每秒几百 M。

文件是存储在硬盘上的，文件的读写访问速度必然受到硬盘的物理限制，那么如何才能 1 分钟完成一个 100T 大文件的遍历呢?

想要知道这个问题的答案，我们就必须知道文件系统的原理。

做软件开发时，必然要经常和文件系统打交道，而文件系统也是一个软件，了解文件系统的设计原理，可以帮助我们更好地使用文件系统，另外设计文件系统时的各种考量，也对我们自己做软件设计有诸多借鉴意义。

让我们先从硬盘的物理结构说起。

硬盘

硬盘是一种可持久保存、多次读写数据的存储介质。硬盘的形式主要两种，一种是机械式硬盘，一种是固态硬盘。

机械式硬盘的结构，主要包含盘片、主轴、磁头臂，主轴带动盘片高速旋转，当需要读写盘上的数据的时候，磁头臂会移动磁头到盘片所在的磁道上，磁头读取磁道上的数据。读写数据需要移动磁头，这样一个机械的动作，至少需要花费数毫秒的时间，这是机械式硬盘访问延迟的主要原因。

如果一个文件的数据在硬盘上不是连续存储的，比如数据库的 B+ 树文件，那么要读取这个文件，磁头臂就必须来回移动，花费的时间必然很长。如果文件数据是连续存储的，比如日志文件，那么磁头臂就可以较少移动，相比离散存储的同样大小的文件，连续存储的文件的读写速度要快得多。

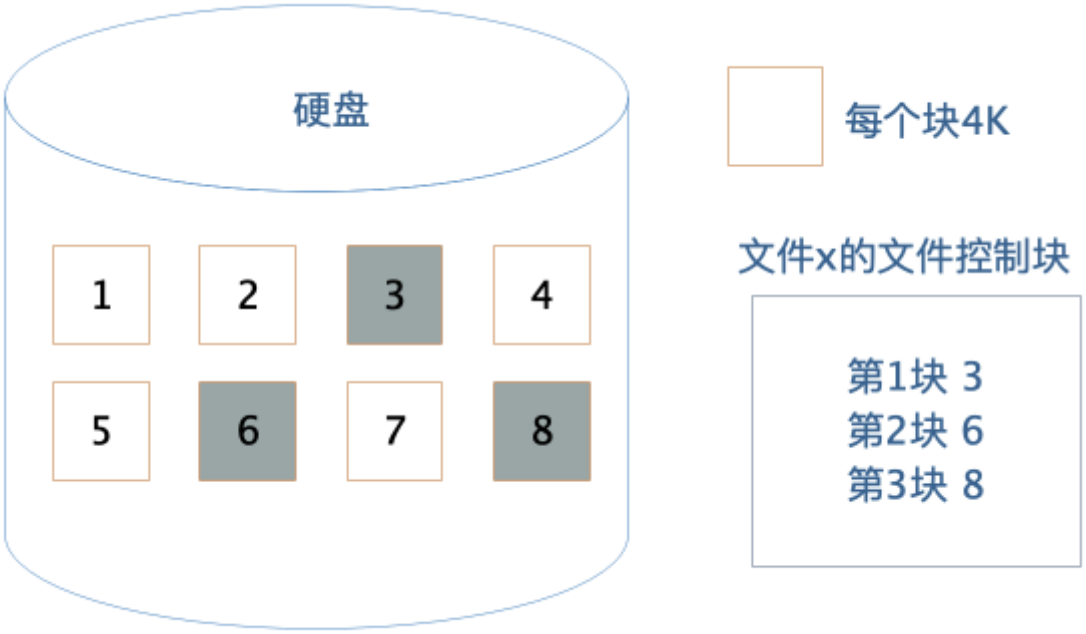
机械式硬盘的数据就存储在具有磁性特质的盘片上，因此这种硬盘也被称为磁盘，而固态硬盘则没有这种磁性特质的存储介质，也没有电机驱动的机械式结构。

