

愤怒老蛞鹬

博客园 :: 首页 :: 博文 :: 闪存 :: 新随笔 :: 联系 :: 订阅 [XML](#) :: 管理 ::

8 随笔 :: 0 文章 :: 0 评论 :: 0 引用

2018年12月

日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

公告

昵称: 愤怒老蛞鹬
园龄: 1年
粉丝: 0
关注: 1
[+加关注](#)

搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[spring\(4\)](#)
[大数据现状分析\(2\)](#)
[杂记\(1\)](#)
[hadoop\(1\)](#)

随笔档案

[2018年7月\(2\)](#)
[2018年6月\(2\)](#)
[2017年12月\(1\)](#)
[2017年11月\(3\)](#)

阅读排行榜

1. hadoop:hdfs的讲解(478)
2. wget window(107)
3. 大数据现状分析 (2) (33)
4. 大数据现状分析 (1) (31)
5. spring-Springmvc搭建 (二) 添加编码Filter, 添加spring配置文件(19)

hadoop:hdfs的讲解

转载自: <http://www.cnblogs.com/tgzhu/p/5788634.html>

在配置hbase集群将 hdfs 挂载到其它镜像盘时, 有不少困惑的地方, 结合以前的资料再次学习; 大数据底层技术的三大基石起源于Google在2006年之前的三篇论文GFS、Map-Reduce、Bigtable, 其中GFS、Map-Reduce技术直接支持了Apache Hadoop项目的诞生, Bigtable催生了NoSQL这个崭新的数据库领域, 由于map-Reduce处理框架高延时的缺陷, Google在2009年后推出的Dremel促使了实时计算系统的兴起, 以此引发大数据第二波技术浪潮, 一些大数据公司纷纷推出自己的大数据查询分析产品, 如: Cloudera开源了大数据查询分析引擎Impala、Hortonworks开源了 Stinger、Facebook开源了Presto、UC Berkeley AMPLAB实验室开发了Spark计算框架, 所有这些技术的数据来源均基于hdsf, 对于 hdsf 最基本的不外乎就是其读写操作

目录:

- hdfs 名词解释
- hdsf 架构
- NameNode(NN)
- Secondary NN
- hdfs 写文件
- hdfs 读文件
- block持续化结构

HDFS名词解释:

- Block: 在HDFS中, 每个文件都是采用的分块的方式存储, 每个block放在不同的datanode上, 每个block的标识是一个三元组 (block id, numBytes, generationStamp), 其中block id是具有唯一性, 具体分配是由namenode节点设置, 然后再由datanode上建立block文件, 同时建立对应block meta文件
- Packet: 在DFSClient与DataNode之间通信的过程中, 发送和接受数据过程都是以一个packet为基础的方式进行
- Chunk: 中文名字也可以称为块, 但是为了与block区分, 还是称之为chunk。在DFSClient与DataNode之间通信的过程中, 由于文件采用的是基于块的方式来进行的, 但是在发送数据的过程中是以packet的方式来进行的, 每个packet包含了多个chunk, 同时对于每个chunk进行checksum计算, 生成checksum bytes
- 小结:
 1. 一个文件被拆成多个block持续化存储 (block size 由配置文件参数决定) 思考: 修改 block size 对以前持续化的数据有何影响?
 2. 数据通讯过程中一个 block 被拆成 多个 packet
 3. 一个 packet 包含多个 chunk
- Packet结构与定义: Packet分为两类, 一类是实际数据包, 另一类是heartbeat包。一个Packet数据包的组成结构, 如图所示
- Packet

Packet Header

Packet Data

pkt Len	Offset In Block	SeqNo	Last Packet In Block	Data Len	Chunk Checksum 1	Chunk Checksum 2	...	Chunk Checksum N	Chunk Data 1	Chunk Data 2	...	Chunk Data N
4	8	8	1	4	4	4	...	4	512	512	...	512 (Maximum)
25					65532 (Maximum)							
65557 (Maximum)												

• 上图中, 一个Packet是由Header和数据两部分组成, 其中Header部分包含了一个Packet的概要属性信息, 如下表所示:

字段名称	字段类型	字段长度	字段含义
pktLen	int	4	4 + dataLen + checksumLen
offsetInBlock	long	8	Packet在Block中偏移量
seqNo	long	8	Packet序列号, 在同一个Block唯一
lastPacketInBlock	boolean	1	是否是一个Block的最后一个Packet
dataLen	int	4	dataPos - dataStart, 不包含Header和Checksum的长度

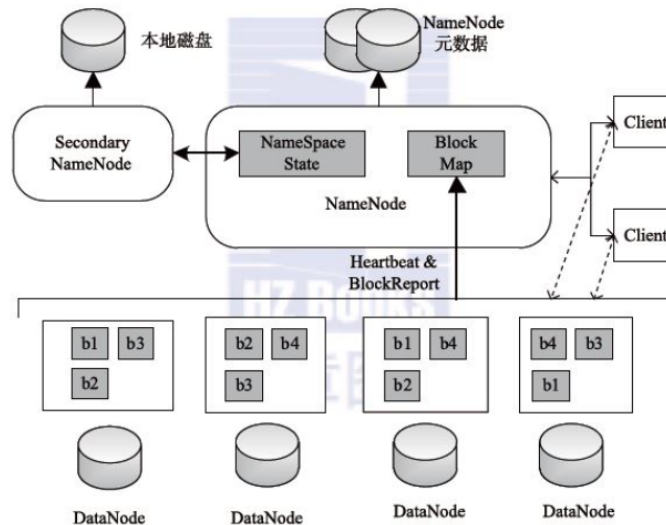
- Data部分是一个Packet的实际数据部分, 主要包括一个4字节校验和 (Checksum) 与一个Chunk部分, Chunk部分最大为512字节
- 在构建一个Packet的过程中, 首先将字节流数据写入一个buffer缓冲区中, 也就是从偏移量为25的位置 (checksumStart) 开始写Packet数据Chunk的Checksum部分, 从偏移量为533的位置 (dataStart) 开始写Packet数据的Chunk Data部分, 直到一个Packet创建完成为止。
- 当写一个文件的最后一个Block的最后一个Packet时, 如果一个Packet的大小未能达到最大长度, 也就是上图对应的缓冲区中, Checksum与Chunk Data之间还保留了一段未被写过的缓冲区位置, 在发送这个Packet之前, 会检查Checksum与Chunk Data之间的缓冲区是否为空白缓冲区 (gap), 如果有则将Chunk Data部分向前移动, 使得Chunk Data 1与Chunk Checksum N相邻, 然后才会被发送到DataNode节点

hdsf架构:

- hdfs的架构图网上一堆, 抓了一张表述比较清楚的图如下, 主要包含因类角色: Client、NameNode、SecondaryNameNode、DataNode

<https://www.cnblogs.com/super-sun/p/7879930.html>

1/8



- HDFS Client: 系统使用者, 调用HDFS API操作文件;与NN交互获取文件元数据;与DN交互进行数据读写, 注意: 写数据时文件切分由Client完成
- Namenode: Master节点 (也称元数据节点), 是系统唯一的管理者。负责元数据的管理(名称空间和数据块映射信息);配置副本策略;处理客户端请求
- Datanode: 数据存储节点(也称Slave节点), 存储实际的数据;执行数据块的读写;汇报存储信息给NN
- Secondary NameNode: 小弟角色, 分担大哥namenode的工作量;是NameNode的冷备份;合并fsimage和fsedits然后再发给namenode, 注意: 在hadoop 2.x 版本, 当启用 hdfs ha 时, 将没有这一角色。(详见第二单)
- 解释说明:

1. 热备份: b是a的热备份, 如果a坏掉。那么b马上运行代替a的工作
2. 冷备份: b是a的冷备份, 如果a坏掉。那么b不能马上代替a工作。但是b上存储a的一些信息, 减少a坏掉之后的损失

• hdfs构架原则:

1. 元数据与数据分离: 文件本身的属性 (即元数据) 与文件所持有的数据分离
2. 主/从架构: 一个HDFS集群是由一个NameNode和一定数目的DataNode组成
3. 一次写入多次读取: HDFS中的文件在任何时间只能有一个Writer。当文件被创建, 接着写入数据, 最后, 一旦文件被关闭, 就不能再修改。
4. 移动计算比移动数据更划算: 数据运算, 越靠近数据, 执行运算的性能就越好, 由于hdfs数据分布在不同机器上, 要让网络的消耗最低, 并提高系统的吞吐量, 最佳方式是将运算的执行移到离它要处理的数据更近的地方, 而不是移动数据

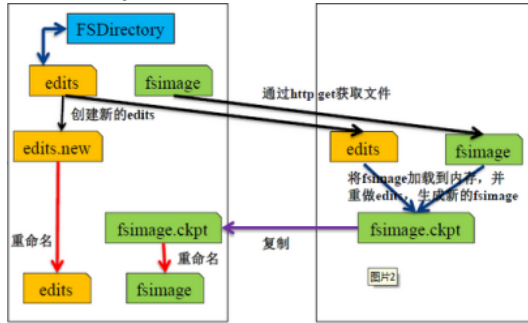
NameNode:

- NameNode是整个文件系统的管理节点, 也是HDFS中最复杂的一个实体, 它维护着HDFS文件系统中最重要的两个关系:
 1. HDFS文件系统中的文件目录树, 以及文件的数据块索引, 即每个文件对应的数据块列表
 2. 数据块和数据节点的对应关系, 即某一块数据块保存在哪些数据节点的信息
- 第一个关系即目录树、元数据和数据块的索引信息会持久化到物理存储中, 实现是保存在命名空间的镜像fsimage和编辑日志edits中, 注意: 在fsimage中, 并没有记录每一个block对应到哪几个Datanodes的对应表信息
- 第二个关系是在NameNode启动后, 每个Datanode对本地磁盘进行扫描, 将本Datanode上保存的block信息汇报给Namenode, Namenode在接收到每个Datanode的块信息汇报后, 将接收到的块信息, 以及其所在的Datanode信息等保存在内存中。HDFS就是通过这种块信息汇报的方式来完成 block -> Datanodes list的对应表构建
- fsimage记录了自最后一次检查点之前HDFS文件系统中所有目录和文件的序列化信息;
- edits是元数据操作日志(记录每次保存fsimage之后到下次保存之间的所有hdfs操作)
- 在NameNode启动时候, 会先将fsimage中的文件系统元数据信息加载到内存, 然后根据edits中的记录将内存中的元数据同步至最新状态, 将这个新版本的FsImage从内存中保存到本地磁盘上, 然后删除旧的Editlog, 这个过程称为一个检查点(checkpoint), 多长时间做一次checkpoint? (见第四章 参数配置) checkpoint 能手工触发吗? 验证重启hdfs服务后editlog没删除呢?
- 类似于数据库中的检查点, 为了避免edits日志过大, 在Hadoop1.X中, SecondaryNameNode会按照时间阈值(比如24小时)或者edits大小阈值(比如1G), 周期性的将fsimage和edits的合并, 然后将最新的fsimage推送给NameNode。而在Hadoop2.X中, 这个动作是由Standby NameNode来完成。
- 由此可看出, 这两个文件一旦损坏或丢失, 将导致整个HDFS文件系统不可用, 在HDP2.4安装(五): 集群及组件安装 集群安装过程中, hdfs 默认的只能选择一个NN, 是否意味着NN存在单点呢? (见第二单 hdfs HA)
- 在hadoop1.X为了保证这两种元数据文件的高可用性, 一般的做法, 将dfs.namenode.name.dir设置成以逗号分隔的多个目录, 这多个目录至少不要在一块磁盘上, 最好放在不同的机器上, 比如: 挂载一个共享文件系统
- fsimage和edits是序列化后的文件, 想要查看或编辑里面的内容, 可通过 hdfs 提供的 oiv/oev 命令, 如下:
 - 命令: hdfs oiv (offline image viewer) 用于将fsimage文件的内容转储到指定文件中以便于阅读, 如文本文件、XML文件, 该命令需要以下参数:
 - -i (必填参数) -inputFile <arg> 输入FSImage文件
 - -o (必填参数) -outputFile <arg> 输出转换后的文件, 如果存在, 则会覆盖
 - -p (可选参数) -processor <arg> 将FSImage文件转换成哪种格式: (Ls|XML|FileDistribution).默认为Ls
 - 示例: hdfs oiv -i /data1/hadoop/dfs/name/current/fsimage_0000000000019372521 -o /home/hadoop/fsimage.txt
 - 命令: hdfs oev (offline edits viewer 离线edits查看器) 的缩写, 该工具只操作文件因而并不需要hadoop集群处于运行状态。
 - 示例: hdfs oev -i edits_000000000000042778-000000000000042779 -o edits.xml
 - 支持的输出格式有binary (hadoop使用的二进制格式)、xml (在不使用参数p时的默认输出格式)和stats (输出edits文件的统计信息)
- 小结:

1. NameNode管理着DataNode, 接收DataNode的注册、心跳、数据块提交等信息的上报, 并且在心跳中发送数据块复制、删除、恢复等指令; 同时, NameNode还为客户端对文件系统目录树的操作和对文件数据读写、对HDFS系统进行管理提供支持
2. Namenode 启动后会进入一个称为安全模式的特殊状态。处于安全模式的 Namenode 是不会进行数据块的复制的。Namenode 从所有的 Datanode 接收心跳信号和块状态报告。块状态报告包括了某个 Datanode 所有的数据 块列表。每个数据块都有一个指定的最小副本数。当 Namenode 检测确认某 个数据块的副本数达到这个最小值, 那么该数据块就会被认为是副本安全 (safely replicated) 的; 在一定百分比 (这个参数可配置) 的数据块被 Namenode 检测确认是安全之后 (加上一个额外的 30 秒等待时间), Namenode 将退出安全模式状态。接下来它会确定还有哪些数据块的副本没 有达到指定数目, 并将这些数据块复制到其他 Datanode 上。

Secondary NameNode：在HA cluster中又称为standby node

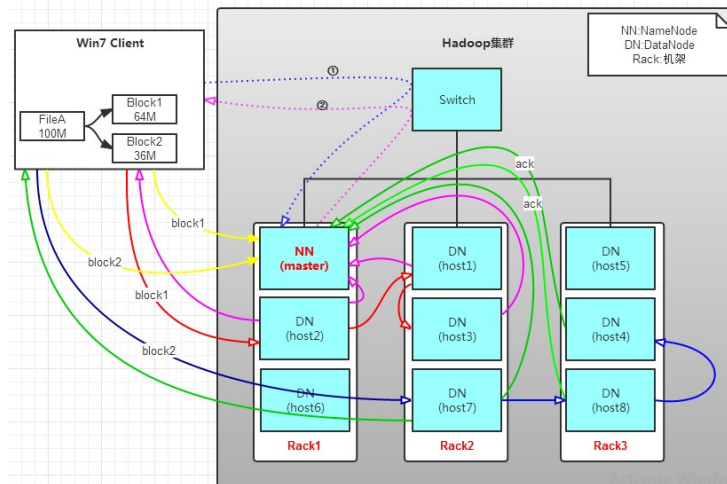
- 定期合并 fsimage 和 edits 日志，将 edits 日志文件大小控制在一个限度下



- namenode 响应 Secondary namenode 请求，将 edit log 推送给 Secondary namenode，开始重新写一个新的 edit log
- Secondary namenode 收到来自 namenode 的 fsimage 文件和 edit log
- Secondary namenode 将 fsimage 加载到内存，应用 edit log，并生成一个新的 fsimage 文件
- Secondary namenode 将新的 fsimage 推送给 Namenode
- Namenode 用新的 fsimage 取代旧的 fsimage，在 fsime 文件中记下检查点发生的时间

