基于范德蒙码的HDFS优化存储策略研究

宋宝燕 王俊陆 王 妍
(辽宁大学信息科学与技术学院 沈阳 110036)

摘 要 随着大数据时代的到来,新型文件系统HDFS(Hadoop分布式文件系统)的应用越来越广泛.但其本身也存在着整体存储成本过高、可扩展性低、节点负载均衡能力不足等问题.因此,该文提出了一种基于范德蒙码的HDFS分散式动态副本存储优化策略,针对HDFS大多部署在大量的廉价硬件集群上的实际情况,在范德蒙码优化策略的基础上,采用分散式动态副本控制的思想对HDFS文件操作的计算过程、计算模式以及译码触发策略进行系统的改进,并通过校验码动态设置的方式将容错度控制在一个理想的范围之内,此外,结合伽罗华有限域理论对范德蒙码的编译码操作及计算方法进行全面优化,在不影响HDFS存储结构的前提下,降低了范德蒙码编译码的时间代价和计算的内存压力,节约了HDFS约30%的存储开销,数据可靠性提高了约200%,均衡HDFS系统节点负载能力,译码恢复效率平均提升约40%,形成了一套完整的、系统的优化方案,为未来HDFS的发展提供了一条有效途径.

关键词 大数据;HDFS;范德蒙码;分散式动态副本;优化存储
中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2015.01825

Optimized Storage Strategy Research of HDFS Based on Vandermonde Code

SONG Bao-Yan WANG Jun-Lu WANG Yan
(*School of Information Science Technology, Liaoning University, Shenyang* 110036)

Abstract With the arrival of the era of big data, the application of the new file management architecture HDFS (Hadoop Distributed File System) is more and more widely. But it also having many problems like the overall storage costs too much, the extensibility is low, the nodes load balance ability is insufficient and so on. So this paper proposes an Optimized Storage Strategy of HDFS Based on Vandermonde Code, according to the actual situation, which the HDFS are deployed in a large number of inexpensive hardware clusters, it uses the thought of decentralized dynamic replication control to optimize the calculation process, calculation mode and decoding trigger strategy of HDFS file operations comprehensively based on Vandermonde Code optimization strategy, and uses the dynamic setting check code to control the fault tolerant in a desirable range. Besides, it uses Galois finite field theory to optimize the encoding and decoding operation of Vandermonde Code and calculation method comprehensively. Under the premise of without affecting storage structure of HDFS, it reduces time cost and the calculation memory pressure of Vandermonde Code, reduces about 30% of the storage cost, increases about 200% of the reliability of HDFS, balances the load of system, increases about 40% of the decoding recovery efficiency, formed a set of complete and systematic optimization solution, providesan effective way fordevelopment of HDFS in the future.

Keywords big data; HDFS; Vandermonde code; decentralized dynamic replication; optimized storage

---

收稿日期:2014-05-16;最终修改稿收到日期:2015-04-01.本课题得到国家自然科学基金(61472169,60873068)、辽宁省教育厅优秀人才支持计划项目基金(LR201017)资助.**宋宝燕**,女,1965年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为数据库理论与技术、RFID事件流处理技术、海量数据处理技术、图数据处理技术等.E-mail:bysong@lnu.edu.cn.**王俊陆**,男,1988年生,硕士研究生,主要研究方向为数据库理论与技术、大数据处理技术、海量数据处理技术等.**王 妍**,女,1978年生,博士研究生,副教授,主要研究方向为数据库理论与技术、感知数据处理技术、海量数据处理技术等.

1 引 言

随着 IT 技术迅猛发展以及信息化的深入，数据的生成量大约每两年就会上翻一倍，几乎达到了电子领域中摩尔定律的标准. 据统计，平均每秒有 200 万用户在使用谷歌搜索，Facebook 用户每天共享的内容超过 40 亿，Twitter 每天处理的推特数量超过 3.4 亿[1]. 大数据时代下，数据量的增长速度会超过存储数据介质容量的增长速度[2]，即存储代价会不断上升，存储介质的成本在不断增加. 面对这种巨大的数据生成量，传统数据管理系统中的数据处理技术受到了极大挑战[3]，如何更高效、稳定的存储这些数据，成为数据处理等许多领域研究的热点[4].

目前人们往往采用 HDFS(Hadoop 分布式文件系统)来存储这些大数据. HDFS 是一种可靠的、容错的文件系统，适合低廉地存放大规模的数据. 但随着 HDFS 被广泛使用，其固有的如整体存储成本过高、可扩展性低、节点负载均衡能力不足等问题也越来越突显. 针对这些问题，许多专家学者进行了深入广泛的研究，取得了卓有成效的进展，如针对云端存储优化的 HDFS 存储传输策略[5]、以高效存储为目的基于 XOR 码的 HDFS 优化策略[6-7]等.

在此基础上，本文亦对大数据的有效存储问题进行了研究，提出了一种基于范德蒙码的 HDFS 优化存储方法：(1)针对 HDFS 存储成本过高的问题，采用了基于范德蒙码的编译码操作对 HDFS 进行数据优化，摒弃 HDFS 原本的多副本镜像复制策略；(2)加入分散式动态副本控制思想，改进单纯范德蒙码策略无法应对灾难性机架和节点失效的问题；(3)采用分组分列策略以及有限域理论对译码操作的计算过程、计算模式和计算方法进行改进，精简矩阵运算的步骤；(4)根据范德蒙码输出矩阵的特点，改进译码触发操作的时机，降低译码操作触发的频率.

2 相关工作

2.1 HDFS

HDFS 有别于已有的分布式文件系统. 其核心设计思想为“一次写入，多次读取[8]”的高效访问方式，且系统不需要架设在价格高昂、性能稳定的硬件设备上[9]. HDFS 采用 Master/Slave 架构[10]设计，目的是对大量数据的吞吐应用进行优化[11]. 同时，与磁盘存储默认的数据块概念类似，HDFS 上的文件被划分为多个 Block 分块[12]，且每个 Block 分块都作为独立的存储单元，以多副本备份策略进行存储工作.

2.1.1 HDFS 架构

HDFS 由两类节点构成：管理者 Namenode 和工作者 Datanode[13]. 这两类节点相互分工又共同协作，分别以不同的模式运行，完成各自的任务. 图 1 为 HDFS 的组织结构图.

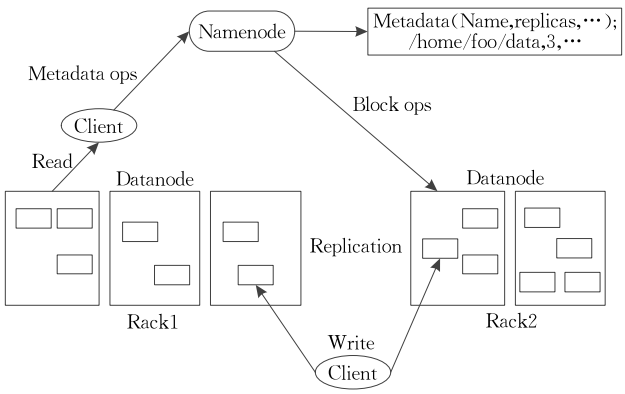

图 1 HDFS 组织结构图

如图 1 所示，HDFS 中管理者 Namenode 暂存所有存储 Block 文件块的 Datanode 节点的信息，并负责数据块的备份操作，周期性地从集群中的各个节点接收反馈心跳信号以及 Block 分块的反馈状态报告 Blockreport，Namenode 会根据这些反馈信息来检测 HDFS 系统中 Block 分块数据映射的状况信息. 工作者 Datanode 是 HDFS 分布式文件系统的实际操作节点. 这些节点由管理者 Namenode 统一调度，根据实际需要进行文件的存储操作或者数据块的检索操作. 同时，Datanode 节点还负责执行管理者 Namenode 发出的对数据块的增删查改等操作的指令，并周期性地向管理者 Namenode 提交 Block 数据块的反馈心跳信号以及 Block 分块的反馈状态报告.

2.1.2 HDFS 的主要优势

HDFS 是通过多副本镜像复制策略来实现备份操作的，默认情况下副本系数为 3，即 HDFS 的副本策略是将 1/3 的备份数据存储在本地机架的某个节点中，另 1/3 的备份数据存储在同一个机架的另一个 Datanode 节点中，最后 1/3 的备份数据是存储在不同机架上的某个 Datanode 节点中的. 该策略确保了 HDFS 的数据移动与安全机制.

