06 | 新技术层出不穷，HDFS依然是存储的王者

2018-11-10 李智慧

从0开始学大数据 进入课程 >

讲述：李智慧
时长 13:38 大小 6.26M

我们知道，Google 大数据“三驾马车”的第一驾是 GFS（Google 文件系统），而 Hadoop 的第一个产品是 HDFS，可以说分布式文件存储是分布式计算的基础，也可见分布式文件存储的重要性。如果我们将大数据计算比作烹饪，那么数据就是食材，而 Hadoop 分布式文件系统 HDFS 就是烧菜的那口大锅。

厨师来来往往，食材进进出出，各种菜肴层出不穷，而不变的则是那口大锅。大数据也是如此，这些年来，各种计算框架、各种算法、各种应用场景不断推陈出新，让人眼花缭乱，但是大数据存储的王者依然是 HDFS。

为什么 HDFS 的地位如此稳固呢？在整个大数据体系里面，最宝贵、最难以代替的资产就是数据，大数据所有的一切都要围绕数据展开。HDFS 作为最早的大数据存储系统，存储着宝贵的数据资产，各种新的算法、框架要想得到人们的广泛使用，必须支持 HDFS 才能获

取已经存储在里面的数据。所以大数据技术越发展，新技术越多，HDFS 得到的支持越多，我们越离不开 HDFS。**HDFS 也许不是最好的大数据存储技术，但依然最重要的大数据存储技术。**

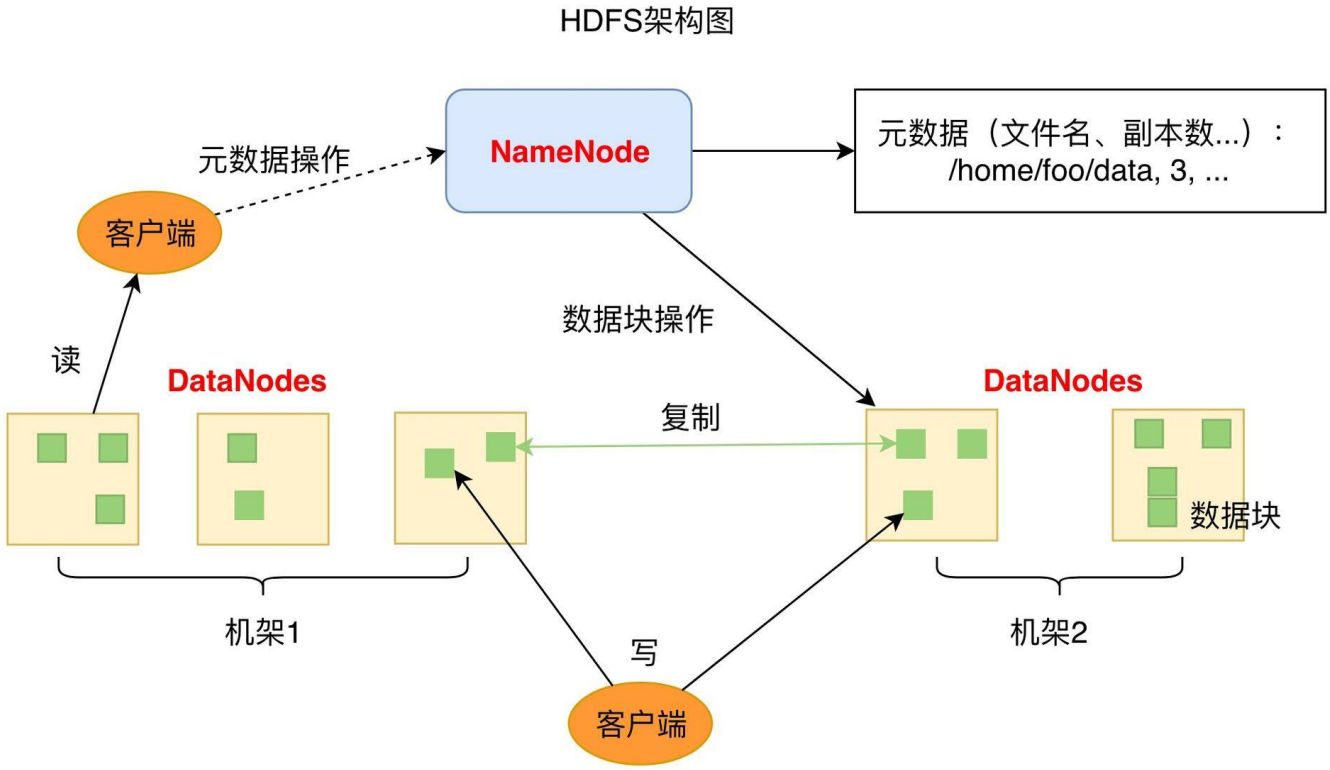
那我们就从 HDFS 的原理说起，今天我们来聊聊HDFS 是如何实现大数据高速、可靠的存储和访问的。