其中主控芯片处理端口输入的指令和数据，然后控制闪存颗粒进行数据读写。由于固态硬盘没有了机械式硬盘的电机驱动磁头臂进行机械式物理移动的环节，而是完全的电子操作，因此固态硬盘的访问速度远快于机械式硬盘。

但是，到目前为止固态硬盘的成本还是明显高于机械式硬盘，因此在生产环境中，最主要的存储介质依然是机械式硬盘。如果一个场景对数据访问速度、存储容量、成本都有较高要求，那么可以采用固态硬盘和机械式硬盘混合部署的方式，即在一台服务器上既有固态硬盘，也有机械式硬盘，以满足不同文件类型的存储需求，比如日志文件存储在机械式硬盘上，而系统文件和随机读写的文件存储在固态硬盘上。

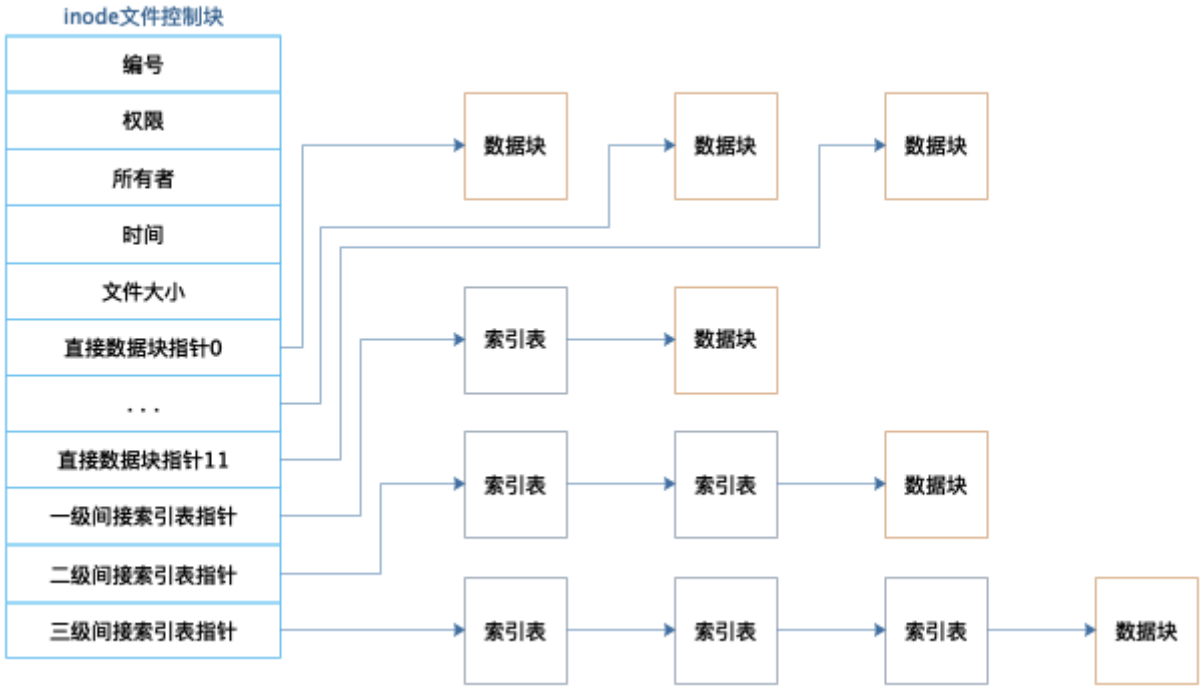
文件系统

作为应用程序开发者，我们不需要直接操作硬盘，而是通过操作系统，以文件的方式对硬盘上的数据进行读写访问。文件系统将硬盘空间以块为单位进行划分，每个文件占据若干个块，然后再通过一个文件控制块 FCB 记录每个文件占据的硬盘数据块。