首先，HDFS 是一种容错的分布式文件系统. HDFS 所有数据经由源数据生成后会立即进行副本的备份操作，并将副本分别存储到集群的各个节点中执行数据的管理调度和数据的分析请求[14]，允许节点数据出现灾难性崩溃的情况.

其次，HDFS 可以部署在大量廉价设备集群上以节约存储成本. HDFS 采用的多副本镜像复制策略保证了数据存储的安全性和可靠性，通过增加数据的副本数量弥补硬件设备的不足，同时也减少了本地机架与异地机架之间数据传输的时间开销.

最后，在实际的数据读写操作过程中，HDFS 的多副本复制策略实现了多节点传输，降低了对网络传输总带宽的依赖，在不损害数据可靠性和读取操作效率的前提下，提高了对文件高效写入操作的支持.

2.1.3　HDFS 存在的不足

在 HDFS 这种多副本镜像复制策略中，各个机架上所存储的备份数据的实际分布状态并不是均匀的. 因此，这种多副本镜像复制的策略也使得 HDFS 产生了诸多问题.

首先，HDFS 的整体存储成本过高[15]. 在默认情况下，HDFS 实际存储数据所需的空间开销为原始存储数据容量的 3 倍. 以此种方式对 PB 至 ZB 数据级的数据进行操作时，增加的数据冗余量会使得实际物理存储介质无法适应，且过大的数据规模也会直接增加存储设备的硬件成本预算以及建立文件索引的时间代价.

其次，HDFS 的可扩展性低[16]. 在 HDFS 的多副本镜像复制策略中，所有系统中的数据都要保持相同的副本数目，且系统无法根据外界实际需求的改变动态调度已有的副本，这就导致了系统的灵活性及可扩展性降低.

最后，HDFS 的节点负载均衡能力不足[17]. 由于所有的 Block 分块数据的位置信息都必须要存储在管理者 Namenode 的运行内存中，Block 文件块数量的不断增加会导致管理者 Namenode 的运行内存压力不断变大，超出系统有限的负载均衡能力.

2.2　HDFS 的优化

随着对数据处理技术要求越来越高，HDFS 的这些缺点会产生越来越多的问题. 各领域的专家学者通过对 HDFS 文件系统存储框架的不断研究与拓展，各类存储备份策略的优化方案已经逐步展现出雏形. 典型优化方案如基于 GE 码的 HDFS 存储优化策略、基于 FEC 码的 HDFS 云存储优化策略、以及企业级应用基于 XOR 码的 HDFS 优化存储策略等.

文献[15,18]提出了一种基于 GE 码的 HDFS 存储优化策略. GE 码是一种垂直码的编码方式，该策略能够通过引入较少的冗余校验块而对数据的传输和数据的存储提供可靠保证，其容错度可以控制在[0.30,0.35]的范围内，在确保数据的可靠性和存储效率的同时，很大程度上也降低了系统的存储开销，并提供了更为灵活的数据负载均衡技术.

但是，基于 GE 码的 HDFS 优化策略也存在着不足之处，如编码分片的方式需要重构 HDFS 分块，译码操作时需要进行大量的动态计算，复杂度较高，这导致了译码的效率较低、后期维护升级难度较大等问题.

文献[5]中讨论了一种基于 FEC 码的 HDFS 优化策略，该策略是针对云端存储文件系统占用空间大，负载均衡能力差等问题进行改进的. 其设计思想是把需要上传的文件先进行编码分块的操作，然后整合编码后的输出数据，最后再把数据传输至云端的 HDFS 的分布式系统集群中. 整个过程不进行数据副本的备份操作或者只进行少量的数据副本备份，降低了整个文件系统操作过程的空间开销和传输代价. 此外，基于 FEC 码的 HDFS 优化策略中加入了数据传输的完成性检测工作，因此，该策略更适用于远程文件的传输.

基于 FEC 码的优化策略优势是存储代价较小，与基于 GE 码优化策略相比，更有利于云端数据上传和下载等传输操作；但是，该策略也存在着计算较为复杂，灵活性较低，本地数据存储效率较低的缺点.

文献[6-7]中提到的一种基于 XOR 码的优化策略，其编码和译码的方式相对简单，采用单一的异或运算生成奇偶校验码 Parity，原始数据是按照 Stripe 进行分条目存储的，且每一个 Stripe 条目只会生成一个 Parity，当某个 Stripe 条目中丢失或者损坏了一个 Block 数据块时，XOR 码优化策略能够通过异或校验的译码运算过程进行数据的恢复操作.

相比上述优化方案，基于 XOR 码的 HDFS 优化存储策略执行效率更高，但由于译码操作只能恢复单一 Block 数据块的数据，即 XOR 码的优化策略中只允许出现一个数据块失效的情况，所以，基于 XOR 码的 HDFS 优化策略对文件系统的纠错能力比较弱.

3 基于范德蒙码的优化策略

由上文分析可知,随着数据生成量的逐渐增大,HDFS 的应用将会越来越广泛,还有很多问题有待进一步探讨.本文对 HDFS 文件系统及现有的几种优化策略进行了深入研究,提出了一种空间开销更优,稳定性和可靠性更好的优化策略:基于范德蒙码的 HDFS 优化存储策略(VanHDFS).

3.1 范德蒙码

范德蒙码也被称为范德蒙阵列纠删码[19],目前,关于纠删码的研究大致上可以分为两个方向:一种是低密度级联纠删码,如 LT 码(Luby Transform Codes)、LDPC 码等;另一种是最大距离可分纠删码,如 RS 码(Reed-Solomon Codes)等.其中,低密度级联纠删码的研究虽然取得了很多进展,但还存在很多的问题,并不适合大规模应用到生产和生活中去;而最大距离可分纠删码的编译码方案具有优秀的存储空间代价和良好的容灾能力,能够平衡整体效率和编译码性能的问题,通过对原始数据块添加校验码的方法来减少数据存储过程中的副本备份操作,在保证数据可靠性的前提下很大程度上降低存储开销,提高存储效率.

范德蒙码隶属于里德-所罗门码(Reed-Solomon Codes)的类别,是一种最大距离可分纠删码.在继承了最大距离可分纠删码优点的同时,范德蒙码的编译码方案还具有更低的迭代算法复杂度、更优秀的空间代价及容错能力.所以,范德蒙码也被公认为是一种可以充分利用信道带宽的优秀编译码解决方案之一.范德蒙码与其他编码策略既有相似之处,又有所不同.从数学的角度进行分析,目前被广泛使用的 RAID5、RAID6 等编码都可以看作是范德蒙码阵列算法的一个子集.当冗余校验码有且只有一个的时候,范德蒙码就退化成了 RAID5 算法,在伽罗华域上将源数据进行 XOR 逻辑运算以得到冗余校验码.同理,当同时产生两个冗余校验码数据的时候,范德蒙码就退化成了 RAID6 算法,在伽罗华域上通过 XOR 码等方法进行运算.因此,范德蒙码也可以看成是已有 RAID5、RAID6 编码基础上的一种延伸.

现阶段,为提高数据传输的实时性和可靠性,大量利用范德蒙码同卷积码、数字喷泉码等编码级联使用的方案已经被广泛地应用到了数字视频信号、远程文件存储、数据传输等实际项目中.如在网络通信领域中,信息传输的过程常常会由于网络拥塞或受到其他原因影响而造成数据传输成功率低的问题.虽然很多利用检错码、纠错码的方式对失效数据进行重传的策略可以改善网络的可靠性,但反复传输相同数据也造成了额外的网络开销和巨大的传输时间代价.因此,为了平衡传输开销和数据可靠性之间的矛盾,范德蒙码及类范德蒙码的纠删码策略被广泛应用到网络传输、数据通信等领域,对传输数据进行纠删码编码提高数据可靠性的同时也不会影响传输效率.此外,如磁盘存储(RAID)、文件系统中也会采用类似的容错技术来增强数据的可靠性.

3.2 VanHDFS 与 HDFS 存储策略

范德蒙码以其优秀的编译码效率和存储空间代价,被广泛应用于电子通信、互联网数据安全、磁盘介质存储等领域的纠错技术中.考虑到 HDFS 存储策略的优势、不足以及范德蒙码的编译码操作特性,本文将范德蒙码编译码引入到 HDFS 存储策略中,提出采用"数据块+校验码"存储方式的 VanHDFS 替代 HDFS 的多副本备份存储策略.

