# 《说明(1)》检测报告

WriteCheck 专注论文检测 用心提供服务

## 总相似率：26.24%

## 基本信息

| 文档名称 | 说明(1) | 报告二维码防伪 |
| --- | --- | --- |
| 报告编号 | f8f82252-8200-4086-9fbd-e943f9e4a845 | |
| 文档字数 | 23571 | |
| 提交人姓名 | - | |
| 提交方式 | 粘贴文本检测 | |
| 检测范围 | 美国（Wiley InterScience、IEEE、EBSCO、ProQuest、Netlibrary...）；荷兰（Sciencedirect、OVID...）；英国（Ingenta、Blackwell...）；德国（Springer...）；小语种（日、俄、法等）资源库; 世界知名论文/期刊/书本资源库; 世界名校资源库; PaperRight云论文库 | |
| 提交时间 | 2017-04-25 22:14:30 | |

## 检测报告指标详情

| 原创率 | 抄袭率 | 抄袭片段 | 抄袭字数 | 总字数 |
| --- | --- | --- | --- | --- |
| 73.76% | 26.24% | 103个 | 6186字符 | 23571字符 |

## 相似片段位置图


注：红色部分为论文相似部分，

## 相似片段详情（仅显示前10条）

| 序号 | 来源 | 对比库 | 相似率 |
| --- | --- | --- | --- |
| 1 | www.webko.cn | PaperRight云资源库 | 9% |
| 2 | www.cnblogs.com | PaperRight云资源库 | 2% |
| 3 | guhuodewu.zhaou.com.cn | PaperRight云资源库 | 1% |
| 4 | www.zhangqi2.com.cn | PaperRight云资源库 | 1% |
| 5 | www.n5188.com | PaperRight云资源库 | 1% |
| 6 | myhostlist.com | PaperRight云资源库 | 1% |
| 7 | www.thebigdata.cn | PaperRight云资源库 | 1% |
| 8 | research.microsoft.com | PaperRight云资源库 | 1% |
| 9 | cso.ccw.com.cn | PaperRight云资源库 | 1% |
| 10 | www.scholat.com | PaperRight云资源库 | 1% |

**文档原文标注**

EVENODD

与实现 学生姓名 江小玉 院系 名称 计算机科学学院 专业

名称 计算机科学与技术 班级 2013级1班 学号 2013110116 指导教师 韩鸿宇 完成时间 2017年5月05日分布式存储系统中基于EVENODD码的容错技术研究与实现学生：江小玉指导教师：韩鸿宇内容摘要：2014年，“大数据”作为新技术被首次写入政府工作报告，大数据具有海量化、多样化的特征，而这样的特征

给世界带来了新的挑战和机遇。

大数据已经走进了人们的生活，比如“城市计算”就是利用

计算机科学，以城市为背景，与 城市规划、交通、能源、环境、社会学和 经济学 等学科融合的新兴领域。城市计算通过 城市感知 不断获取、整合和分析城市中不同领域的大数据来解决城市所面临的挑战（

如环境恶化、交通拥堵、能耗增加、规划落后等）的过程[[

endnoteRef:0]][[endnoteRef:1]]。数据驱动的新型业务模式日益普遍，大数据时代的技术等各方面环境有新的变化，传统的数据安全保护方案也需要与时俱进。传统存储技术难以适应大数据时代。 [0: []

Yu Zheng, Licia Capra, Ouri Wolfson, Hai Yang. Urban Computing: concepts, methodologies, and applications. ACM Transaction on Intelligent Systems and Technology. 5(3), 2014 ] [1: []城市计算概述

郑宇.城市计算概述，武汉大学学报. 2015年1月，40卷第一期]

传统存储技术依靠本地集中式存储，硬盘一旦出现损坏或者操作系统出现故障，数据都将面临丢失或者失效的困境。当庞杂的数据进行存储时，传统存储技术无法保证数据 的安全性及可靠性。为了提高存储的高可靠性和高可用性，早期研究人员采用备份技术作为冗余手段，当一个节点的数据流失时，可以从其他的备份节点复制数据恢复原数据，以此重构故障盘符中丢失的数据。但是节点可靠性较低，复制策略增加了巨大的存储开销，在庞大的数据量面前显得十分低效。 RAID的出现，将多个磁盘组织成为一个逻辑盘，一度有效的提高了磁盘的性能的存储容量。但是随着时代的发展，RAID也不能完全满足海量数据存储的需求。为了降低冗余度，减少成本，提高存储效率，研究人员们在RAID的基础上又采用纠删码对文件进行编码存储。编码的方式非常多，例如EVENODD[[endnoteRef:2]]就是RAID-6常用的一种编码策略。适用于恢复任意俩个磁盘信息失效的情况。Hadoop内置的HDFS文件系统的NameNode/DataNode可以提供数据切片并冗余存储,平衡数据的功能。Hadoop的分布式存储系统，可在廉价的服务器上搭建大规模的结构化存储集群。 Facebook在Hadoop上已经实现了纠删码模块HDFS-RAID，本论文在此研究结果基础上设计实现HDFS-EVENODD模块。 [2: []

BLAUM M，BRADY J，et al. EVENODD:an efficient scheme for tolerating double sisk failures in RAID archeitectures [J].IEEE Trans. Comput ,1995,44(2):192-202]

关键词：大数据城市计算冗余手段纠删码HADOOP HDFS RAID EVENODD The Technoloty of Packet Interception in Kernel Layers Abstract: Because the network security problems have gone from bad to worse, computer users begin to notice network security technology and network security products. Among all kinds of security technologies, the technology of packet interception in kernel layer is very important for it is the basis for packet filter

firewall and it is widely used in packet filtering, protocol conversion, interception of message analysis and network sniffering. Therefor, I devote myself to studying on the technology of the packet interception in kernel layer.In this article, firstly, the present situation of this technolony is described to tell reader why I study on it. Secondly, principles of packet interception are told about in detail.In this detailed description part, the concept

of the LSP ,the theory of the LSP and the usage of LSP are

introduced in more detail. What' s more, I also explain many aspects of the network drivers in kenel layer and tell reader how to implement the interception.Thus, reader can understand the technology of packet interception in kernel layer accurately. Finally, to implement and check out the technology I have discussed, I compile the software named NetFilter which is analyzed and designed systematically in this article. By testing , the software can intercept packets in kenel layer and analyze packet in application layer. Key words: network security kernel layer packet interception 目录 1