Hadoop 分布式文件系统 HDFS 的设计目标是管理数以千计的服务器、数以万计的磁盘，将这么大规模的服务器计算资源当作一个单一的存储系统进行管理，对应用程序提供数以 PB 计的存储容量，让应用程序像使用普通文件系统一样存储大规模的文件数据。

如何设计这样一个分布式文件系统？其实思路很简单。

我们先复习一下专栏上一期，我讲了 RAID 磁盘阵列存储，RAID 将数据分片后在多块磁盘上并发进行读写访问，从而提高了存储容量、加快了访问速度，并通过数据的冗余校验提高了数据的可靠性，即使某块磁盘损坏也不会丢失数据。将 RAID 的设计理念扩大到整个分布式服务器集群，就产生了分布式文件系统，Hadoop 分布式文件系统的核心原理就是如此。

和 RAID 在多个磁盘上进行文件存储及并行读写的思路一样，HDFS 是在一个大规模分布式服务器集群上，对数据分片后进行并行读写及冗余存储。因为 HDFS 可以部署在一个比较大的服务器集群上，集群中所有服务器的磁盘都可供 HDFS 使用，所以整个 HDFS 的存储空间可以达到 PB 级容量。



上图是 HDFS 的架构图，从图中你可以看到 HDFS 的关键组件有两个，一个是 DataNode，一个是 NameNode。

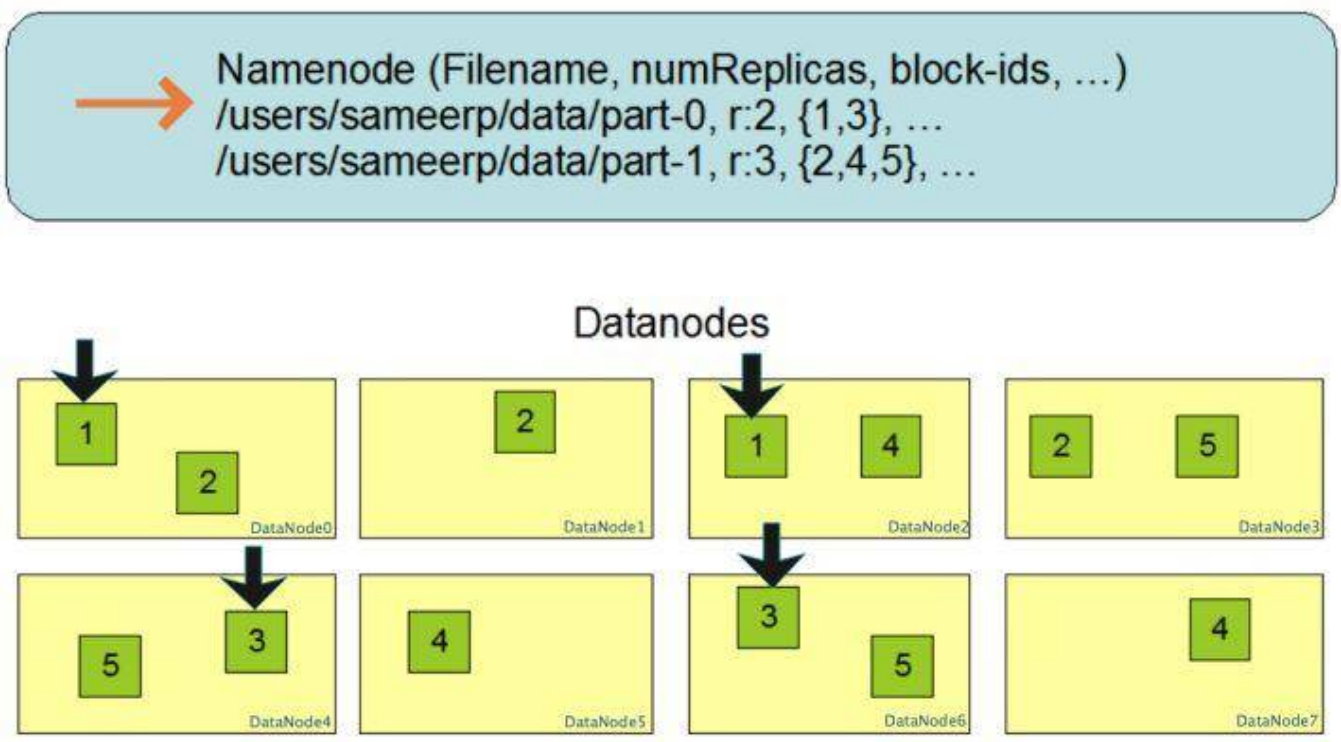
DataNode 负责文件数据的存储和读写操作，HDFS 将文件数据分割成若干数据块（Block），每个 DataNode 存储一部分数据块，这样文件就分布存储在整个 HDFS 服务器集群中。应用程序客户端（Client）可以并行对这些数据块进行访问，从而使得 HDFS 可以在服务器集群规模上实现数据并行访问，极大地提高了访问速度。