HDFS 存储框架大多是建立在大量廉价硬件存储设备集群基础之上的,硬件环境的执行效率和整体的稳定性都较低,节点数据丢失或损坏的情况是不可避免的.VanHDFS 改变了 HDFS 需要存储大量的冗余数据来确保数据可靠性的传统方式,在适当的容错度设置下,只需额外存储较少的校验码块.当 HDFS 中一个或者多个数据块中的数据失效时,通过 VanHDFS 策略的译码操作即可将数据直接进行恢复.因此,VanHDFS 策略可以在确保 HDFS 存储数据可靠性的同时节约大量的存储成本.

在 VanHDFS 策略中,设存入 HDFS 的原始数据包的数量为 k,对这 k 包数据进行范德蒙码的编码操作,此时,根据容错度设置的不同,生成的 q 包数据中需要有相应的额外数量的冗余校验码块 l(其中,$q=k+l$).在 HDFS 的读写操作中,当任意包的数据发生丢失或者失效的时候(失效数据包的数目要小于 $q-k$),范德蒙码的译码策略就能够选取生成的 q 包数据中其他有效的 k 包数据来恢复构建原始的 q 包数据.图 2 为 VanHDFS 策略优化过程示意图.

如图所示,数据包 a 到数据包 e 中的数据为原始数据,而数据包 a 到数据包 G 中的数据为编码后的输出数据,其中,数据包 F 和数据包 G 为校验码

包，生成编码后数据 q 的过程可表示为 $q=\boldsymbol{P}(a,b,c,d,e)$，其中，$\boldsymbol{P}$ 为编码操作的构造矩阵. 范德蒙码的译码过程为 $q=\boldsymbol{P}^{-1}(a',b',c',d',e')$，其中，$\boldsymbol{P}^{-1}$ 为 $\boldsymbol{P}$ 的求逆运算.

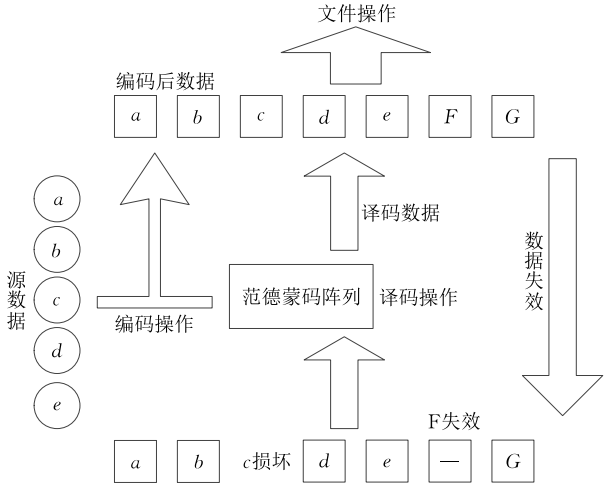

图 2　VanHDFS 策略优化过程示意图

VanHDFS 策略可以在一定范围内最大程度降低 HDFS 的存储空间代价，节约存储成本，这是 HDFS 的多副本镜像复制策略无法比拟的. 此外，与其他存储优化策略不同，范德蒙码的编码和译码操作均是基于范德蒙矩阵进行的，故采用范德蒙码对 HDFS 文件系统进行优化可以避免对 HDFS 的整体结构和内部的 Block 分块进行重构，与 HDFS 的存储模式保持最大程度的相似性.

3.3　VanHDFS 的编译码策略

VanHDFS 策略中使用的范德蒙码本质上是基于数学上范德蒙矩阵转换而来的，故范德蒙码本身并不会完全适用于 HDFS 的存储，需要进一步有针对性的改进. 在 VanHDFS 编码策略中，范德蒙码需要生成系统码矩阵以优化 HDFS 的读取操作；而在译码策略中，根据 HDFS 数据的存储特点，VanHDFS 采取整行译码的方式，确保 HDFS 的可靠性.

3.3.1　编码策略

单从数学角度分析，设本源根 α 的幂级数为 w，本源多项式为 $P(\boldsymbol{x})=1+\boldsymbol{x}+\boldsymbol{x}^2+\boldsymbol{x}^3+\cdots+\boldsymbol{x}^w$，那么，这个给定的伽罗华域 $GF(2^w)$ 就可以由多项式 $P(\boldsymbol{x})$ 的解集 α 来生成：$GF(2^w)=\{0,1,\alpha,\alpha^2,\cdots,\alpha^{254},\cdots\}$；构造 $n\times k$ 阶的范德蒙矩阵 $\boldsymbol{B}$（其中，$n>k$），设 $\boldsymbol{B}_{n\times k}$ 为编码生成矩阵，则由矩阵 $\boldsymbol{B}$ 中任意 k 行组成的子矩阵 $\boldsymbol{B}^*$ 都满足 $|\boldsymbol{B}^*|\neq 0$ 的特性，故子矩阵 $\boldsymbol{B}^*$ 可逆，矩阵 $\boldsymbol{B}$ 同时满足纠删码的生成矩阵的特性. 设输入数据为 $D(d_1$ 到 $d_k)$，则范德蒙码编码后的输出数据为 c_1 到 c_n，那么，输入数据和编码后输出数据之间的关系如式(1)所示.

$$\boldsymbol{B}\times\boldsymbol{D}=\begin{bmatrix}1 & 1 & \cdots & 1\\ x_1 & x_2 & \cdots & x_k\\ x_1^2 & x_2^2 & \cdots & x_k^2\\ \vdots & \vdots & & \vdots\\ x_1^{n-1} & x_2^{n-1} & \cdots & x_k^{n-1}\end{bmatrix}\begin{bmatrix}d_1\\ d_2\\ \vdots\\ d_k\end{bmatrix}=\begin{bmatrix}c_1\\ c_2\\ \vdots\\ c_n\end{bmatrix}\tag{1}$$

构造辅助矩阵 $\boldsymbol{y}$，令矩阵 $\boldsymbol{y}=\boldsymbol{Bx}$，且其中 $\boldsymbol{x}=(x_0,x_1,\cdots,x_{k-1})^{\mathrm{T}}$. 可以发现生成矩阵 $\boldsymbol{B}$ 属于原始范德蒙矩阵，符合纠删码生成矩阵的固有特点，可以直接用来进行数据的编码操作.

但是，考虑到对 HDFS 文件系统的应用，如果使用该范德蒙生成矩阵对 HDFS 中的原始数据进行编码，那么，编码后的数据矩阵和原始数据矩阵将会完全不同，即数据部分不是系统码，得到的编码矩阵中将无原数据块. 这对于通信领域中数据传输的加密有很大益处，但在实际的 HDFS 文件操作中，这样的编码矩阵会导致用户每次从 HDFS 节点读取数据都要进行范德蒙码译码操作，大大增加对于数据读写操作的计算代价和文件系统的时间开销，故并不适用于 HDFS 的大规模数据集操作. 因此，必须对矩阵 $\boldsymbol{B}$ 进行转化，使编码后的矩阵生成系统码矩阵，保证 HDFS 的存储数据中存在原始数据.