绪论1 1.1研究目的和意义1 1.2国内外研究现状1 1.3主要 贡献2 1.4 文章 的 结构 2

2 Hadoop平台介绍2 2.1 Hadoop生态系统框架3 2.2浅析Hadoop核心组件HDFS 4 2.3 Hadoop相比RAID 4 2.4 Hadoop集群部署方法5 3 EVENODD算法详解 3.1有限域的介绍 3.2 MDS阵列 3.3 EVENODD算法 3.3.1编码算法 3.3.2任意两个磁盘失效修复算法 3.3.3单个磁盘出错修复过程 4 EVENODD基于Hadoop平台的实现 4.1 HDFS-RAID实现过程 4.2 HDFS-EVENODD设计实现 4.2.1在WINDOWS系统下的实现 4.2.2在HADOOP系统下的实现 5

总结5.1性能分析5.2总结 6结束语 7 致谢 8 参考文献

分布式存储系统中基于EVENODD码的容错技术

研究与实现1绪论1.1研究 目的和 意义《

2017中国大数据发展报告》，该报告基于

国家发改委互联网大数据分析中心、国家信息中心“

一带一路”大数据中心所掌握的30多个种类，总计40多亿条数据，对我国大数据产业发展进行全面分析[[endnoteRef:3]]。数据的最大价值不在量大，而在于对数据 的分析应用，但是如何存储，提取数据是一个急需解决的问题。 [3: [] 2017中国大数据发展报告国家信息中心、南海大数据应用研究院. 2017年3月7日]随着科技发展，各种传感器技术以及云计算、云存储单位日趋成熟，我们有了社交媒体、交通流量、气象、地理等多种数据。数据存储的方式不仅限于文字、图片、视频...数据的多元性和多源性绕不开对数据的存储、管理、提取和分析。当前，大数据已经

从互联网领域 延伸至 电信、金融、地产、贸易等各行各业，

与

大数据市场 相关联 的新技术、新产品、新服务、新业态不断涌现，

并不断融入社会公众生活。各行各业通过新媒体形式迅速产生庞大复杂的数据。汹涌而来的大数据时代，数据在传统技术下难以存储和传输

。基于大数据的网络用户行为 数据 分析，

百度、Goolge、阿里巴巴、Facebook等大型网站的总数据量早已超过数千PB，未来将以ZB为单位计算[[endnoteRef:4]]。而据2011年的IDC研究报告预测，全球的数据总量在

到2020年这未来的十年里将

增长近50倍[[endnoteRef:5]]。 [4:

[]基于大数据的网络用户行为分析左军，《

软件工程师》.2014年10期] [5: []

IDC：2011年全球数据产生量达到1.8ZB到2020年将增长50倍 新浪科技IDC .2011年06月

29日]数据存储最简单的一种冗余机制是复制策略，以存储原始数据的N个副本来保证N-1个擦除的容错性，以牺牲存储空间为代价提高系统的性能。巨大的存储容量必然要求更多高性能的服务器，这无疑将增加昂贵的开发成本，所以在保证可用性下如何优化成本 是迫切需要解决的问题。1988年

，RAID（Redundant Array of Inexpersive Disks），即 “廉价磁盘冗余 矩阵”概念 的

提出[[endnoteRef:6]]，有效的解决了磁盘存取速度与CPU处理速度不匹配对计算机整体性能的掣肘。 RAID的设计逻辑是通过对多个独立的磁盘并行操作，提高输入输出速率，提供更大的存储量。由于RAID采用分布式存储，将数据存储在不同的磁盘，节点可靠性较低，在成百上千个节点同时工作时，必须有良好的容错性。 [6: []

D.A.Patterson,G.A.Gibson,R.H.Katz.” A Case for Redundant Arrays of Inexpensive Disks(RAID)” In:Proc.of ACM SICMOD ,June

1998.109-116]传统的技术和方式无法满足海量数据增长的挖掘应用、分析处理以及存储需求。Hadoop作为研究大数据的一项重要技术，拥有存储和处理千兆字节以上数据的扩容能力，将普通的计算机组成数千个节点的服务器集群来分发数据并

在数据所在的节点上并行地处理

数据，实现低成本和高效率，为了提高可靠性。Hadoop集群的优势由其分布式存储文件系统—HDFS的分布式特点而来，通过网络在多台主机上实现共享，即使系统中某些节点损坏，利用数据副本维持运作而不会有数据损失。HDFS通过复制保存副本保障系统的容错能力，底层数据存储默认使用三副本冗余以保障数据的可靠性，存储浪费与数据总量成正比，造成巨大的存储浪费，降低系统的性能。HDFS也以牺牲存储空间来提高系统可靠性，因为它的设计原理主要是基于复制冗余和镜像冗余，而针对HDFS数据多副本容灾设计而带来的存储空间利用率低，恢复效率低等问题，国内很多 学者结合纠删码的思想，引入编码译码模块，保证了集群数据的安全性，提高数据恢复的速度，减少了空间代价，降低了整体的存储成本。为了降低冗余度，提高效率，许多研究人员使用纠删码技术作为分布式存储系统中一种冗余机制来保证数据的可靠性。纠删码技术的基本思想就是

：首先将要存储在系统中的文件分割成M

个块（M根据系统节点的个数设置），然后采用纠删码技术的编码技术对这些块编码得到N个文件块(N > M)，将这些文件碎片分布到系统的不同节点上。当系统节点失效或者文件块丢失时，只需要从这些节点中获得M’ (M’ > = M)个可用的文件片，通过译码技术就可以恢复数据，得到原始文件。使用纠删码算法对文件进行编码存储，可以减少存储副本数目，提供合理的的额外存储，避免了过度的存储开销，如果部分磁盘损坏或丢失，可以通过译码恢复源文件，保障了系统的高可靠性