HDFS写文件：

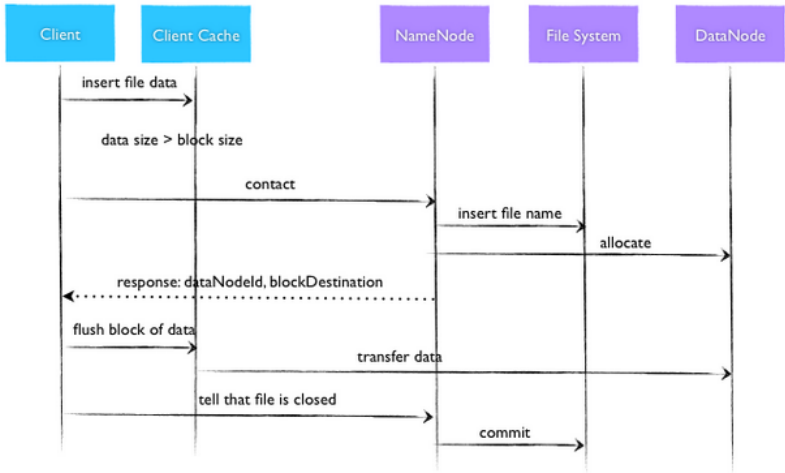
- 写文件部分参考blog 地址 (<http://www.cnblogs.com/laov/p/3434917.html>)，2.X版本默认block的大小是 128M（见第四章参数配置）



- Client将FileA按64M分块。分成两块，block1和Block2;
- Client向nameNode发送写数据请求，如图蓝色虚线①----->
- NameNode节点，记录block信息。并返回可用的DataNode (NameNode按什么规则返回DataNode? 参见第三单 hadoop机架感知)，如粉色虚线②----->
 - Block1: host2,host1,host3
 - Block2: host7,host8,host4
- client向DataNode发送block1；发送过程是以流式写入，流式写入过程如下：
 - 将64M的block1按64k的packet划分
 - 然后将第一个packet发送给host2
 - host2接收完后，将第一个packet发送给host1，同时client想host2发送第二个packet
 - host1接收完第一个packet后，发送给host3，同时接收host2发来的第二个packet
 - 以此类推，如图红线实线所示，直到将block1发送完毕
 - host2,host1,host3向NameNode，host2向Client发送通知，说“消息发送完了”。如图粉红颜色实线所示
 - client收到host2发来的消息后，向namenode发送消息，说我写完了。这样就真完成了。如图黄色粗实线
 - 发送完block1后，再向host7，host8，host4发送block2，如图蓝色实线所示

- 说明：

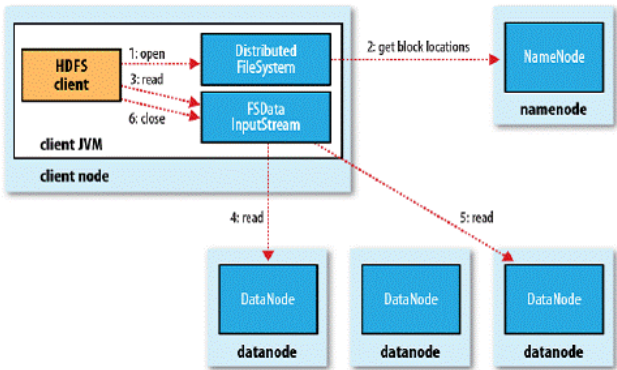
- 当客户端向 HDFS 文件写入数据的时候，一开始是写到本地临时文件中。假设该文件的副本系数设置为 3，当本地临时文件累积到一个数据块的大小时，客户端会从 Namenode 获取一个 Datanode 列表用于存放副本。然后客户端开始向第一个 Datanode 传输数据，第一个 Datanode 一小部分一小部分 (4 KB) 地接收数据，将每一部分写入本地仓库，并同时传输该部分到列表中 第二个 Datanode 节点。第二个 Datanode 也是这样，一小部分一小部分地接收数据，写入本地 仓库，并同时传给第三个 Datanode。最后，第三个 Datanode 接收数据并存储在本地。因此，Datanode 能流水线式地从前一个节点接收数据，并在同时转发给下一个节点，数据以流水线的方式从前一个 Datanode 复制到下一个
- 时序图如下：