使用高斯消元法对矩阵 $\boldsymbol{B}$ 的前 k 行和前 k 列进行消元运算. 转换后的矩阵 $\boldsymbol{B}$ 上半部分为 k 阶的单位矩阵 $\boldsymbol{I}_k$，余下部分即为消元运算后的冗余校验码生成行. 在矩阵变换的过程中，所有的运算均采用标准的模 2 消元方式，范德蒙码的生成矩阵 $\boldsymbol{B}$ 使用 k 阶单位矩阵与 $(n-k)\times k$ 阶的范德蒙变形矩阵联合构成，如式(2)所示.

$$\boldsymbol{G}=\boldsymbol{B}=\begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ 0 & 1 & 0 & \cdots & 0\\ \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ y_1 & y_2 & y_3 & \cdots & y_k\\ \vdots & \vdots & \vdots & & \vdots\\ y_1^{n-k} & y_2^{n-k} & y_3^{n-k} & \cdots & y_k^{n-k}\end{bmatrix}\tag{2}$$

设矩阵 $\boldsymbol{D}$ 为原始数据包矩阵，使用编码矩阵 $\boldsymbol{G}$ 和原始数据 $\boldsymbol{D}$ 进行矩阵乘法运算，输出矩阵 $\boldsymbol{E}$ 上半部分 $k\times m$ 包数据和原始数据 $\boldsymbol{D}$ 完全相同，即 $\boldsymbol{E}_{k\times m}$ 为系统码矩阵，且整个矩阵的转换过程不会破坏纠删码生成矩阵的特性，如式(3)所示.

$$\boldsymbol{E}=\boldsymbol{G}\times\boldsymbol{D}=\begin{bmatrix}1&0&0&\cdots&0\\0&1&0&\cdots&0\\\vdots&\vdots&\vdots&&\vdots\\0&0&0&\cdots&1\\y_1&y_2&y_3&\cdots&y_k\\\vdots&\vdots&\vdots&&\vdots\\y_1^{n-k}&y_2^{n-k}&y_3^{n-k}&\cdots&y_k^{n-k}\end{bmatrix}\begin{bmatrix}d_1\\d_2\\\vdots\\d_k\end{bmatrix}=\begin{bmatrix}d_1\\d_2\\\vdots\\d_k\\L_1\\\vdots\\L_{n-k}\end{bmatrix}\tag{3}$$

至此,采用范德蒙码生成矩阵 $\boldsymbol{E}$ 对 HDFS 的原始数据进行编码,实际操作中读取的数据都为原始数据,省去了每次读取都需要进行范德蒙码译码操作的步骤,符合用户对 HDFS 的实际应用需求.

3.3.2 译码策略

由上文可知,VanHDFS 策略编码后的输出矩阵属于系统码矩阵,当系统中的源数据 d_1 到 d_k 或者校验码数据 L_1 到 L_{n-k} 中发生数据失效问题时,就需要对 HDFS 进行范德蒙码的译码操作.从数学的角度进行分析,即只要将读取的有效数据和生成矩阵的逆矩阵相乘就可以恢复丢失的数据.

根据 HDFS 以 Block 文件块方式存储数据的特点,在 VanHDFS 策略中,如果 HDFS 数据块单行的错误比特数大于 1 或者等于 1,那么就对整行都添加错误标识,这样就可以确保当失效块所占的行数小于或等于 HDFS 系统编码容错度设置的范围时,VanHDFS 策略经过一次译码操作即可对 HDFS 系统中全部的失效数据块进行恢复,且在此过程中无需考虑编码输出矩阵中的各个行列的数据块的具体失效情况.

设编码后 HDFS 的输出矩阵中第 3 行与第 k 行数据块同时发生错误,且 $k\leq n$,则从冗余校验码矩阵部分中选取第 1 行与第 2 行参与恢复运算(设第 1 行与第 2 行冗余校验码正常),同原始数据矩阵一并构造辅助矩阵 $\boldsymbol{D}'$.辅助矩阵形可用如式(4)所示的矩阵表示.

$$\boldsymbol{D}'=\begin{bmatrix}d_{11}&d_{12}&\cdots&d_{1m}\\d_{21}&d_{22}&\cdots&d_{2m}\\c_{11}&c_{12}&\cdots&c_{1m}\\\vdots&\vdots&&\vdots\\d_{k-1,1}&d_{k-1,2}&\cdots&d_{k-1,m}\\c_{21}&c_{22}&\cdots&c_{2m}\end{bmatrix}\tag{4}$$

构造译码还原矩阵 $\boldsymbol{A}''$,通过对单位矩阵和冗余校验码矩阵组合后进行消元运算可得,矩阵 $\boldsymbol{A}''$可用如式(5)所示的矩阵表示.

$$\boldsymbol{A}''=\begin{bmatrix}1&0&\cdots&0&0\\0&1&\cdots&0&0\\a_{11}&a_{12}&\cdots&a_{1,k-1}&a_{1,k}\\\cdots&\cdots&&\cdots&\cdots\\0&0&\cdots&1&0\\a_{21}&a_{22}&\cdots&a_{2,k-1}&a_{2,k}\end{bmatrix}\tag{5}$$

对译码还原矩阵 $\boldsymbol{A}''$求逆,设其逆矩阵为 $\boldsymbol{A}'^{-1}$.将矩阵 $\boldsymbol{A}'^{-1}$与矩阵 $\boldsymbol{D}'$作为译码恢复操作输入参数,则恢复数据的表达式可用如式(6)所示的矩阵运算表示.

$$\boldsymbol{D}=\boldsymbol{A}'^{-1}\boldsymbol{D}'=\begin{bmatrix}d_{11}&d_{12}&\cdots&d_{1m}\\d_{21}&d_{22}&\cdots&d_{2m}\\\vdots&\vdots&&\vdots\\d_{k1}&d_{k2}&\cdots&d_{km}\end{bmatrix}\tag{6}$$

数据矩阵 $\boldsymbol{D}$ 为译码恢复后的输出矩阵,即 HDFS 的原始数据矩阵,失效数据恢复完成.

4 VanHDFS 动态副本分散存储策略

由 VanHDFS 编译码过程可知,对于 HDFS 系统中的数据块失效情况,VanHDFS 策略通过一次译码操作即可恢复全部的失效数据,且无需考虑 HDFS 存储中实际数据块的具体失效情况.由于需要进行矩阵运算,因此,VanHDFS 更适合在系统环境相对较稳定、处理能力较好的硬件集群上部署,实现最优的存储开销.随着硬件设备的发展,硬件环境部署的成本必然会不断降低,设备的稳定性和处理能力也会随之提高,故 VanHDFS 策略在未来 HDFS 的发展过程中会有一个很好的前景.

但在目前的实际情况中,对于执行效率不高、稳定性差的廉价硬件设备集群环境来说,VanHDFS 策略在时间开销和计算内存方面的压力都是巨大的.同时,单纯的 VanHDFS 策略没有办法应对节点或整个机架完全崩溃的状况,会导致数据完全丢失的状况发生,不能很好的适用于实际的使用环境,因此,本文对 VanHDFS 策略进行进一步改进,提出 VanHDFS 动态副本分散存储策略(D-VanHDFS).图 3 所示的为 D-VanHDFS 策略架构图.

如图所示,针对上文提出的实际因素考虑,在 VanHDFS 的基础上,D-VanHDFS 策略进行了系统的、完整的改进设计,使优化策略更适用于实际的应用操作.

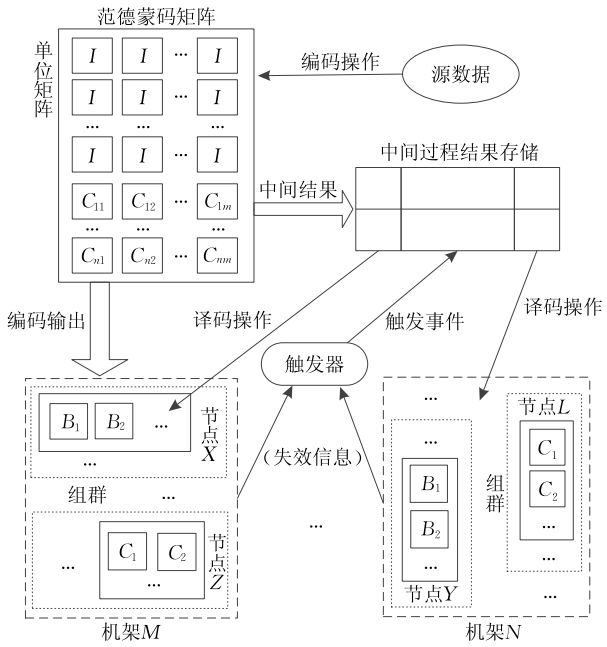

图 3　D-VanHDFS 策略架构图

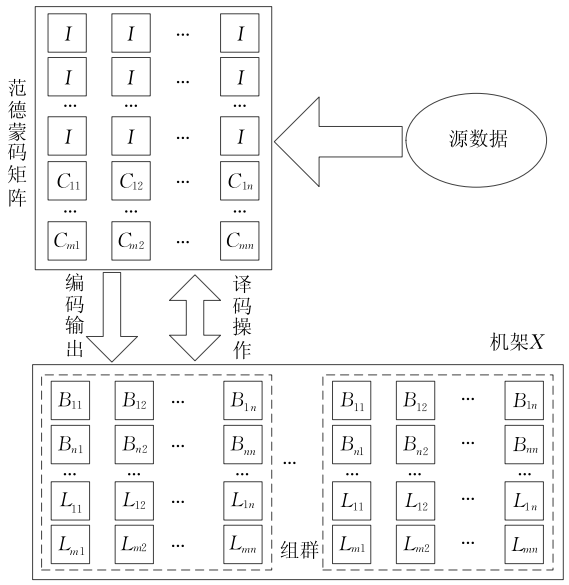

图 4　D-VanHDFS 分组策略示意图

4.1　分散式动态副本策略

针对单纯的 VanHDFS 没有办法应对节点或整个机架完全崩溃的状况，D-VanHDFS 策略中引入了动态副本分散控制的思想.