这个文件控制块在 Linux 操作系统中就是 inode，要想访问文件，就必须获得文件的 inode 信息，在 inode 中查找文件数据块索引表，根据索引中记录的硬盘地址信息访问硬盘，读写数据。

inode 中记录着文件权限、所有者、修改时间和文件大小等文件属性信息，以及文件数据块硬盘地址索引。inode 是固定结构的，能够记录的硬盘地址索引数也是固定的，只有 15 个索引。其中前 12 个索引直接记录数据块地址，第 13 个索引记录索引地址，也就是说，索引块指向的硬盘数据块并不直接记录文件数据，而是记录文件数据块的索引表，每个索引表可以记录 256 个索引；第 14 个索引记录二级索引地址，第 15 个索引记录三级索引地址，如下图：



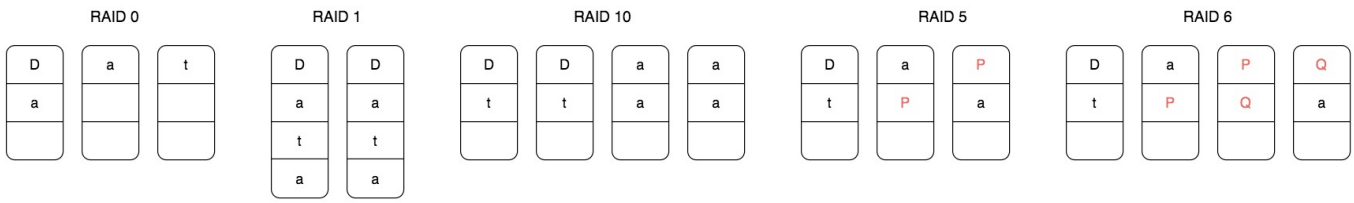
这样，每个 inode 最多可以存储 12+256+256256+256256*256 个数据块，如果每个数据块的大小为 4k，也就是单个文件最大不超过 70G，而且即使可以扩大数据块大小，文件大小也要受单个硬盘容量的限制。这样的话，对于我们开头提出的一分钟完成 100T 大文件的遍历，Linux 文件系统是无法完成的。

那么，有没有更给力的解决方案呢?

RAID

RAID，即独立硬盘冗余阵列，将多块硬盘通过硬件 RAID 卡或者软件 RAID 的方案管理起来，使其共同对外提供服务。RAID 的核心思路其实是利用文件系统将数据写入硬盘中不同数据块的特性，将多块硬盘上的空闲空间看做一个整体，进行数据写入，也就是说，一个文件的多个数据块可能写入多个硬盘。

根据硬盘组织和使用方式不同，常用 RAID 有五种，分别是 RAID 0、RAID 1、RAID 10、RAID 5 和 RAID 6。



RAID 0 将一个文件的数据分成 N 片，同时向 N 个硬盘写入，这样单个文件可以存储在 N 个硬盘上，文件容量可以扩大 N 倍，（理论上）读写速度也可以扩大 N 倍。但是使用 RAID 0 的最大问题是文件数据分散在 N 块硬盘上，任何一块硬盘损坏，就会导致数据不完整，整个文件系统全部损坏，文件的可用性极大地降低了。

RAID 1 则是利用两块硬盘进行数据备份，文件同时向两块硬盘写入，这样任何一块硬盘损坏都不会出现文件数据丢失的情况，文件的可用性得到提升。

RAID 10 结合 RAID 0 和 RAID 1，将多块硬盘进行两两分组，文件数据分成 N 片，每个分组写入一片，每个分组内的两块硬盘再进行数据备份。这样既扩大了文件的容量，又提高了文件的可用性。但是这种方式硬盘的利用率只有 50%，有一半的硬盘被用来做数据备份。

RAID 5 针对 RAID 10 硬盘浪费的情况，将数据分成 N-1 片，再利用这 N-1 片数据进行位运算，计算一片校验数据，然后将这 N 片数据写入 N 个硬盘。这样任何一块硬盘损坏，都可以利用校验片的数据和其他数据进行计算得到这片丢失的数据，而硬盘的利用率也提高到 N-1/N。

RAID 5 可以解决一块硬盘损坏后文件不可用的问题，那么如果两块文件损坏？RAID 6 的解决方案是，用两种位运算校验算法计算两片校验数据，这样两块硬盘损坏还是可以计算得到丢失的数据片。