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磁盘，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、数据块的 ID 以及存储位置等信息，相当于操作系统中文件分配表（FAT）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。这样当有磁盘损坏，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。

下面这张图是数据块多份复制存储的示意，图中对于文件 /users/sameerp/data/part-0，其复制备份数设置为 2，存储的 BlockID 分别为 1、3。Block1 的两个备份存储在

DataNode0 和 DataNode2 两个服务器上，Block3 的两个备份存储 DataNode4 和 DataNode6 两个服务器上，上述任何一台服务器宕机后，每个数据块都至少还有一个备份存在，不会影响对文件 /users/sameerp/data/part-0 的访问。



和 RAID 一样，数据分成若干数据块后存储到不同服务器上，可以实现数据大容量存储，并且不同分片的数据可以并行进行读 / 写操作，进而实现数据的高速访问。你可以看到，HDFS 的大容量存储和高速访问相对比较容易实现，但是 HDFS 是如何保证存储的高可用性呢?

我们尝试从不同层面来讨论一下 HDFS 的高可用设计。

1. 数据存储故障容错

磁盘介质在存储过程中受环境或者老化影响，其存储的数据可能会出现错乱。HDFS 的应对措施是，对于存储在 DataNode 上的数据块，计算并存储校验和（CheckSum）。在读取数据的时候，重新计算读取出来的数据的校验和，如果校验不正确就抛出异常，应用程序捕获异常后就到其他 DataNode 上读取备份数据。

2. 磁盘故障容错

如果 DataNode 监测到本机的某块磁盘损坏，就将该块磁盘上存储的所有 BlockID 报告给 NameNode，NameNode 检查这些数据块还在哪些 DataNode 上有备份，通知相应的 DataNode 服务器将对应的数据块复制到其他服务器上，以保证数据块的备份数满足要求。