和可用性，给系统的性能带来了数量级的提高。本论文研究的便是分布式存储系统中基于EVENODD码的纠删码容错技术。 1.2国内外研究现状2012年，Goplan等人提出了局部修复性（repair locality）[[endnoteRef:7]]。称一个码的第i位有局部性r（locality r），如果该位上的值可以通过读取码的其他不超过r个位上的值来修复。如果存在一个由信息位组成的信息集，使得集合中的每一位都有局部性r，称此码满足信息位局部性r（information locality r）。类似的，如果一个码的所有 位都具有局部性r，那么称该码具有任意位局部性（all symbol locality r）满足局部修复性的编码能够有效的降低修复过程中帮助节点的访问数，从而降低修复的复杂度。 [7: [] Gopalan, P., Huang, et al. On the Locality of Codeword Symbol

[J]. IEEE Transactions on Information Theory, 2012, 58(

11): 6925-6934.] 2014年，付园[[endnoteRef:8]]针对HDFS原有冗余机制的不足，在分析现有改进方法的基础上，设计了结合完全备份和改进的RS（Read-Solomon）纠删码两种冗余方法的优化数据冗余策略RIRS（Replication Improved RS）。并且利用上述策略和模型对源代码进行修改，实现了基于HDFS的优化数据冗余策略。并在自主搭建的Hadoop集群上分别

对系统的功能和性能进行了

测试。 [8: []付园.基于HDFS的优化数据冗余策略的研究[D].硕士论文.吉林大学，2014 ] 2015年，王安宇[[endnoteRef:9]]针对局部修复码三种典型的局部修复性：局部修复性r，局部修复性(r，δ)，局部修复性(r，δ)c，讨论了相应码的极小距离上界以及达到最优极小距离的码的构造等问题，极小距离较大的局部修复码能够保证系统有较高的整体容错能力。 [9: []王安宇.局部修复码中的若干问题[D].博士论文.中国科学院大学，2015] 1.3主要贡献 （1）、根据国内外研究

发展现状 分析 大数据的 现实 应用，以及 在 大数据

时代传统存储技术的弊端和纠删码在冗余存贮技术中的优势。 （2）、详细介绍了Hadoop平台的组成和结构，分布式文件系统的工作原理，和RAID技术及RAID的发展，以及搭建Hadoop伪分布式环境的配置详细步骤。 （3）、对EVENODD的算法进行了详细的讲解，并举例演绎了编码和译码的过程。对EVENODD编码技术中涉及的有限域，MDS阵列等概念都做了介绍。并在Windows环境下实现单个矩阵的编码译码以及实现丢失图片的恢复。 （4）、在Facebook的HDFS-RAID模块上设计实现HFDS-EVENODD模块。 1.4文章的结构主要内容： 1)、学习Hadoop体系结构以及周边框架；搭建基于Linux系统环境的Hadoop伪分布式环境，运行实例测试Hadoop环境是否搭建成功，包括从本地上传、下载、修改、删除HDFS文件系统中的文件等；重点学习HDFS模块的工作原理 2)、阅读Facebook版本的Hadoop源码，源代码实现过程复杂，结构十分庞大，提供了大量的接口以及抽象类，需理解项目的编译方式、部署方式和配置文件的结构，着重学习RAID模块的运行流程以及相关类的实现3)、编码基于有限域和素数域，补充线性代数，有限域的相关知识4)、学习纠删码的基础理论知识，包括MDS阵列的概念及构造，RAID技术的起源、成熟和发展等。理解纠删码编码译码的思想，例如X码，B码[[endnoteRef:10]]，S码[[endnoteRef:11]]等这样与EVENODD编码一样可以容许两个存储设备同时故障的数据分布策略的编码[10: [] XU L.

Highly available distributed storge systems[D]. Pasadena, California: California Institute of Technology,

1997.] [11: [] KATTI R, RUAN Xiao-yu. S-

code: New distance-3 MDS array codes with optimal encoding[ C ]//In: Proceedings of IEEE ICASSP’ 05.

Piscataway, NJ: IEEE Signal Processing Society,2005. ] 5)、学习EVENODD编码及译码的算法原理，并使用java语言实现。先在windows下实现基本功能，再在Hadoop平台基础上，嵌入Hadoop源码中进行二次开发6)、结合搭建的Hadoop伪分布式环境测试EVENODD代码模块的功

能，是否能恢复删除的文件块 7) 、对EVENODD编码的性能经行分析，与其他纠删码的性能进行比较安排

：本论文 一共 分为5章第1章绪论。

主要分析论文研究的背景和意义，国内外发展的情况，以及本论文的行文思路和内容安排。第2章Hadoop平台介绍。主要介绍了研究大数据的方法，着重研究Hadoop平台，讲解Hadoop基础框架结构，包含的组件；介绍了RAID的工作原理，以及搭建伪分布式环境的流程；第3章EVENODD算法详解。主要介绍了EVENODD的算法逻辑，举例演绎此算法编码译码的过程第4章EVENODD基于Hadoop平台的实现。在基于Hadoop的RAID模块上，研究代码的层次，设计实现EVENODD；在windoWS下先实现了对单个矩阵的编码、擦除和恢复后，再实现了一个图片分块、编码以及擦除任意俩块和恢复后，将代码移植到了Hadoop平台上，实现HDFS-EVENODD模块。第5章总结。分析EVENODD编码的性能并进行总结2 Hadoop基本架构 2.1 Hadoop生态系统框架Hadoop作为研究大数据的重要部件