实践中，使用最多的是 RAID 5，数据被分成 N-1 片并发写入 N-1 块硬盘，这样既可以得到较好的硬盘利用率，也能得到很好的读写速度，同时还能保证较好的数据可用性。使用 RAID 5 的文件系统比简单的文件系统文件容量和读写速度都提高了 N-1 倍，但是一台服务器上能插入的硬盘数量是有限的，通常是 8 块，也就是文件读写速度和存储容量提高了 7 倍，这远远达不到 1 分钟完成 100T 文件的遍历要求。

那么，有没有更给力的解决方案呢？

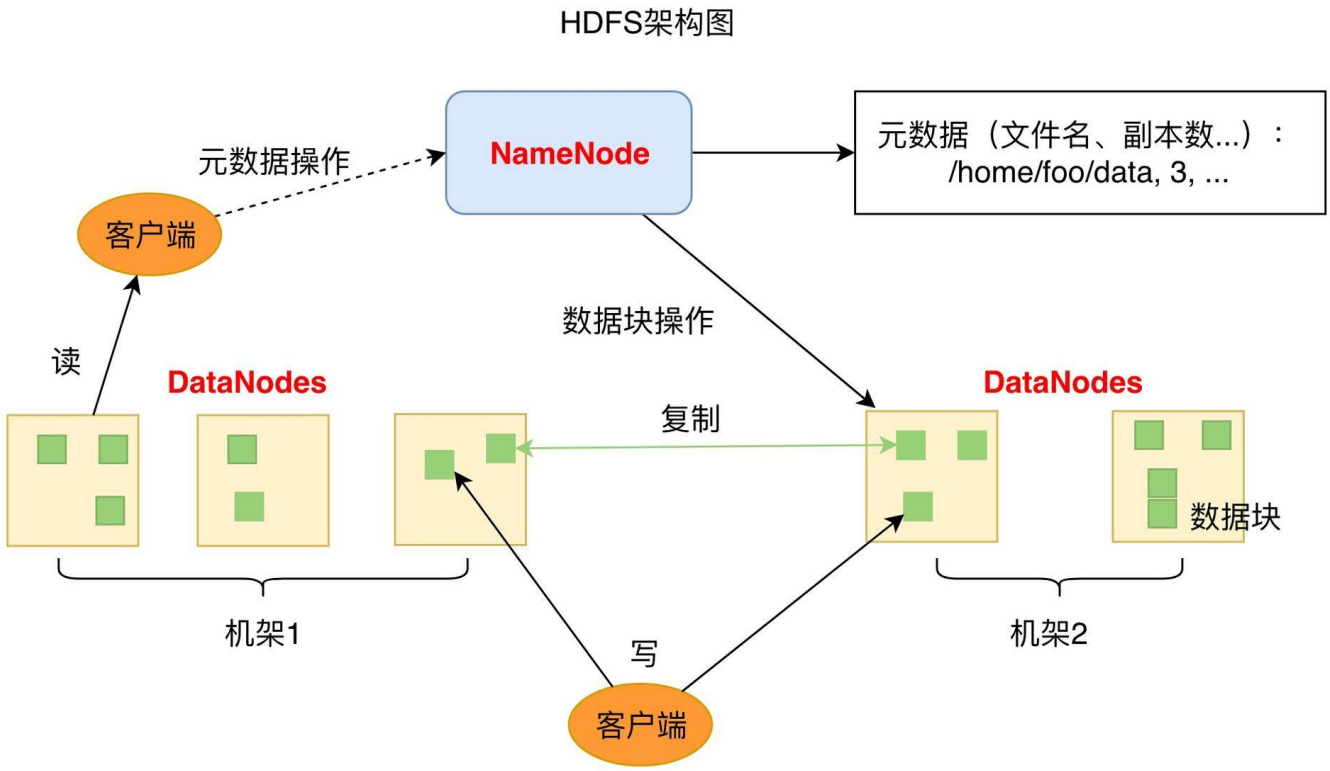
分布式文件系统

我们再回过头看下 Linux 的文件系统：文件的基本信息，也就是文件元信息记录在文件控制块 inode 中，文件的数据记录在硬盘的数据块中，inode 通过索引记录数据块的地址，读写文件的时候，查询 inode 中的索引记录得到数据块的硬盘地址，然后访问数据。