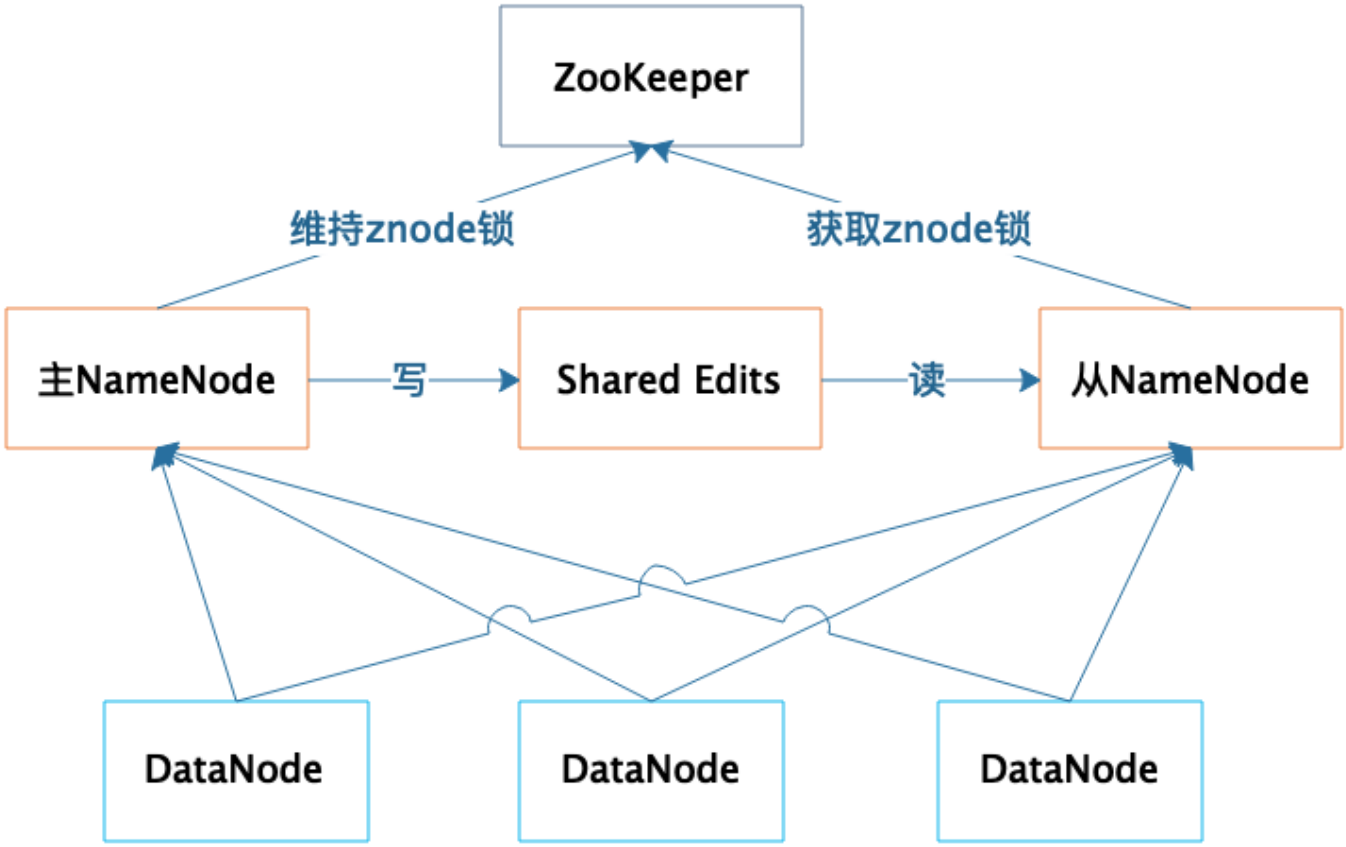
3.DataNode 故障容错

DataNode 会通过心跳和 NameNode 保持通信，如果 DataNode 超时未发送心跳，NameNode 就会认为这个 DataNode 已经宕机失效，立即查找这个 DataNode 上存储的数据块有哪些，以及这些数据块还存储在哪些服务器上，随后通知这些服务器再复制一份数据块到其他服务器上，保证 HDFS 存储的数据块备份数符合用户设置的数目，即使再出现服务器宕机，也不会丢失数据。

4.NameNode 故障容错

NameNode 是整个 HDFS 的核心，记录着 HDFS 文件分配表信息，所有的文件路径和数据块存储信息都保存在 NameNode，如果 NameNode 故障，整个 HDFS 系统集群都无法使用；如果 NameNode 上记录的数据丢失，整个集群所有 DataNode 存储的数据也就没用了。