- 小结：
 1. 写入的过程，按hdfs默认设置，1T文件，我们需要3T的存储，3T的网络流量
 2. 在执行读或写的过程中，NameNode和DataNode通过HeartBeat进行保存通信，确定DataNode活着。如果发现DataNode死掉了，就将死掉的DataNode上的数据，放到其他节点去。读取时，要读其他节点去
 3. 挂掉一个节点，没关系，还有其他节点可以备份；甚至，挂掉某一个机架，也没关系；其他机架上，也有备份

hdfs读文件：

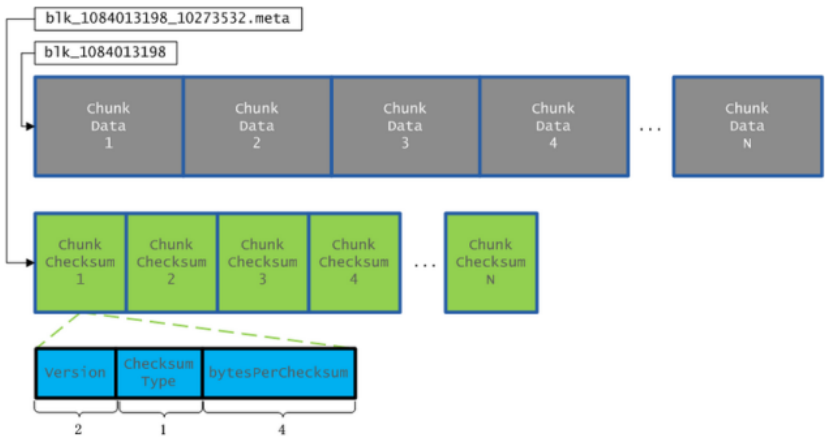
- 读到文件示意图如下：



- 客户端通过调用FileSystem对象的open()方法来打开希望读取的文件，对于HDFS来说，这个对象是分布文件系统的一个实例；
- DistributedFileSystem通过使用RPC来调用NameNode以确定文件起始块的位置，同一Block按照重复数会返回多个位置，这些位置按照Hadoop集群拓扑结构排序，距离客户端近的排在前面 (详见第三章)
- 前两步会返回一个FSDataInputStream对象，该对象会被封装成DFSInputStream对象，DFSInputStream可以方便的管理datanode和namenode数据流，客户端对这个输入流调用read()方法
- 存储着文件起始块的DataNode地址的DFSInputStream随即连接距离最近的DataNode，通过对数据流反复调用read()方法，将数据从DataNode传输到客户端
- 到达块的末端时，DFSInputStream会关闭与该DataNode的连接，然后寻找下一个块的最佳DataNode，这些操作对客户端来说是透明的，客户端的角度看来只是读一个持续不断的流
- 一旦客户端完成读取，就对FSDataInputStream调用close()方法关闭文件读取

block持续化结构：

- DataNode节点上一个Block持久化到磁盘上的物理存储结构，如下图所示：



- 每个Block文件（如上图中的blk_1084013198文件）都对应一个meta文件（如上图中的blk_1084013198_10273532.meta文件），Block文件是一个一个Chunk的二进制数据（每个Chunk的大小是512字节），而meta文件是与每一个Chunk对应的Checksum数据，是序列化形式存储

-----漫画版

根据Maneesh Varshney的漫画改编，以简洁易懂的漫画形式讲解HDFS存储机制与运行原理。

一、角色出演

如上图所示，HDFS存储相关角色与功能如下：

Client：客户端，系统使用者，调用HDFS API操作文件;与NN交互获取文件元数据;与DN交互进行数据读写。

Namenode：元数据节点，是系统唯一的管理者。负责元数据的管理;与client交互进行提供元数据查询;分配数据存储节点等。

Datanode：数据存储节点，负责数据块的存储与冗余备份;执行数据块的读写操作等。

二、写入数据

1、发送写数据请求

HDFS中的存储单元是block。文件通常被分成64或128M一块的数据块进行存储。与普通文件系统不同的是，在HDFS中，如果一个文件大小小于一个数据块的大小，它是不需要占用整个数据块的存储空间的。