，是一个基于JAVA 语言 的分布式 软件框架，进行 数据处理和数据分析， 可用于海量数据 的 存储 。Hadoop

的核心设计为

分布式文件系统(Hadoop distributed file system,HDFS)和MapReduce编程

框架。 Pig：基于Hadoop的数据流系统

，Pig可加载数据，转换数据格式以及存储最终结果等一系列过程

Hive：基于Hadoop的数据仓库，可以将 数据化 的数据文件映射为一张数据库表，

可向

HDFS添加数据，并允许使用类似SQL的语言进行数据查询，

可以将sql语句转换为MapReduce任务进行运行Zookeeper：分布式 协作 服务，

解决分布式环境下的数据管理问题，统一命名，状态同步，集群管理

为HBase提供稳定服务和failover机制Avro：数据系列化

系统，用于支持大批量数据 交互 的应用

MapReduce：分布式计算框架，超大型数据集的并行计算框架，负责计算。 Hbase：分布式列存数据库

，面向列的分布式存储系统，用于在Hadoop中支持大型稀疏表的列存储数据环境，数据

库集群，位于结构化存储层

，是一个开源的非关系型分布式数据库（ NoSQL ），基于 谷歌的BigTable建模，

是一个高可靠性、高性能、 高伸缩 的分布式存储系统

HDFS：分布式文件系统，负责存储数据图2.1 Hadoop的生态系统架构2.2 Hadoop核心组件

HDFS简介HDFS可以管理多台机器上的文件，位于系统底层，存储管理Hadoop集群所有节点上的数据

，是Hadoop体系中数据存储管理的基础

部件。HDFS

能检测和应对硬件故障，在低成本的硬件上运行,

通过网络管理分布在不同主机存储的文件系统，让多台

机器上的多用户分享文件和存储空间，

在

程序与用户看来，就 如同 访问本地的磁盘一般，

在遇到故障时 能 继续运行不让用户 明显 察觉到中断。

HDFS以流式数据访问模式来存储超大文件， 提高吞吐量 。HDFS 以块Block 的

形式操纵文件。HDFS的安全性和可靠性也成为目前研究的热点难点。 HDFS适合大文件存储，不适合小文件。NameNode将Block分布到不同的 DataNode节点上，小文件的Block越多，对NameNode的内存压力越大。 HDFS为主从式

结构，主节点( NameNode )只有一个，从节点( DataNode)可以 有

多个。当程序运行时，可以在文件目录下找到in_use.lock文件，in_use.lock文件没有什么实际内容，但是这个文件表示name目录已经被NameNode进程占用了，如果在此时启动NameNode，进程就会报错，无法进入，这意味着如果多个进程同时编辑数据会出现问题，in_use.lock文件的存在，表示锁定NameNode节点，其他NameNode进程无法进入,这也表示HDFS不支持并发写情况，所以只能允许一个NameNode存在。下面介绍一些基础的概念： Block：数据块是最基本的存储单位，是linux文件系统划分的一个概念。对于文件内容而言，一个文件的长度大小是Size，那么从文件的0偏移量开始，按照固定的大小，顺序对于文件进行划块分并编号，每一个块称为一个Block。 HDFS默认Block大小是64MB，以一个大小为256M的文件为例，共有256/64=4个Block。与普通文件系统不同的是，HDFS中一个比单个Block块小的文件不会占据整个Block的大小。HDFS可以根据实际情况将Block的大小设置为128M，256M，或者更大的单位。设置方法：复制

< name > dfs.block.size < /name > < value > 67108864 < /value > 到 hdfs-

site.xml文件中修改大小HDFS

Client：切分文件；访问HDFS；与NameNode交互，获取文件位置信息；与DataNode交互，读取数据。

从HDFS中读取数据时，首先连接到Client，Client通过访问NameNode获取到文件包含的所有的Block以及其所在的DataNode，然后Client从各个DataNode上下载数据块。向HDFS写入数据时，首先连接到Client，Client通过访问NameNode获得文件块被分配到的DataNode(DataNode的分配可通过配置策略优化)，最后由Client使用流式数据接口像DataNode写入数据，并从该DataNode向被选为备份节点的DataNode复制数据，作为冗余数据以降低数据损失风险。NameNode：系统的名字节点，管理系统的命名空间。接收Client的操作请求；维护系统的目录结

构和文件；管理文件与数据块之间的关系。为了提升检索速度，文件目录树一般放在内存中，为了持久保存，则写入硬盘中保存。故存储在硬盘上，但运行时在内存中。 DataNode：系统的工作节点，存储实际的数据，文件被分成多个数据块存储在硬盘上，反馈存储信息给NameNode，默认存储多个副本以保障安全性，HDFS可以存储海量数据，实际上指的就是DataNode节点的快速扩展，副本越多会占用磁盘空间。副本默认 是三个，三个副本一般存在三个机器上，一般就近放在最近机架上的电脑上(确定)，同一机架上的另一台电脑(随机)，另一机架的另一台电脑(随机))。目录没有副本，副本越多会占用磁盘空间，可以根据实际情况在配置文件中设定。配置方法：进入hadoop/conf目录下，找到hdfs-site.xml配置文件，可以看到
＜b＞＜name＞dfs.replication＜/name＞ ＜/br＞ ＜ value＞1＜/value＞＜/b＞ ＜/br＞，则可以设置副本数。 SecondaryNameNode：次名字节点，辅助NameNode节点，分担名字节点的工作量。执行过程

：从NameNode上下载元数据信息(fsimage.edits)，然后把二者合并，生成新的fsimage，在本地保存，并将其推送到NameNode ，同时重置NameNode的edits。

名字节点是集中的单一故障点，次名字节点为名字节点创建检查点，在紧急情况下，可用次名字节点恢复名字节点，降低名字节点数据丢失的风险。图2.2 Hadoop的生态系统架构HDFS Client DataNode DataNode DataNode NameNode Secondary NameNode 2.3 Hadoop相比

RAID RAID（Redundant Array of Independent Disks），即“独立 冗余 磁盘阵列”。

在

1987年，Patterson、Gibson和Katz这三位工程师

发表了题为《A Case of Redundant Array of Inexpensive Disks》(《廉价磁盘冗余阵列方案》)的论文，其基本思想就是将多个容量较小的、比较 廉价的 硬盘 进行有机组合， 让它的 性能 可以 超过一个昂贵的大硬盘。 RAID 的

设计原理就是：通过资源冗余来提高服务质量，它将多个

独立的 磁盘（物理硬盘）按照 不同的方式组合起来形成一个硬盘组（逻辑 盘），以 提供比单个磁盘 更高的存储性能，

更大的存储容量和

提供数据备份技术。 而 在用户看来，组成的磁盘组就像是一个硬盘，用户可以对它进行分区、格式化 等操作 ，对磁盘阵列的操作与单个

磁盘一致。RAID通过数据分割、多通道并行读/写来提高I/O速率

，磁盘阵列的存储速度 就 比单个 磁盘的要 高很多。

RAID还通过保存冗余的数据、校验信息以及简单的容错编码来提高存储的可靠性

，在用户数据一旦发生损坏后，利用 这些 冗余信息可以使损坏 的 数据得以恢复。

因此，RAID技术在提高系统I/O速度、数据传输率、数据可靠 性和数据存储容量方面取得了很好的效果。早期的RAID系统使用最简单的

奇偶校验技术 来保障 数据的可靠性，

至多允许一张磁盘失效。但是当系统的规模较大时，同时发生多张磁盘失效的概率比较高，存储系统的可靠性迅速降低。 RAID

经过多年的发展，现在已经有了

许多等级

。RAID技术主要包含 了从 RAID 0 到 RAID 7等数个规范 等级，且 它们的侧重点各不相同，常见的规范有如下几种: RAID 0:

把多个（ 最少2个 ）硬盘合并成1个逻辑盘。

连续以位或字节为单位分割数据， 数据写入 /读

出

时对各硬盘 并行操作 ，不同硬盘写入不同数据，速度 很 快。

但是RAID 0没有冗余数据，所以RAID 0并不能

算是真正的RAID结构，只是单纯地提高性能，并没有为数据的可靠性提供保证， 如果 其中的一个磁盘失效，将影响到所有数据，因此RAID 0不能应用于数据安全性要求高的场合。 所以 RAID 0的优点 是

传输速率高，缺点是无法保障数据的安全性。 RAID 1

：同时对2个硬盘读写（ 同样的数据 ）。

RAID 1通过磁盘数据镜像实现数据冗余,在成对的独立磁盘上产生 相互 备份的数据。当原始数据繁忙时，可直接从镜像拷贝中读取数据，因此RAID 1可以提高读取性能。RAID 1是磁盘阵列中单位成本最高的，但提供了很高的数据安全性和可用性。当一个磁盘失效时，系统可以自动切换到镜像 磁盘 上读写，而不需要重组失效的数据。 所以 RAID 1 的优点 是

强调数据的安全性，缺点是比较浪费。

RAID 0+1：也 称为 RAID 10标准， 即把 RAID 0和RAID 1标准结合的产物，在连续地以位或字节为单位分割数据并且并行读/写多个磁盘的同时，为每一块磁盘作磁盘镜像进行冗余，

比较适合速度要求高，又要完全容错

的情况。所以RAID 1+0的

优点是同时拥有RAID 0的 高速度 和RAID 1的数据高可靠性， 缺点是 CPU占用率同样也更高，磁盘的利用率比较低。

值得注意的是：RAID 0+1最少需要4块硬盘，且做RAID 1+0时

要先作RAID 1，再把数个RAID 1做成RAID 0，

这样比先做 RAID 0 ，再做 RAID 1 有更高的可靠性。

RAID 2：将数据条块化地分布 在 不同的硬盘上，条块 的 单位为位 或者 字节，并使用称为 “加重

平均纠错码(海明码)”的编码技术来提供错误检查及恢复。这种编码技术需要多个磁盘存放检查及恢复信息，使得RAID 2技术实施更复杂，因此在商业环境中很少使用。 RAID 3： 与 RAID 2非常类似，都是将数据条块化分布 在 不同的硬盘上， 但是 区别在于RAID 3使用简单的奇偶校验， 并使用 单块磁盘存放奇偶校验信息。如果一块磁盘失效，奇偶盘及其他数据盘可以重新产生数据；如果奇偶盘失效则不影响数据使用。RAID 3 对 于 大量的连续数据可提供很好的传输率，但对于随机数据来说，奇偶盘会成为写操作的瓶颈。 RAID 4：RAID 4同样也将数据条块化并分布 在 不同的磁盘上，但条块单位为块或记录。 RAID 4使用一块磁盘作为奇偶校验盘，每次写操作都需要访问奇偶盘，这时奇偶校验盘会成为写操作的瓶颈，因此RAID 4在商业环境中也很少使用。 RAID 5：

把多个（最少3个）硬盘合并成1个逻辑盘，数据 写入 /读

出指针可以

同时对阵列设备进行操作，提供更高的数据流量，

且

读写时会建立奇偶校验信息。

RAID 5不单独指定奇偶盘，而是

把

奇偶校验信息和相对应的数据分别 交叉 存储 在 不同的磁盘上。当 RAID 5 的一个磁盘数据发生损坏 时，可以 利用剩下的数据和相应的奇偶校验信息去恢复被损坏的数据。 RAID 5 相当于 RAID 0 和 RAID 1 的综合，

更适合于小数据块和随机读写的数据。RAID 5 与RAID 3 相比， 其中 最主要的区别在于RAID 3 每一次 进行数据传输 都 需涉及到所有的阵列 磁盘 ;而 相 对于RAID 5来说，大部分数据传输 都 只对一块磁盘操作， 并且可以 进行并行操作。在RAID 5中有“写损失”，即每一次写操作将产生四个实际的读/写操作，其中两次读旧的数据及奇偶信息，两次写新的数据及奇偶信息。 RAID 6：与 RAID 5相比，RAID 6增加了第二个独立的奇偶校验信息块。两个独立的奇偶系统使用不同的算法，数据的可靠性非常高，即使两块磁盘同时失效也不会影响数据 的使用。但RAID 6需要分配给奇偶校验信息更大的磁盘空间，相对于RAID 5有更大的“写损失”，因此“写性能”非常差。较差的性能和复杂的实施方式使得RAID 6很少得到实际应用。 RAID 7：这是一种新的RAID标准，其自身带有智能化实时操作系统和用于存储管理的软件工具，可完全独立于主机运行，不占用主机CPU资源。 RAID 7可以看作是一种存储计算机(Storage Computer)，它与其他RAID标准有明显区别。除了以上的各种标准，我们可以如RAID 0+1那样结合多种RAID规范来构筑所需的RAID阵列，例如 RAID 5+3(RAID 53)就是一种应用较为广泛的阵列形式。用户一般可以通过灵活配置磁盘阵列来获得更加符合其要求的磁盘存储系统。

总结：在存储系统中使用RAID技术

的好处主要有以下三种： 一是 通过把多个

小的、廉价的磁盘有机组织

在一起作为一个逻辑卷提供磁盘跨越功能； 二 通过把数据分成多个数据块（ Block ），并行 的 写入/读出多个磁盘以提高访问磁盘的速度； 三是 通过镜像或校验操作提供容错能力。 RAID 和 Hadoop 的