所以，NameNode 高可用容错能力非常重要。NameNode 采用主从热备的方式提供高可用服务，请看下图。



集群部署两台 NameNode 服务器，一台作为主服务器提供服务，一台作为从服务器进行热备，两台服务器通过 ZooKeeper 选举，主要是通过争夺 znode 锁资源，决定谁是主服务器。而 DataNode 则会向两个 NameNode 同时发送心跳数据，但是只有主 NameNode 才能向 DataNode 返回控制信息。

正常运行期间，主从 NameNode 之间通过一个共享存储系统 shared edits 来同步文件系统的元数据信息。当主 NameNode 服务器宕机，从 NameNode 会通过 ZooKeeper 升级成为主服务器，并保证 HDFS 集群的元数据信息，也就是文件分配表信息完整一致。

对于一个软件系统而言，性能差一点，用户也许可以接受；使用体验差，也许也能忍受。但是如果可用性差，经常出故障导致不可用，那就比较麻烦了；如果出现重要数据丢失，那开发工程师绝对是摊上大事了。

而分布式系统可能出故障地方又非常多，内存、CPU、主板、磁盘会损坏，服务器会宕机，网络会中断，机房会停电，所有这些都可能会引起软件系统的不可用，甚至数据永久丢失。

所以在设计分布式系统的时候，软件工程师一定要绷紧可用性这根弦，思考在各种可能的故障情况下，如何保证整个软件系统依然是可用的。

根据我的经验，一般说来，常用的保证系统可用性的策略有冗余备份、失效转移和降级限流。虽然这 3 种策略你可能早已耳熟能详，但还是有一些容易被忽略的地方。

比如**冗余备份**，任何程序、任何数据，都至少要有一个备份，也就是说程序至少要部署到两台服务器，数据至少要备份到另一台服务器上。此外，稍有规模的互联网企业都会建设多个数据中心，数据中心之间互相进行备份，用户请求可能会被分发到任何一个数据中心，即所谓的异地多活，在遭遇地域性的重大故障和自然灾害的时候，依然保证应用的高可用。

当要访问的程序或者数据无法访问时，需要将访问请求转移到备份的程序或者数据所在的服务器上，这也就是**失效转移**。失效转移你应该注意的是失效的鉴定，像 NameNode 这样主从服务器管理同一份数据的场景，如果从服务器错误地以为主服务器宕机而接管集群管理，会出现主从服务器一起对 DataNode 发送指令，进而导致集群混乱，也就是所谓的“脑裂”。这也是这类场景选举主服务器时，引入 ZooKeeper 的原因。ZooKeeper 的工作原理，我将会在后面专门分析。

当大量的用户请求或者数据处理请求到达的时候，由于计算资源有限，可能无法处理如此大量的请求，进而导致资源耗尽，系统崩溃。这种情况下，可以拒绝部分请求，即进行**限流**；也可以关闭部分功能，降低资源消耗，即进行**降级**。限流是互联网应用的常备功能，因为超出负载能力的访问流量在何时会突然到来，你根本无法预料，所以必须提前做好准备，当遇到突发高峰流量时，就可以立即启动限流。而降级通常是为可预知的场景准备的，比如电商的“双十一”促销，为了保障促销活动期间应用的核心功能能够正常运行，比如下单功能，可以对系统进行降级处理，关闭部分非重要功能，比如商品评价功能。

小结

我们小结一下，看看 HDFS 是如何通过大规模分布式服务器集群实现数据的大容量、高速、可靠存储、访问的。

1. 文件数据以数据块的方式进行切分，数据块可以存储在集群任意 DataNode 服务器上，所以 HDFS 存储的文件可以非常大，一个文件理论上可以占据整个 HDFS 服务器集群上的所有磁盘，实现了大容量存储。

2.HDFS 一般的访问模式是通过 MapReduce 程序在计算时读取，MapReduce 对输入数据进行分片读取，通常一个分片就是一个数据块，每个数据块分配一个计算进程，这样就可以同时启动很多进程对一个 HDFS 文件的多个数据块进行并发访问，从而实现数据的高速访问。关于 MapReduce 的具体处理过程，我们会在专栏后面详细讨论。

3.DataNode 存储的数据块会进行复制，使每个数据块在集群里有多个备份，保证了数据的可靠性，并通过一系列的故障容错手段实现 HDFS 系统中主要组件的高可用，进而保证数据和整个系统的高可用。