2、文件切分

3、DN分配

4、数据写入

5、完成写入

6、角色定位

三、HDFS读文件

1、用户需求

HDFS采用的是“一次写入多次读取”的文件访问模型。一个文件经过创建、写入和关闭之后就不需要改变。这一假设简化了数据一致性问题，并且使高吞吐量的数据访问成为可能。

2、先联系元数据节点

3、下载数据

前文提到在写数据过程中，数据存储已经按照客户端与DataNode节点之间的距离进行了排序，距客户端越近的DataNode节点被放在最前面，客户端会优先从本地读取该数据块。

4、思考

四、HDFS容错机制——第一部分：故障类型及监测方法

1、三类故障

(1)第一类：节点失败

(2)第二类：网络故障

(3)第三类：数据损坏(脏数据)

2、故障监测机制

(1)节点失败监测机制

(2)通信故障监测机制

(3)数据错误监测机制

3、回顾：心跳信息与数据块报告

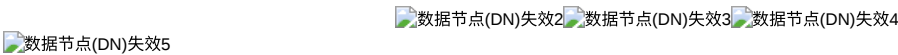
HDFS存储理念是以最少的钱买最烂的机器并实现最安全、难度高的分布式文件系统(高容错性低成本)，从上可以看出，HDFS认为机器故障是种常态，所以在设计时充分考虑到单个机器故障，单个磁盘故障，单个文件丢失等情况。

五、容错第二部分：读写容错

1、写容错

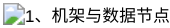
2、读容错

六、容错第三部分：数据节点(DN)失效

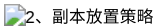


七、备份规则

1、机架与数据节点



2、副本放置策略



数据块的第一个副本优先放在写入数据块的客户端所在的节点上，但是如果这个客户端上的数据节点空间不足或者是当前负载过重，则应该从该数据节点所在的机架中选择一个合适的数据节点作为本地节点。

如果客户端上没有一个数据节点的话，则从整个集群中随机选择一个合适的数据节点作为此时这个数据块的本地节点。

HDFS的存放策略是将一个副本存放在本地机架节点上，另外两个副本放在不同机架的不同节点上。

这样集群可在完全失去某一机架的情况下还能存活。同时，这种策略减少了机架间的数据传输，提高了写操作的效率，因为数据块只存放在两个不同的机架上，减少了读取数据时需要的网络传输总带宽。这样在一定程度上兼顾了数据安全和网络传输的开销。

标签: [hadoop](#)

好文要顶

关注我

收藏该文



愤怒老蛞蝓

关注 - 1

粉丝 - 0

+加关注

0

推荐

0

反对

« 上一篇: [大数据现状分析 \(2\)](#)
» 下一篇: [wget window](#)

posted on 2017-11-22 16:37 愤怒老蛞蝓 阅读(478) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

- 【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库！
- 【福利】华为云4核8G云主机免费试用
- 【活动】申请成为华为云云享专家 尊享9大权益
- 【活动】腾讯云+社区开发者大会12月15日首都北京盛大起航！

腾讯云AMD云服务器

节省IT成本30%

1核1G AMD机型0.57元/天起

立即抢购


- 相关博文：
- [深度剖析hdfs原理](#)
 - [hadoop\(-\):深度剖析hdfs原理](#)
 - [2017.5.12 开源大数据查询分析引擎现状](#)
 - [MapReduce/GFS/BigTable三大技术资料](#)
 - [开源大数据查询分析引擎](#)

×

AI护老虎, 智护生态

英特尔® 用人工智能解决大问题





- 最新新闻：
- [你每个月10GB流量，6GB都被广告偷走了！](#)

- [高通骁龙855发布了，明年Android手机会有什么变化？](#)
- [.NET Core 2.2正式发布，有你喜欢的特性吗？](#)
- [科学家第一次测量宇宙全部星光，证实令人不安的结论](#)
- [猎豹应用的小烦恼，出海绕不过去的本地化](#)
- » [更多新闻...](#)

Powered by:
[博客园](#)
Copyright © 愤怒老蛞蝓