如果将数据块的地址改成分布式服务器的地址呢？也就是查询得到的数据块地址不只是本机的硬盘地址，还可以是其他服务器的地址，那么文件的存储容量就将是整个分布式服务器集群的硬盘容量，这样还可以在不同的服务器上同时并行读取文件的数据块，文件访问速度也将极大的加快。

这样的文件系统就是分布式文件系统，分布式文件系统的思路其实和 RAID 是一脉相承的，就是将数据分成很多片，同时向 N 台服务器上进行数据写入。针对一片数据丢失就导致整个文件损坏的情况，分布式文件系统也是采用数据备份的方式，将多个备份数据片写入多个服务器，以保证文件的可用性。当然，也可以采用 RAID 5 的方式通过计算校验数据片的方式提高文件可用性。

我们以 Hadoop 分布式文件系统 HDFS 为例，看下分布式文件系统的具体架构设计。



HDFS 的关键组件有两个，一个是 DataNode，一个是 NameNode。

DataNode 负责文件数据的存储和读写操作，HDFS 将文件数据分割成若干数据块（Block），每个 DataNode 存储一部分数据块，这样文件就分布存储在整个 HDFS 服务器集群中。应用程序客户端（Client）可以并行对这些数据块进行访问，从而使得 HDFS 可以在服务器集群规模上实现数据并行访问，极大地提高了访问速度。在实践中，HDFS 集群

的 DataNode 服务器会有很多台，一般在几百台到几千台这样的规模，每台服务器配有数块硬盘，整个集群的存储容量大概在几 PB 到数百 PB。

NameNode 负责整个分布式文件系统的元数据（MetaData）管理，也就是文件路径名、访问权限、数据块的 ID 以及存储位置等信息，相当于 Linux 系统中 inode 的角色。HDFS 为了保证数据的高可用，会将一个数据块复制为多份（缺省情况为 3 份），并将多份相同的数据块存储在不同的服务器上，甚至不同的机架上。这样当有硬盘损坏，或者某个 DataNode 服务器宕机，甚至某个交换机宕机，导致其存储的数据块不能访问的时候，客户端会查找其备份的数据块进行访问。

有了 HDFS，可以实现单一文件存储几百 T 的数据，再配合大数据计算框架 MapReduce 或者 Spark，可以对这个文件的数据块进行并发计算。也可以使用 Impala 这样的 SQL 引擎对这个文件进行结构化查询，在数千台服务器上并发遍历 100T 的数据，1 分钟都是绰绰有余的。

小结

文件系统从简单操作系统文件，到 RAID，再到分布式文件系统，其设计思路其实是具有统一性的。这种统一性一方面体现在文件数据如何管理，也就是如何通过文件控制块管理文件的数据，这个文件控制块在 Linux 系统中就是 inode，在 HDFS 中就是 NameNode。

另一方面体现在如何利用更多的硬盘实现越来越大的文件存储需求和越来越快的读写速度需求，也就是将数据分片后同时写入多块硬盘。单服务器我们可以通过 RAID 来实现，多服务器则可以将这些服务器组成一个文件系统集群，共同对外提供文件服务，这时候，数千台服务器的数万块硬盘以单一存储资源的方式对文件使用者提供服务，也就是一个文件可以存储数百 T 的数据，并在一分钟完成遍历这样一个大文件的遍历。

思考题

在 RAID 5 的示意图中，P 表示校验位数据，我们看到 P 不是单独存储在一块硬盘上，而是分散在不同的盘上，实际上，校验数据 P 的存储位置是螺旋式地落在所有硬盘上的，为什么要这样设计？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

上一篇　04 | 网络编程原理：一个字符的互联网之旅

下一篇　06 | 数据库原理：为什么PrepareStatement性能更好更安全?