由于范德蒙码策略本身是基于矩阵进行的，编码和译码操作都需要多个辅助矩阵和构造矩阵共同参与进行恢复计算. 因此，如果对全部的数据块进行一次性编译码操作，虽然可以保证当存在多个失效数据块时，通过一次矩阵译码计算就能将所有失效的数据块全部恢复，但编码和译码操作的计算代价以及对硬件设备的性能、可靠性要求都会大幅提高. 鉴于目前 HDFS 都是架设在大量的廉价硬件设备环境中的，如果需要在性能和稳定性都较低的硬件平台上采用 VanHDFS 策略，大量的矩阵计算会增加译码操作的时间开销和系统的整体效率. 因此，基于并行性和编译码时间代价的考虑，需要将原数据块进行 Group 分组操作，每个数据块中要添加一个分组标志位，用来标记该数据块是属于哪个 Group 分组. 图 4 为 D-VanHDFS 分组策略的示意图.

如图所示，由于分组大小的设置均为统一的矩阵形式，故在系统容错度设置一定的情况下，每个 Group 分组都具有完全相同的性质，系统的译码矩阵就只有一个，可以通用于文件系统的所有数据分组，降低了系统设计的复杂度，提高了系统的灵活性和可操作性.

此外，在 D-VanHDFS 策略存储过程中，将每个 Group 分组的编码后矩阵按照纵列 Column 进行标记，即所有数据块被分为若干个 Group 分组，每个 Group分组又由若干个 Column 数据列构成. Column 数据列包含两部分内容：数据块部分和校验码部分. 当编码输出数据中的数据块或者冗余校验码块失效时，D-VanHDFS 的译码操作将不再进行矩阵与矩阵之间的运算，转换为矩阵行列之间的计算.

HDFS 的多副本镜像复制策略本质上是通过增加副本复制的数量来提高可靠性的. 通常状况下，系统默认的三副本策略就可以满足数据存储及读写操作的需求. 故对于 D-VanHDFS 策略，可将动态副本数目设置为$\lceil N/2 \rceil$（其中 N 为原 HDFS 的镜像副本数量），在要保证优秀的存储空间开销外，兼顾整个 HDFS 的可靠性. 与此同时，加入了范德蒙码编码后的 Column 数据列的副本对于机架和节点的崩溃状况有了很好的控制，但对于单纯的节点失效状况不能很好的应对，当存储某 Column 列数据的节点完全失效，该动态副本策略的可靠性上就会低于 HDFS 的多副本镜像复制策略. 因此，还需要采用分散存储策略. 在同一机架上，当容错度一定时，需要将每个 Column 数据列中包含的数据块部分以容错度为单位进行分散式存储，各个节点中只保留对应容错度的数据块信息. 图 5 所示的为分散式动态副本策略示意图.

如图所示，此时若出现节点失效的情况，在逻辑上只是丢失了 D-VanHDFS 译码容错范围内的部分数据块，可通过译码操作快速恢复，不会对文件产生影响.

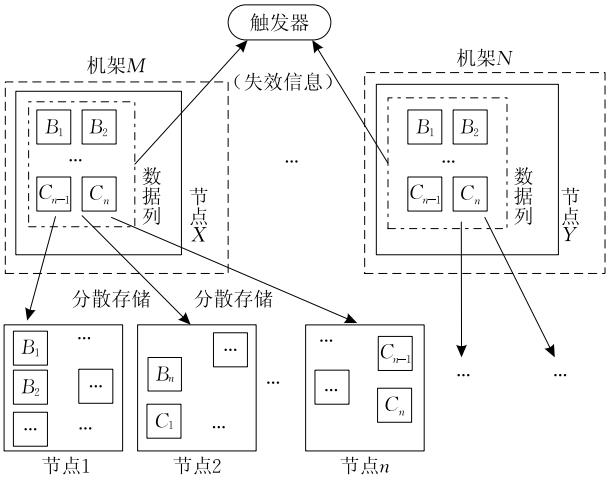

图 5 分散式动态副本策略示意图

4.2 计算过程的改进

通过对 VanHDFS 译码操作过程的研究发现，还原矩阵和构造矩阵在计算的过程中，只有部分矩阵参与实际的运算过程，并且都是基于行或者列来计算的，因此在进行译码操作之前，就必须对还原矩阵和构造矩阵进行变形，这又增加了计算的时间代价. 基于这一点考虑，D-VanHDFS 策略利用 Column 分列操作进行计算，将数据部分和校验码部分当成统一的整体，这样便省去了建立构造矩阵的过程，对于提高运算效率有重要作用.

建立辅助矩阵 $\boldsymbol{G}$，则矩阵 $\boldsymbol{G}$ 和原始数据矩阵 $\boldsymbol{D}$ 之积为去掉失效数据块的 Column 数据列中有效数据构成的矩阵 $\boldsymbol{S}$，即 $\boldsymbol{G}\times\boldsymbol{D}=\boldsymbol{S}'$. 将矩阵 $\boldsymbol{G}$ 进行求逆运算，求出矩阵 $\boldsymbol{G}$ 的逆矩阵为 $\boldsymbol{G}^{-1}$，如图 6 所示，在等式两边同时左乘以矩阵 $\boldsymbol{G}^{-1}$，矩阵运算后的输出结果 $\boldsymbol{D}$ 即为原始数据 Column 数据列.

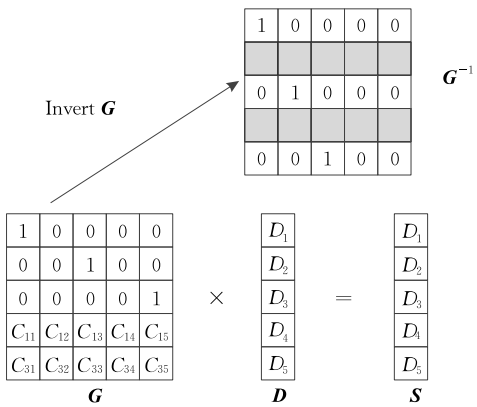

图 6 D-VanHDFS 策略译码操作过程

矩阵 $\boldsymbol{G}^{-1}$ 和矩阵 $\boldsymbol{G}$ 之积为 $k\times k$ 阶的单位矩阵 $\boldsymbol{E}$，故等式左侧即为原始数据. 这样，通过余下的数据矩阵 $\boldsymbol{S}$ 与对应矩阵 $\boldsymbol{G}$ 的系数组成的矩阵的逆做矩阵乘法操作就可以得到源数据列，也就得到了原始数据. 译码过程描述如下所示：

1. 获取失效块所在 Column 数据列的信息 *Failure-Column*[m]，包括分组号 *Groupno*，队列号 *Columnno* 以及失效数据块号 *Blockno*；其中，失效 Block 分块的数量为 m，且 $m\leq parity\_length$；构造辅助矩阵 *AuxiliaryArray*[k][n]，其中辅助矩阵的行数为 $k=n+parity\_length-m$，且 k 对应的行为失效 Block 分块所在 Column 数据列的有效数据行；

2. 构造译码操作的辅助数据列 *Validcolumn*[k]，其中，$k=n+parity\_length-m$. 选取的数据块为失效 Block 分块所在的 Column 数据列的有效数据块. *Validcolumn*[k]为译码操作的输入参数；

3. 使用 ErasuredCode 编码生成器构造范德蒙码的译码矩阵. 对矩阵 *AuxiliaryArray*[k][n]进行逆运算操作 *InverseOperation*(*AuxiliaryArray*)，生成范德蒙码的译码矩阵 *DecodeArray*[n][k]；

4. 使用 ErasuredCode 编码生成器对输入的数组进行编码操作，输入参数为 *Encoder.encode*(*AuxiliaryArray*, *ValidColumn*)，输出结果为 Column 数据列的源数据 *OutputColumn*[$n+parity\_length$].