思考题

今天留一道有意思的思考题，你可以先想象一个场景，我们想利用全世界的个人电脑、手机、平板上的空闲存储空间，构成一个可以付费共享的分布式文件系统，希望用户可以安装一个 App 在自己的个人设备上，将个人资料安全地存储到这个分布式文件系统中，并支付一定费用；用户也可以用这个 App 将自己设备上的空闲存储空间共享出去，成为这个分布式文件系统存储的一部分，并收取一定费用。

我想问你的是，如果是你来设计这个分布式文件系统，你是怎么思考的？你的设计方案是什么？

欢迎你写下自己的思考或疑问，与我和其他同学一起讨论。

上一篇　05 | 从RAID看垂直伸缩到水平伸缩的演化

精选留言 (66)

写留言

上个纪元的...
2018-11-10

38

听过本期音频，我想，在现实的条件下，实现这样的设想非常困难，例如：【1】用户空间（尤其是手机，iPad）不能保障高可用的性能，随时被访问被验证；【2】网络条件要求过高，尤其是被需求或者需要均衡时频繁的文件迁移；【3】要验证HDFS所有备份块的可用性，因此个人中端上不能过多不同用户，过碎的数据块；【4】为了保证系统的高效一般一块数据不会过小，要不然会浪费过多的计算资源（进程），如果单块数据在128M左右，...

展开

作者回复: 很周全

方得始终
2018-11-10

11

最近两大Hadoop发行商Cloudera 和Hortonworksg合并,网上包括极客时间也报道过HDFS跟云端对象存储系统已没有性能和价格上的优势.在我工作实践中也碰见过HDFS上存储的文件丢失,虽然只是一个机架(rack)断电.请问老师对此有何看法?

文大头
2018-11-12

10

1、互联网上用户交分散，需要用CDN的模式，分层分区域部署NameNode，NameNode上数据汇总到上级，上级数据按需分发到下级。同一个区域的用户（DataNode）分配到同一个NameNode
2、用户DataNode机器可用性极差，按10%算，平均一个数据需要10个备份。不过可以有一定策略改进，比如让用户活跃时间跟用户等级挂钩，等级跟功能挂钩，以鼓励用户增...

展开

作者回复: 想的很周全

落叶飞逝的...
2018-11-12

7

关于思考题的想法:首先这个就是各大厂商的提出的云服务的概念。而对于手机、ipad的这些设备作为分布式容器的一部分，是不可取的。首先不能不确保手机的网速的可用性，而且三大运营商都有流量这个概念。第二，手机无法实时的给NameNode进行发送心跳，因为用户可以主动关闭网络，或者用户在无网区域。

展开

Zach_
2018-11-26

3

老师、我想提一个问题： 主NameNode 向 Sared Edits 写数据的过程中、主Namenade没洗完的数据是不是会丢失？ 那这样 从NameNode被选举为主NameNode 后，是不是会有一部分数据找不见存在哪个DataNode上了？ 大家都可以回答哈 另外 一个数据块 在什么情况下、不是一个分区？

朱国伟
2018-11-10

3

关于DataNode 故障容错感觉处理起来好复杂啊 假设numReplicas=2 由于机器故障导致DataNode 1宕机 此时为了保证numReplicas=2会复制数据 像下面的情况怎么处理呢
- 等全部复制完了 DataNode1重启了 那此时numReplicas=3 这种情况会处理吗?
- 或者复制到一半（即有些Block还没来得及复制） DataNode1重启了 这种情况又怎么办
- 或者集群勉强够用 实在没有多余的机器来复制DataNode1对应的数据了 又该如何...

展开

作者回复: DataNode即使重启也认为其上的数据已经丢失，不会再使用。

Jack Zhu
2018-11-24

2

“如果 DataNode 监测到本机的某块磁盘损坏，就将该块磁盘上存储的所有 BlockID 报告给NameNode。”
有个疑问，望解答:磁盘损坏，怎么获得BlockID，BlockID存在哪?