精选留言 (14)

写留言

miracle
2019-11-27

老师可以每篇最后问答下上篇文章结尾留下的问题吗

展开

2　3

vouv
2019-11-27

校验数据分散存储在不同磁盘上最主要目的是为了提高并发IO

3

golangboy
2019-11-27

老师讲的透彻，成体系，感谢！分布式存储对数据的读写，都要经过元数据节点，此后的

数据读写能力会提升很多。但元数据节点应该有性能瓶颈问题，找的过程会限制读写能力，请教老师，这种一般怎么处理？

作者回复: 元数据节点NameNode只提供类似文件控制块的数据读写，数据量非常小，不会成为瓶颈。一个数据块Block大小64M，对应的NameNode控制块数据大概只有几十个字节。

3

Geek_22cbf4
2019-11-28

老师，针对校验数据的生成过程还是不太理解！能帮忙解释的详细一些么？

作者回复: 就是异或运算

所有数据的bit位，逐位进行异或，得到的就是校验位。

如果丢失部分数据，用校验数据和其余数据逐位进行异或运算，可到丢失部分数据。

举例，5块磁盘做RAID5，四块磁盘上的bit为：0 1 1 1 ，那么异或计算后，校验位为 1，如果丢失了第一块盘上的bit位0，那么校验位1和其他三块盘上的bit位进行异或运算，可以算出0

1 1

奔奔奔跑
2019-11-27

老师好，关于为什么P分散存储的问题我认为原因有以下两点:
1.高可用，避免检验盘损坏了所有都用不了了。
2.读取速度快，实现了检验数据的并行访问，大大加快了检验速度。

展开

1

禾斗君
2019-11-27

主要为了优化数据读取吧，如果校验码都放在同一块硬盘上，那么业务数据读取只有N-1块硬盘可以提供服务。 采用螺旋式分布时，N块硬盘都可以提供服务。

展开

1

龙哥

2019-11-27

我觉得螺旋存储校验位是为了提高磁盘使用率，校验位应该比数据块要小。如果校验位只存一块，应该会有数据盘满了，而校验盘还有一大块空间的情况

展开

1

星辰

2019-11-30

针对校验数据的螺旋式存储 和 真正的数据存储一样 ， 也是为了防止某块硬盘坏了 在丢了校验数据 和 某部分真实数据的时候 可以通过其他硬盘上的这部分备份 或 校验数据 来恢复吧

俊伟

2019-11-29

老师，RAID那里图没看懂，D，a，t，p，Q都是什么意思？图有点没太明白。

作者回复: d a t a表示需要写入RAID的数据，p q表示两种不同校验算法得到的校验数据。

老王的老李头

2019-11-29

我觉得题目改成，如何完整的将100T的数据存起来，更搭

丁丁历险记

2019-11-28

为啥我的硬盘是顺序读3700M

展开

扬帆丶启航

2019-11-28

校验数据P存储在所有硬盘上，这样每一块数据丢失后都能通过校验数据与其他硬盘的数据进行运算获得丢失的数据。

展开

老男孩
2019-11-28

突然发现专栏的名字好像变了？😁这估计是平台改的，这个名字目的性更强一些吧。今天的内容，老师从文件系统到RAID再到分布式文件系统讲解很系统也很全面。这里我有个问题想问一下，在分布式文件系统中，一个文件被分成多个数据块保存在不同datanode上，而且对这些数据块进行了备份。那么我们是否可以直接用RAID 0的方式把单节点的读写速度扩大N倍？还是采用RAID 5在速度和容错性之间做一个权衡？

展开

> 作者回复: HDFS缺省的高可用策略是RAID0，数据会做多个备份，应用可以指定备份数，如果想要加快读的速度，可以增加备份个数。


苏志辉
2019-11-27

因为是每行的校验所以一行一个，为了防止一个坏了影响最小，所以每一块一个