设计原理与思路一样，但是两者不在一个量级上。RAID只能在一台机器上，Hadoop可以部署在成

千上万台机器上。简单点来说，即RAID的并发读写速度提高最多只能是单张磁盘的十几倍，但是Hadoop可以是数万 倍。RAID的可靠性最大程度能做到单张磁盘损坏时数据不丢失，但是Hadoop可以做到很多机器损坏而数据不丢失。 2.4 Hadoop集群部署方法这里使用FackBook的包含RAID模块的源码，在主机上部署为伪分布式环境。在此系统中能进行EVENODD编码的仿真实现。准备工具：步骤：（1）、按照Linux系统的虚拟机Ubantu，网上有许多详细教程，难度不大，这里就不进行详细讲解。（2）、 3 EVENODD算法详解EVENODD是基于RAID阵列的一种纠删码算法，由简单的异或操作产生两个冗余磁盘，可以容忍两个磁盘故障。这种冗余存储是最佳的，非常高效。EVENODD编码的小规模写操作也非常的高效，当一个磁盘扇区被修改时，只需要同时修改俩个额外的磁盘扇区。 在我们的假设中的一个明显的约束条件是，信息磁盘的数量M是一个素数。如果所需的磁盘数量不是素数，可以简单地假设有更多的磁盘，而不影响的编码和解码程序。假设有M + 2个磁盘，原始数据存储在第一个M磁盘中，而冗余数据存储在最后两个磁盘中。M，信息磁盘的数量，这里假设是一个素数，这个要求是很重要的，因为没有这个假设，该算法将失效。但是在实际情况中，M的素性不是很好保证，如果我们想存储任意数量的磁盘，这个数不一定是素数，到时可以采取这个任意数的下一个素数，并假设有没有信息的磁盘（即所有的信息位是0）。 EVENODD和针对两个磁盘故障恢复提出的其他方法想比较，有明显的优势：（1）、EVENODD编码基于简单的水平异或运算和对角线异或运算产生校验数据，不涉及递归运算。（2）、EVENODD编码只需要奇偶校验的硬件，通常是在标准的RAID-5控制器中就可实现，因此对其可实施的STA标准RAID-5没有任何硬件的变化。而基于RS代码的方案需要特殊的硬件，以支持有限域类型的计算，因此RS不能被纳入标准的RAID-5控制器。（3）、它可以被纳入到已知的RAID技术。例如

，奇偶校验 可以 分布在 所有 的磁盘，

避免瓶颈效应，当涉及重复写操作的时候（RAID-5）。（4）、操作符号可以有任意大小，从比特到多个扇区，没有限制的位或字节。（5）、在涉及多个磁盘的系统中，经常会遇到需要许多小规模写操作的情况。一个小规模写操作更新一个单一的磁盘扇区（一个符号）。EVENODD编码提供了极大的灵活性，因为所涉及的符号可以具有任意大小。通常情况下，我们将一个符号作为一个磁盘扇区。而且大多数小规模写操作只会影响两个冗余符号，也就是说，每一个写我们需要三读和三写操作。只有当受影响的符号是在对角线

（M -2，1）、（M- 3，2），...，（0，M - 1）

时，我们要修改列M + 1和列M上的一个符号。在任何情况下，所有的校验符号都是独立的。（6）、传统的已知方案中，采用的最佳冗余存储（两个额外的磁盘）是基于Reed Solomon（RS）纠错码，需要在有限域上计算并导致一个更复杂的实现。例如，我们发现在15个磁盘的磁盘阵列实现，EVENODD编码的复杂性是使用RS编码的50%。 3.1有限域的介绍有限域，即只

含有限多个元素的域。它首先由E.伽罗 瓦 所发现，因而又称为伽罗 瓦 域， 记为GF 。它和有理数域、实数域比较，有着许多不同的性质，

有限域的代表性应用是编码理论、开关理论、纠错码和AES。有限域的元素

个数称为F的阶，记为， 例如 GF(

2)。全体整数可以构成环，不够成域。最简单的有限域是整数环Z模一个素数p得到的余环Z/(p),由p个元素0,1,...,p-1组成,按模p相加和相乘。纠删码中的计算是基于某个有限域讨论的，比如素数域、二进制域，这俩种有限域上都包括了模加、模减、模乘、模平方、模逆和模的约减这几种运算。素域和二进制域被人们广泛采用，例如AES是1997年美国政府世界公开征集的密码算法，2000年称为美国国家标准。它利用扩成有限域，8次不可约多项式。一个字节就可视为一个多项式，成为中的一个元素。 EVENODD算法基于素数域，M为一个素数。在算法中，需要做有限域上的模运算，与实数域上的模运算有不一样的计算方式。一般情况下，模运算经常与取余运算混为一谈

，因为在大多数的编程语言里，都用“%”符号表示取模或者求余运算。

但是值得注意的是

在有负数存在的情况下，两者的结果是不一样的。

对于整型数a，b来说，取模运算或者求余运算的方法都是：1、求整数商：c = a / b；2、计算模或者余数：r = a - c * b。求模运算和求余运算在第一步不同:取余运算在取c的值时，向 0方向 舍入(fix()函数)；而取模运算在计算c的值时，向负无穷方向舍入(floor()函数)。例如：计算-7 Mod 4，那么：a = -7；b = 4； 第一步：求整数商c， 如果是 进行求模运算， 则 c = -2（向负无穷方向舍入）， 如果做 求

余运算，则c =

-1（向0方向舍入）；第二步：计算模和余数的公式相同，但因c的值不同，求模 运算 时r = 1，求余运算时 r = -3。 规律 ：当a和b符号一致时，求模运算和求余运算所得的c的值一致，因此结果一致，

比如11%3 = 2，-11%-3 = -2。但是当a和b

符号不一致时，结果不一样。求模运算结果的符号和b一致，求余运算结果的符号和a一致。

EVENODD编码算法中需要做有限域上的模运算，符号，则，并且，例如，。可得运算公式：a % b =（a + b）% b。 3.1 MDS阵列MDS（Maximum-Distance-Separable codes），即最大距离可分码。任何编码都由一组码字组成，两个码字间变化的二进制位数称为码距。而在一种编码中任意两个码字之间最少变化的二进制位数称为该数据编码的最小码距。 [[endnoteRef:12]] [12: []孙德文．计算机组成基础：机械工业出版社，2009] 这里先介绍一些纠删码中的基本概念：码重：码字的重量，即一个码字中“1”码的个数。通常用W表示。例如码字10011000的码重为W3，而码字00000000的码重为W0。同理10011110011100110111的码重为W13。码距