4.3 计算模式的改进

在 D-VanHDFS 译码过程中，由于将分组矩阵进一步划分为多个 Column 数据列，使数据的计算范围进一步减小，因此，每个 Column 数据列中的数据块失效的情况也变为了可控状态，且该失效情况只和优化策略的容错度设置相关；而范德蒙码编码时设置的容错度通常都是一定的，故还原矩阵的总变形矩阵数也是一定的，总数为 C_n^m，其中 n 为每个 Column 数据列中数据部分和冗余校验码部分的总和，m 为编码容错度. 所以，可以将所有的变形还原矩阵都提前进行计算，并存统一存储到计算中间过程索引表中，当进行范德蒙码的译码操作时，可以直接查询索引表，将矩阵作为中间结果参与运算.

在此过程中，由于各个相关的参数都是在编码过程的时候就提前计算好的，且中间参数均为通用形式，在需要进行译码操作的情况下，将原本的动态还原计算模式转化成为了查表计算的静态计算模式，一次矩阵运算即刻恢复原始数据. 因此，在商用的大量廉价硬件集群环境下，D-VanHDFS 策略可以极大程度的弥补廉价硬件设备执行效率不高、计算能力不足的缺陷，降低运算内存的压力，从而提高整个分布式文件系统的存储效率. 译码过程如下所示：

1. 获取失效块所在 Column 数据列的信息 *Failure-Column*[m]，包括分组号 *Groupno*，队列号 *Columnno* 以及失效数据块号 *Blockno*；其中，失效 Block 分块的数量为 m，且 $m\leq parity\_length$；获取计算过程中间索引表的数据 *AuxiliaryArray*[$n+parity\_length-m$][n]；

2. 构造译码操作的辅助数据列 $Validcolumn[k]$，其中，$k=n+parity\_length-m$. 选取的数据块为失效 Block 分块所在的 Column 数据列的有效数据块. $Validcolumn[k]$为译码操作的输入参数；

3. 使用 ErasuredCode 编码生成器对输入的数组进行编码操作，输入参数为 $Encoder.encode(AuxiliaryArray, ValidColumn)$，输出结果为 Column 数据列的源数据 $OutputColumn[n+parity\_length]$.

4.4　计算方法的改进

由上文叙述可知，在 VanHDFS 策略的编译码运算过程中，最主要的计算量是来自范德蒙矩阵和构造矩阵之间的加法与乘法运算. 因此，为了降低矩阵运算的计算复杂度，减少内存压力，D-VanHDFS 策略将二进制矩阵与伽罗华域上的元素建立一个映射关系，将矩阵运算中的加法运算转换成为伽罗华域上的逻辑“与”运算，将乘法运算转换为逻辑“异或”运算，进而大幅提高范德蒙码的编译码效率.

从数学上有限域理论的角度分析，若 w 次本源多项式 $P(x)=a_0+a_1x+a_2x^2+\cdots+a_wx^w$ 的系数全部取自于二进制域 $GF(2)$ 中，那么伽罗华域中任意一个 $GF(2^w)$ 域上的元素都可以映射到 $GF(2)$ 二进制域，对应的 $w\times w$ 阶二元矩阵 $\boldsymbol{W}$ 可以用式(7)所示的矩阵表示.

$$\boldsymbol{W}=\begin{bmatrix}0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ a_0 & a_1 & a_2 & \cdots & a_{w-1}\end{bmatrix} \tag{7}$$

矩阵 $\boldsymbol{W}$ 称为伽罗华域 $GF(2^w)$ 的本源二进制镜像矩阵. 通过镜像矩阵 $\boldsymbol{W}$ 可以将范德蒙码矩阵转换成二进制生成矩阵，即将 $k(k+m)$ 的生成矩阵转换为 $(w\cdot k)(w(k+m))$ 二进制矩阵. 转换过程示意图如图 7 所示.

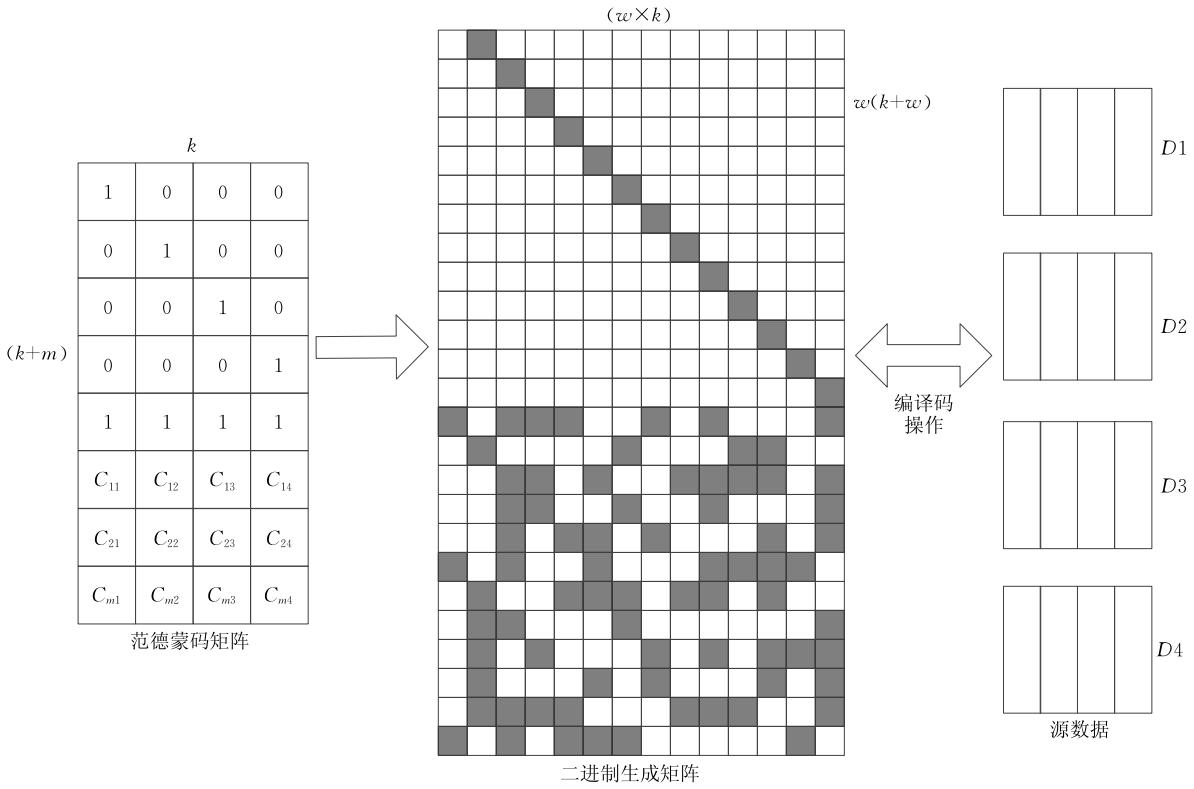

图 7　二进制矩阵转换过程示意图

通过矩阵转换，很大程度上简化了 $GF(2^w)$ 域中的乘法运算，将矩阵运算转换成伽罗华有限域上的“AND”和“XOR”逻辑与运算，降低了 VanHDFS 策略运算复杂度和内存的运算压力.

4.5　译码触发时机选取

范德蒙码编码后的 Group 分组被分成了若干 Column 数据列，每个 Column 数据列同样都由原数据部分和校验码部分构成. 在实际的操作中，用户最关心的是原始数据部分，因为原数据部分是直接需要进行操作的部分，所以，当原数据部分中的 Block 数据块存在信息丢失或者数据失效时，必须立刻进行数据恢复操作. 校验码部分的作用是当数据块的数据损坏或者丢失时，通过范德蒙码译码操作进行数据恢复，因此，校验码部分本身并不是我们真正需

要进行读写操作的数据，故当校验码部分的数据失效时，可以延迟进行恢复操作.

D-VanHDFS 策略不同于 VanHDFS 策略，当一个 Block 数据块文件损坏时，D-VanHDFS 可以采取两种恢复方式，即副本复制恢复方式和范德蒙码译码恢复方式. 因此需要恰当的选择两种恢复策略进行的时机来提升整个系统的效率. 图 8 所示的为系统译码时机选取的整体过程流程图.