展开

鸠摩智
2018-11-12

2

如果在hdfs上存储了大量的小文件，每个文件都不到一个块（128M）大小。而每个文件确实是单独的，比如一张张图片，不能把它们合并为一个大文件，这样即使一个文件namenode只存储几字节的元数据，是不是也有可能超出namenode单台机器限制了呢?

往事随风，...
2018-11-10

2

可以先通过定位寻找离自己近的人的手机存储容量，如果不够就扩大搜索范围直到可以满足自己需求，然后给对应这些设备设置一个哈稀函数，怎么来存储对应分片数据，根据不同分钱到的对应设备上付钱，并且设置一个校验的设备，定时的去检测设备，哪些可用，哪些不可用，然后进行数据转移备份

Sam.张朝
2018-12-17

1

从第一篇看到第六篇，感觉脑袋有点装不下了。

展开

dingwood
2018-12-15

1

Shared edits是啥。。网上搜半天没找到，求指点

展开

blacksmith...
2018-11-21

1

讲技术不讲技术脉络的也都是流氓啊。那些去中心化存储的区块链项目，就没谁写出去中心存储应是借鉴或发展于hdfs等分布式存储方案。raid到hdfs立马好理解多了。我是看过ipfs，storj，sia等几个去中心化的存储方案。通过看今天的内容，我突然感觉开窍了，他们整得太复杂了，基于hdfs加上存储时空证明就能实现去中心化存储，实现高可用的技术细节考虑的当然不同了，而存储时空权益就把终端的高可用工作分散到具体用户了。当...

展开

作者回复: 深刻

wmg

1

2018-11-12

类似于hdfs的机制，只不过将用户设备作为datanode，namenode是中心化的（否则和区块链就比较类似）。有几个问题需要考虑：1.用户设备存储空间有限，所以block的大小不能太大；2.由于block不能太大所以元数据会比较大，对namenode存储空间要求较高；3.由于datanode是不可信的，所以需要对datanode设计身份识别机制，存储的数据也必须是加密，副本数量也要设置的多一些；4.由于所有的datanode都需要和namenode通信...

展开

作者回复: 很赞，不过NameNode不应该是存储空间的制约，该怎么办?

一步

2018-11-10

1

这个思考题的实现思路是和IPFS的实现思路应该一样的

展开

作者回复: 是的

姜文

2018-11-10

1

首先要部署个name node存储元数据，记录用户数据存储的位置，为保证name node的高可用，必须做备份，通过zookeeper选举主 name node，data node就是全世界的移动设备，用户的数据要做备份，至少三份，用户的app必须和name node的主备服务器做心跳，用于移动设备故障时能主动上报或者name node能及时发现保证数据可用。用户如果要存储数据必须通知name node做好元数据记录及datanode的数据备份。第一次回答...

展开

水电工⁹(...

2019-06-02

由于在这方面几乎是个小白，各种计算机原理并不理解，所以我就只能大概说说需要考虑哪些方面。
一是存储资料安全性，当个人设备空闲空间分享出去后，需要保证用户在这些分享空间存储资料的安全性、私密性，即不被他人破获。
二是对各区域的管理，假设这是一个全球性的系统，所以所有分享的空间以及申请存储...

展开

子榕
2019-05-12

老师请问下hdfs一般集群规模都会上百吗？那像我们小公司是不是没有意义啦，我们现在每天日志量100g左右，适合用hdfs来存储吗？单机采购什么规格的合适（几核几g多少t存储）？

展开

作者回复: 大数据对于硬件和环境没有约束和要求的，主要是看你的业务场景是否适合

gkb111
2019-04-24

数据冗余存储，并计算存储校验和，不一致则访问其他存储块

展开

雪候鸟
2019-04-22

看了您讲的这节，感觉分布式其实就是单机技术的横向扩展，现在流行的Tidb的底层的分布式存储的原理跟hdfs的底层存储方式感觉挺像，其实灵感都来源于RAID的磁盘存储原理。

展开

掌心童话
2019-04-22

这讲的诸多点让我想到了区块链，脑裂，不就是共识算法问题引起的经典拜占庭问题(•̀_•́)。这把客户端当数据存储节点也是各种币做交易记录数据库的玩儿法

展开