：码元距离， 即 两个码组中对应码位上码元不同的个数，也称汉明 距 。码距反映的是码组之间的差异程度，

比如00和 01两组码 的码距为1，011和100的码距

为3， 11000与10011之间的码距为3，码组10011001和11110101之间的码距为4。最小码距：码集（不同信息码元经差错编码后形成的多个码字组成的集合）中所有码字之间码距的

最小值即称为最小码距，用 dmin或者 d0表示。 例如，如果 码 集包含的 码

字有10010、00011和11000，则各码字两两之间的码距分别如下：10010和00011之间码距为2，10010和11000之间码距为2，00011和11000之间距离为4，因此该码集的最小码距为2，即dmin=2。000、001、110三个码

组相比较，码距有1和 2两个 值，则

dmin=1

。最小码距是码的一个重要参数，它是衡量码检错、纠错能力的依据。

码距与纠错能力的关系： 数字通信系统中送入信道的信息都是“0”、“1”组合的数字信号，例如待传送的信息是“晴”和“雨”，则只需一位数字编码就可以表示。若用“1”表示“晴

”，“0”表示“雨”。当“0”、“1”形式的信息在信道中传输时，将0错成1或将1错成0时，由于发生差错后的信息编码状态是发送端可能出现的状态，因此接收端无法发现差错。但是如果发送信息送进信道之前，在每个编码之后附加一位冗余码，变成用两位编码“11”表示“晴”，“00”表示“雨”，则在传输过程中由于干扰造成信息编码中一位码发生差错，错成“10”或“01”时，由于“10”或“01”都是发送端不可能出现的编码，接收端就能发现差错，但此时并不能判断出差错是第一比特还是第二比特，因此不能自动纠错。若继续增加冗余码位数用“111”表示“晴”，“000”表示“雨”，当编码在传输中出现1位或2位码差错，如错成“001”或“101”等编码时，接收端都能检测到并能确定只有1位码差错时错误码位的位置，此时这种编码方式可以检测1位或2位差错，并能纠正单个的误码。这主要是因为冗余码位数增加后，发送端使用的码集中码字之间

最小码距 增大。由于 最小码距 反映了 码

集中每两个码字之间的差别程度，如果最小码距越大，从一个编码错成另一个编码的可能性越小，则其检错、纠错能力也就越强。因此最小码距是衡量差错控制编码纠、检错能力大小的标志。（M - 1）x（M + 2）数组定义上可以恢复在任何两列中丢失的信息。换句话说，代码的最小距离为3，在这个意义上，在代码中的任何非零数组至少有3个列，这些列是非零 的。这意味着，该代码的最小距离是3，因为，如果我们编码一个数组只有一个非零的信息列，所产生的编码阵列将有（列）的重量正是3。一个特殊的对角线进行偶数或奇数奇偶校验的条件不是任意的。我们将在例子中看到，如果没有这个假设，生成的代码没有最小的距离3，因此它不能检索被删除的两个列。它将被执行当一个磁盘失败，或当两个磁盘同时失败。然后我们证明效果的算法校正两擦除。我们还想指出的是，如果我们不做假设的对角线进行奇偶校验，代码没有最小距离3（编码理论术语，代码不是最大距离可分或MDS [ 12 ]）一个完整的证明版本可以在文献[27]中找到[1]

F. J. MacWilliams and N. J. A. Sloane, The Theory of Error-Correcting Codes. Amsterdam, The Netherlands: North-Holland, 1977. 3.

3 EVENODD算法3.3.1编码算法为了简化演绎算法，这里假设每个磁盘只有M - 1个符号的信息。程序的工作原理与任意容量的磁盘，分别处理每一个块上的M - 1个符号。为了简单明了，在以下的一些例子中，假设每个符号的大小都是一个位。但是在一些实际应用中，一个符号可能像一个512字节的磁盘扇区一样大。 这里假设共有M+2张磁盘，其中M张磁盘存储原始数据信息，最后的两张磁盘存储奇偶校验信息。在(M-1)×(M+2)磁盘阵列中，表示第j张盘上的第i个码元，其中，。在下面的译码过程中，会给矩阵加上0行，即在矩阵的最后一行加上全为0的一行，表示为，这个假设不是必要的，但它是有用的标记，便于计算。这时阵列表示为M×（M+2）在(M-1)×M磁盘阵列中，通过行校验和对角线校验产生两列冗余数据充当校验集来保证任意两个磁盘同时故障时（位置已知）数据不会丢失，或者纠错一列出错的情况。水平奇偶校验相关的符号集如下所示:？？？？？？？？？？？□ □ □ □ □ □？？？？?表3.1水平校验相关符号对角线奇偶校验相关的符号的集合如下：（请注意，∞是特殊的对角线，对角线奇偶校验产生S符号，S为偶数表示偶校验，S为奇数表示奇校验）？？□？∞？□？∞？？□？∞？？□∞？？□？表3.2对角线校验相关符号 编码过程： (3-1) 当0≦l≦m-2时 (3-2) (3-3) 例：当m=5时，按照公式，如图方式进行编码，D0~D4是数据盘，D5和D6为冗余数据，充当校验盘。图3.1编码过程图3.3.2任意两个磁盘失效修复算法修复失效磁盘的方法即译码算法，任意两个磁盘失效，需要知道具体错误位置。假设存在故障的是Di和Dj列，则0 ≦ i < j ≦ m+1，这里有4种情况，可以分别进行讨论。 1. i = m , j = m + 1 两个冗余磁盘已经失效，重建就相当于编码的情况。使用上述（3-1）、（3-2）、（3-3）公式即可恢复 2.i < m,j=m 即一个冗余磁盘和一个数据磁盘已经失效，此处假设D1和D5出现故障。 译码过程： 1、先为所有磁盘增加一个0的数据，即令，其中0 ≦ l ≦ m+1，再通过公式（3-4） 计算奇偶符号S：（3-4） 可得： 2、通过公式（3-5）计算D1 （3-5） 如图可得D1： 图3.2译码D1列过程图 3、最后可用公式（3-2）得到D5： 图3.3译码D5列过程图 3. i < m , j = m+1 一个冗余磁盘和一个数据磁盘已经失效，此处假设D1和D6出现故障。 译码过程：先通过水平奇偶公式（3-2）计算D1，通过公式（3-1）和（3-3）计算出 D6 4.i < m,j < m 两个数据磁盘已经失效，这是主要的情况，这里假设D0和D2出现了故障译码过程： 1、先为所有磁盘增加一个0的数据，即，其中0 ≦ l ≦ m+1，我们首先需要一个入口，在那里我们可以开始。例如，对角线（3，1），（2，2），（1，3），（0，4）相交在条目的列2（2，2）只：这是特殊的对角线，它有奇数奇偶性。由于在这个对角线上的唯一的