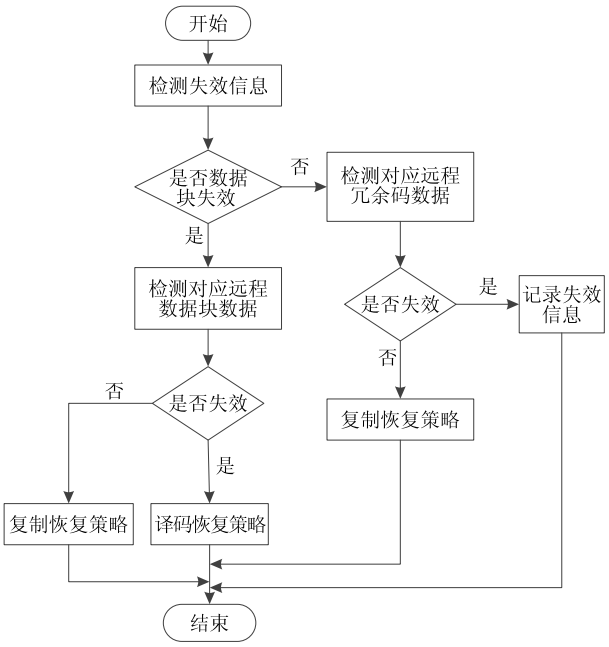

图 8 译码时机选取规则流程图

4.5.1 校验码部分译码时机的选取

校验码部分实质上是恢复操作的冗余编码，与实际的数据操作无关，因此，需要优先考虑系统效率的问题.

触发规则：在机架 m(机架 n)中，当节点 x(节点 z)上的 Group 分组中产生 R 个失效校验码块，先检测机架 n(机架 m)上对应的节点 z(节点 x)，若对应校验码块有效，则直接进行副本恢复方式；若对应校验码块失效，则触发器只记录下失效的校验码块的信息，不进行任何恢复操作. 当数据块部分进行译码操作时，通过查询之前记录的校验码块失效的信息，以该失效记录为标志，对存在校验码块失效的Column 数据列进行范德蒙码译码恢复操作，同样，对于对应的 Column 数据列，只进行数据块部分的副本恢复策略，校验码块不进行其他操作.

4.5.2 数据块部分译码时机的选取

数据块部分的数据是实际使用的真实数据，因此，必须优先考虑数据的正确性.

触发规则：在机架 m(机架 n)中，当节点 x(节点 z)上的 Group 分组中产生 R 个失效数据块，先检测机架 n(机架 m)上对应的节点 z(节点 x)上的数据块信息，若对应数据块有效，则直接进行副本恢复方式，并同时恢复失效的冗余校验码块；若对应数据块失效，则触发器记录下该数据块信息，并查询冗余校验码块失效记录，选取其中一个存在冗余校验码失效记录的 Column 数据列进行范德蒙码译码恢复方式，之后再对对应的 Column 数据列的失效数据块部分进行副本复制恢复策略.

综上所述，D-VanHDFS 策略这种数据块/校验码块分别进行触发恢复操作的方式，增加了 HDFS 的灵活性，并且在极大程度上减少了译码操作触发的频率，降低了文件系统存储时间开销，提高了系统整体的存储效率.

5 范德蒙码优化策略评测与分析

实验平台依托搭载了 Hadoop 的主机构成的小型集群环境，Hadoop 的版本为 0.20.203，操作系统为 Ubuntu-11.10，JDK 版本为 6u30-linux-i586.

5.1 存储开销

表 1 所示的为各备份策略存储开销情况的数据对比，HDFS 采用默认的 3 副本复制策略，Block 分块的大小为默认的 64M.

表 1 存储开销对比

数据列	容错度	D-VanHDFS	FEC	GE	XOR	HDFS
64	2	2.06	2.01	1.95	2.00	3.00
64	3	2.09	2.03	1.98	2.00	3.00
128	3	2.05	2.06	2.01	2.04	3.00
128	4	2.06	2.08	2.04	2.04	3.00
256	4	2.03	2.10	2.08	2.07	3.00
256	5	2.05	2.13	2.11	2.07	3.00

由表 1 所示的存储空间开销比率可知，D-VanHDFS策略同其他优化策略的存储开销情况相近，与原 HDFS 系统的 3 副本策略相比可以节约 30%～32%的存储空间开销. 但随着数据量的持续增大，D-VanHDFS 策略相比其他策略存储开销的节约程度会逐渐增高，优势也越来越明显，这符合 HDFS 海量数据存储的特征.

5.2 可靠性

表 2 所示的为各备份策略的数据可靠性对比. 其中，D-VanHDFS 策略的动态副本数设置为 2，容错度设置为 3，HDFS 采用默认的 3 副本复制策略，表格中“√”代表可正常工作，“×”代表有概率出现失效情况.

表 2　数据可靠性对比

失效类型	失效数目	D-VanHDFS	FEC	GE	XOR	HDFS
机架	1	√	×	×	×	√
机架	≥2	×	×	×	×	×
节点	1	√	√	√	√	√
节点	2	√	√	√	×	√
节点	3	√	×	√	×	×
节点	≥4	×	×	×	×	×
数据块	1	√	√	√	√	√
数据块	2	√	√	√	√	√
数据块	3	√	√	√	√	×
数据块	4	√	√	√	×	×
数据块	5	√	×	√	×	×
数据块	6	√	×	×	×	×
数据块	≥7	×	×	×	×	×

由表 2 所示的对比数据可知，D-VanHDFS 策略采用的是译码恢复和副本恢复的双重恢复策略，大幅降低了数据失效问题，与原 HDFS 策略相比数据可靠性提高了约 200%，在各种情况下的可靠性也均优于其他副本备份策略. 同时，D-VanHDFS 策略还可以在保证存储开销变动极小的情况下，通过控制冗余校验行的数目来动态提升可靠性，很大程度上增加了文件系统的灵活性.

5.3　负载均衡能力

D-VanHDFS 策略可以显著改善 HDFS 的负载均衡能力. 图 9 所示的为各备份策略的节点资源负载均衡能力对比图. 其中，横坐标表示的为在系统集群中选取的 10 个 Datanode 节点组成的小型集群环境，纵坐标表示的为随机读取 500 次数据的节点资源负载标准差的绝对值.

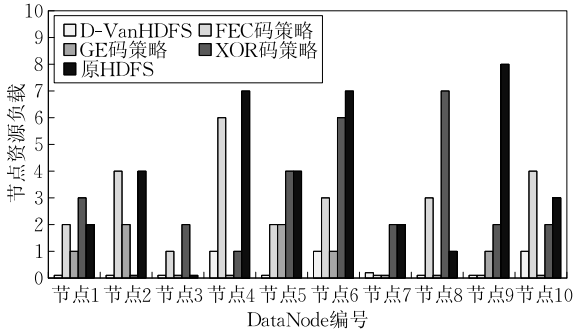

图 9　节点资源负载均衡情况对比

D-VanHDFS 中采用了分散式动态副本的策略，所有的文件都分散的存储在各个节点上，负载被平均分配到整个集群中，故各个节点的资源利用率浮动范围很小，负载均衡. 如图 9 所示，D-VanHDFS 策略拥有最佳的负载均衡能力，而其他备份策略的系统资源负载波动比较大，即会产生某些节点负载过重，而另一些节点却几乎没有负载的问题，这直接影响了系统运行的稳定性.

5.4　译码恢复效率

D-VanHDFS 策略的译码操作中，计算模式变为了静态计算，其时间复杂度远低于矩阵的运算. 图 10 所示的为实验环境下各备份策略进行数据恢复的时间对比. 其中，D-VanHDFS 策略的动态副本数设置为 2，容错度设置为 3，HDFS 采用默认的 3 副本复制策略.

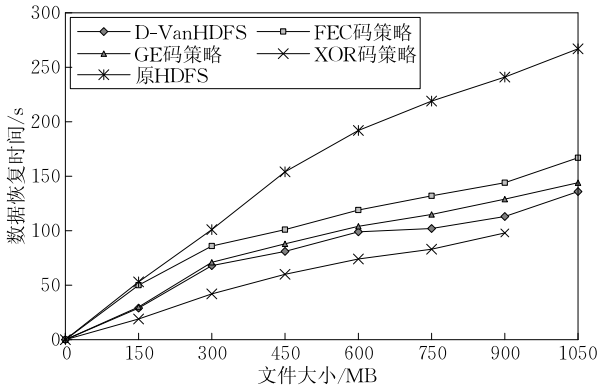

图 10　文件系统数据恢复效率对比

由图 10 所示的系统译码效率对比图可知，与原 HDFS 存储策略相比，D-VanHDFS 的译码恢复效率平均提升约 40%左右. 随着数据规模的不断增大，各备份策略在数据的恢复时间上都是以线性递增的. 其中，XOR 码优化策略的数据恢复效率最高，但在数据量达到一定程度时，数据概率性丢失的问题就会愈发明显；除此之外，D-VanHDFS 在数据的恢复效率上均略高于其他优化策略，在降低存储开销的同时也保证了系统的存储效率.