位丢失位（2，2），通过检索它使用其他位再通过公式（3-6）计算奇偶符号S： （3-6） 可得 S=1： 图3.4求符号S图 2、再通过公式（3-7）计算： （3-7） 可得： 又可得： 最终可得： 3、再通过公式（3-8）计算 （3-8） 可得： 又可得： 最终可得： 4、再通过以下步骤计算： （1）令S ← <-(j-i)-1>m， （2）令 （3）令S ← <S-(j-i)-1>m，若S = m-1停止，否则返回（2） 图 3.5计算步骤图 可得： 图3.6译码结果图 3.3.3单个磁盘出错修复过程 当前的算法可以纠算出一个磁盘出现错误的情况，并且进行修改。解码仍然分为四种情况。首先通过公式（3-9）和（3-10）计算水平综合征和对角线综合征： （3-9） 易知： （3-10） 这里也分4种情况讨论： 1、 这种情况说明磁盘阵列没有数据错误 2、 这种情况说明磁盘阵列的m列出现错误，只需要用公式（3-2）就可以恢复m列 的数据 3、 这种情况说明磁盘阵列的m+1列出现错误，只需要用公式（3-1）和（3-3）就可 以恢复m+1列的数据 4、 这种情况下说明一个数据盘j (0≦j≦m-1)出现错误，需要计算出确定的位置。 （1）、首先为各个盘虚拟添加一行数据0，然后通过公式（3-9），（3-10）得 出： （2）、然后对向量里的数据右移直到，（其中j是右移的次数），经计算， j=2，即D2出现错误，此时只需要的前m-1个数据与D2的数据逐一相加即可恢复。 4 EVENODD基于Hadoop平台的实现 4.1 HDFS-RAID实现过程 4.2 HDFS-EVENODD设计实现 4.2.1在WINDOWS系统下的实现 1、实现单个的矩阵丢失任意两列的恢复 其中实现矩阵水平校验的核心代码：将数组的每行异或存储为额外的一列 public static byte[] horiExclusive_OR(byte[][] dataCache) { // TODO Auto-generated method stub int l=M-1; byte[] horiExculsive=new byte[l];

for(int i=0;i < l ;i++){ byte temp= 0; for(int j=0;j < M;j++) temp=( byte) (temp^ dataCache [i][ j]); horiExculsive [i]= temp; } return

horiExculsive; } 实现矩阵对角线校验的核心代码：将数据的对角线的数据异或然后存储为额外的一列 public static byte[] diagExclusive_OR(byte[][] dataCache, byte commonFactor){ // TODO Auto-generated method stub int l=M-1; byte[] diagExclusive = new byte[l];

for(int i=0;i < l ;i++){ byte temp=commonFactor; for(int j=0; j < M;j++){ int t= getMod ((i-j),M); if (t > =

l) continue; temp=(byte) (temp^dataCache[t][j]); } diagExclusive[i]=temp; } return diagExclusive; }

----------我是华丽丽的分割线---------- 图

4.1水平校验算法代码图 图4.2对角线校验算法代码图

----------我是华丽丽的分割线---------- 图

4.3编码结果图 这里假设错的是D0和D5 图4.4译码结果图 2、图片碎块丢失后恢复原图 其中实现文件分块的核心代码：用java自带的文件流读写文件，一次读取 DISK_SIZE(一个文件块的大小)大小的字节，放入ReadBuffer，并写入文件流，为文件 碎块编号 public static void split() throws IOException { int blockNo=0; String filePath="./1.jpg"; byte[] ReadBuffer=new byte[(int) DISK_SIZE];//一次性读取DISK_SIZE大小字节 //以二进制打开文件 DataInputStream fpr=new DataInputStream(new BufferedInputStream(new FileInputStream(filePath))); try{ while(true){ DataOutputStream fpw=new DataOutputStream(new FileOutputStream("./2-"+blockNo+".jpg")); fpr.read(ReadBuffer); fpw.write(ReadBuffer); fpw.close(); blockNo++; if(blockNo > =M) break; } }catch(EOFException e){ fpr.close(); } } 实现文件碎块合并的核心代码：同样用java自带的文件流读写文件，遍历读取文件碎块，依次写入文件流，最后一块特殊化处理，将做运算补齐的字节去掉后再读入文件流，最后合并成的文件就是原文件public static void merge() throws FileNotFoundException,IOException { String filePath="./3.jpg"; int blockNo=0; byte[] WriteBuffer=new byte[(int) DISK_SIZE]; int differ=(int) (DISK_SIZE * M - length); //去除补充的字节数 byte[] WriteBufferDiffer=new byte[(int)(DISK_SIZE-differ)]; DataOutputStream fpw=new DataOutputStream(new FileOutputStream(filePath)); try{

for(int k=0;k < M-1;k++){

DataInputStream fpr=new DataInputStream(new BufferedInputStream(new FileInputStream("./2-"+blockNo+".jpg"))); fpr.read(WriteBuffer); fpw.write(WriteBuffer); fpr.close(); blockNo++; } //最后一块特殊对待 DataInputStream fpr=new DataInputStream(new BufferedInputStream(new FileInputStream("./2-"+blockNo+".jpg"))); fpr.read(WriteBufferDiffer); fpw.write(WriteBufferDiffer); fpr.close(); blockNo++; }catch(EOFException e){ fpw.close(); } }



? ? ? ?