综上所述，本文提出的 D-VanHDFS 策略在存储开销程度，数据的可靠性，系统资源负载均衡能力及数据的恢复效率等方面的综合表现均优于其他备份策略.

6　结　论

伴随着大数据概念逐渐成为研究的热点，各个领域的相关技术都会产生很大的变革. HDFS 作为新型文件存储系统，以其高容错性和低成本的特点广泛受到人们的关注并迅速发展. 基于范德蒙码的 HDFS 优化存储策略改进了 HDFS 自身存在的缺陷，通过范德蒙码编译码操作很大程度上降低了存储的空间开销，节约了 HDFS 整体的存储成本，配合分组分列策略和有限域理论，改进计算方案，提高存储操作的灵活性，降低矩阵计算的时间开销和计算内存的压力，同时，引入动态副本分散控制的思想，使数据的可靠性以及负载均衡能力有了显著提升，为未来 HDFS 的发展提供了一条有效途径.

参 考 文 献

[1] Feng Deng-Guo, Zhang Min, Li Hao. Big data security and privacy protection. Chinese Journal of Computers, 2014, 31(1): 246-258(in Chinese)
(冯登国, 张敏, 李昊. 大数据安全与隐私保护. 计算机学报, 2014, 31(1): 246-258)

[2] Panian Z. A new data management challenge: How to handle big data//Proceedings of the International Conference on Humanities, Geography and Economics. Dubai, UAE, 2013: 47-51

[3] Shi Ying-Jie, Meng Xiao-Feng. A survey of query techniques in cloud data management system. Chinese Journal of Computers, 2013, 36(2): 209-225(in Chinese)
(史英杰, 孟小峰. 云数据管理系统中查询技术研究综述. 计算机学报, 2013, 36(2): 209-225)

[4] Rousseau R. A view on big data and its relation to informetrics. Chinese Journal of Library and Information Science, 2012, 5(3): 12-26

[5] Chen Bao-Chun. The Cloud File System Based on Erasure Code and HDFS[M. S. dissertation]. Jilin University, Jilin, 2012(in Chinese)
(陈宝纯. 基于纠删码与 HDFS 的云文件系统[硕士学位论文]. 吉林大学, 吉林, 2012)

[6] Zhou Song. Research on Erasure Coding Based Fault-Tolerant Storage Technology for Data Intensive Super Computing [M. S. dissertation]. National University of Defense Technology, Changsha, 2010(in Chinese)
(周松. 面向数据密集型超级计算的基于纠删码的容错存储技术研究[硕士学位论文]. 国防科技大学, 长沙, 2010)

[7] Zhu Yun-Feng, Lee P P C, Hu Yu-Chong, et al. On the speedup of single-disk failure recovery in XOR-coded storage systems: Theory and practice//Proceedings of the 28th IEEE Conference on Massive. London, UK, 2012: 106-114

[8] Dong Xin-Hua, Li Rui-Xuan, Zhou Wan-Wan, et al. Performance optimization and feature enhancements of Hadoop system. Journal of Computer Research and Development, 2013, 50: 1-15(in Chinese)
(董新华, 李瑞轩, 周湾湾等. Hadoop 系统性能优化与功能增强综述. 计算机研究与发展, 2013, 50: 1-15)

[9] Cui Ji-Feng, Zhang Yong, Li Chao, Xing Chun-Xiao. A packaging approach for massive amounts of small geospatial files with HDFS//Proceedings of the Web-Age Information Management. Beijing, China, 2012: 210-215

[10] Dong Bo, Zheng Qing-Hua, Tian Feng, et al. Performance models and dynamic characteristics analysis for HDFS write and read operations: A systematic view. Journal of Systems and Software, 2014, 93: 132-151

[11] Nie Rui-Hua, Zhang Ke-Lun, Liang Jun. Improved fault-tolerance mechanism in cloud storage system. Application Research of Computers, 2013, 30(12): 3725-3728 (in Chinese)
(聂瑞华, 张科伦, 梁军. 一种改进的云存储系统容错机制. 计算机应用研究, 2013, 30(12): 3725-3728)

[12] Wang Yong-Gang, Wang Sheng. Research and implementation on spatial data storage and operation based on hadoop platform//Proceedings of the 2010 2nd IITA International Conference on Geoscience and Remote Sensing. Qingdao, China, 2010: 275-278

[13] Harold C L, Shivnath B, Jeffrey S C. Automated control for elastic storage//Proceedings of the 7th International Conference on Autonomic Computing (ICAC'10). Washington, USA, 2010: 1-10

[14] Zhao Tie-Zhu, Yuan Hua-Qiang. Performance analysis of distributed file systems for data-intensive applications//Proceedings of the 2013 IEEE International Conference on Computer Science and Automation Engineering. Guangzhou, China, 2012: 1417-1420

[15] Li Xiao-Kai, Dai Xiang, Li Wen-Jie, Cui Zhe. Improved HDFS scheme based on erasure code and dynamical-replication system. Journal of Computer Applications, 2012, 32(8): 2150-2153, 2158(in Chinese)
(李晓凯, 代翔, 李文杰, 崔喆. 基于纠删码和动态副本策略的 HDFS 改进系统. 计算机应用, 2012, 32(8): 2150-2153, 2158)

[16] Ashish T, Zheng S, et al. Data warehousing and analytics infrastructure at facebook//Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD'10). Indiana, USA, 2010: 1013-1020

[17] Hsiao H C, Chung H Y, Shen H, et al. Load rebalancing for distributed file systems in clouds. Parallel and Distributed Systems, 2013, 24(5): 951-962

[18] Zhu Yuan-Yuan, Wang Xiao-Jing. HDFS optimization program based on GE coding. Journal of Computer Applications, 2013, 33(3): 730-733(in Chinese)
(朱媛媛, 王晓京. 基于 GE 码的 HDFS 优化方案. 计算机应用, 2013, 33(3): 730-733)

[19] Luo Xiang-Hong, Shu Ji-Wu. Summary of research for erasure code in storage system. Journal of Computer Research and Development, 2012, 49(1): 1-11(in Chinese)
(罗象宏, 舒继武. 存储系统中的纠删码研究综述. 计算机研究与发展, 2012, 49(1): 1-11)

SONG Bao-Yan, born in 1965, Ph. D. professor. Her research interests include database theory and techniques, RFID event stream processing techniques, massive data processing techniques and graph data processing techniques.

WANG Jun-Lu, born in 1988, M. S. candidate. His main research interests include database theory and techniques, big data processing techniques and massive data processing techniques.

WANG Yan, born in 1978, Ph. D. candidate, associate professor. Her main research interests include database theory and technology, sensor data processing technology and massive data processing technology.

Background

With the advent of Big Data era, all kinds of the traditional data processing techniques have become increasingly unable to adapt the volume of the big data era. The new file management architecture HDFS begins to enter into the vision of people, and gains more and more attention of people.

Through the analysis and development of the HDFS storage framework, this paper identifies several problems of the storage framework, and proposes the storage optimization approach based on Vandermonde code to improve the HDFS based on the problem of the lack of the processing abilities of the existing solutions. In addition, according to the situation of the practical applications, this paper adopts the idea of distributed dynamic copy control to optimize the whole system. This paper optimizes the time cost and computational memory pressure of the Vandermonde code decoding by dynamically controlling the number and position of the copies to enhance the reliability of the whole system. The storage efficiency where the condition of the hardware implementation is not high can be improved largely, and the system reliability, storage cost, the execution efficiency and load balancing ability are all improved drastically, which can provide an effective way for the future development of HDFS.

This research is supported by the National Natural Science Foundation of China under Grant Nos. 61472169 and 60873068, and the Excellent Talent Support Program of Education Department of Liaoning Province, China under Grant No. LR201017.