两种纠删码在分布式文件存储系统中的应用

课程名：海量存储

姓名：刘佶豪

学号：2017286190094

院系：遥感测绘国家重点实验室

# 摘 要

分布式存储系统越来越依赖于使用纠删码，因为它们提供的可靠性比数据复制的存储成本要低得多。然而，这些代码在需要多个磁盘读取和网络传输以重构不可用的数据块时，需要权衡恢复性能。因此，大多数现有系统都使用一种消除代码来优化存储开销或恢复性能。本文基于麦基尔大学Xia Mingyuan教授等人的论文《A Tale of Two Erasure Codes in HDFS》，介绍了HACFS，它是一个新的erasUNK编码的存储系统，它使用两个不同的纠删码，并动态地适应工作负载的变化。它使用一种快速的代码来优化恢复性能和紧凑的代码，以减少存储开销。一种新的转换机制被用来有效地提高代码和代码之间的数据块之间的快速和压缩的代码。在该论文中研究者证明了HACFS设计技术是通用的，并成功地将其应用于两个不同的代码家族:产品和LRC代码。

HACFS系统其实可以理解为Hadoop分布式存储系统HDFS的一种拓展，论文从五种不同的工作负载从支持duction集群中评估它。与三种流行的单代码存储系统相比，HACFS系统总是保持低的存储开销，并大大提高了重新编码的性能。它降低了被降级的读取率，高达46%，重建时间和磁盘/网络流量增加了45%。

1. **概述**

随着数据的存储呈现出集中化（以分布式存储系统为基础的云存储系统）和移动化（互联网移动终端）的趋势，数据可靠性愈发引起大家的重视。集群所承载的数据量大大上升，但存储介质本身的可靠性进步却很小，这要求我们必须以更加经济有效的方式来保障数据安全。

系统必须容忍由不可靠组件、软件故障、机器重新引导和维护操作引起的不同故障，使用删除代码代替重新编码数据的权衡是性能。如果一个数据块是三路复制，它可以通过复制它的一个可用副本来重建。然而，对于一个已编码的系统，重构一个不可用的块，在代码条内获取多个数据和奇偶校验块，这导致磁盘和网络流量显著增加。最近在一本关于人脸的数据仓库集群上的测量[19,20]存储了大量的erasm编码数据的pb级，要求数据传输量中位数超过180兆兆字节的数据，以从每天50个机器不可用事件中恢复过来。

在恢复过程中，读取和传输数据量的增加会导致两个主要问题:高退化读取延迟和更长的重建时间。首先，读到不可用块需要多个磁盘读取、网络传输和计算周期来解码块。访问块的应用程序等待整个恢复过程的整个过程，从而导致更高的la- tencies和降级的读取性能。第二，一个失败的或退役的机器，或一个失败的磁盘的结果，比在一个复制的系统中，重建的时间要长得多。虽然，从一个失败的磁盘或机器上丢失的数据的恢复可以在后台执行，这严重影响了系统的总吞吐量，以及在重建阶段的退化读取的延迟。

结果,减少开销的问题恢复erasure-coded系统收到了明显的关注最近在理论上和美容,大多数的解决方案之间的权衡两个维度:存储开销和恢复成本。存储开销帐户用于编码方案的额外奇偶校验块。恢复成本是在失败后重建数据块所需的块总数。

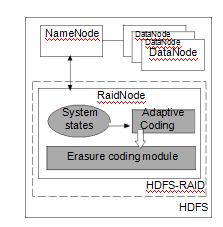
一般来说，大多数生产系统使用的是单一的era - sure代码，这些代码可以优化恢复成本或存储开销。例如，reed - solomon是一种流行的代码家族，用于谷歌的Colos- susFS,Facebook的HDFS，以及其他一些存储系统。在ColossusFS中使用的reed - solomon代码有一个1.5 x的存储开销，而它需要六个磁盘读取和网络传输来恢复丢失的数据块。相比之下，HDFS中使用的reed - solomon代码将存储开销降低到1.4 x，但恢复成本为10个块。另一个流行的代码是本地重建代码(LRC)，并有类似的权衡。

在该论文中，研究者介绍了Hadoop adaptively编码的分布式文件(HACFS)，这是一种新的基于内存的存储系统，它使用的是来自同一代码家族的两种不同的删除代码。它使用一种快速的代码，低恢复成本和低存储开销的紧凑代码。它利用Hadoop工作负载中观察到的数据访问倾斜[9,7,22,5]来决定数据块的初始编码。HACFS系统使用快速编码来编码一小部分频繁处理的数据，并为系统提供整体的低恢复成本。它使用紧凑的代码来编码不经常访问的数据块的主要特性，并保持低和有界的存储开销。该论文的主要贡献如下：

1. 作者设计了HACFS，它是一种新的基于内存的存储系统，它通过使用两个不同的擦除码来适应工作负载的变化——这是一种快速的代码，用来优化已降级的读取和重建失败的磁盘/节点的恢复成本，以及一个紧凑的代码来提供低的和有界的存储开销。
2. 在HACFS上设计了一种新的转换机制，有效地提高了两个代码之间的代码数据块。转换机制是通用的，我们实现了两个代码families - Product和LRC代码——在分布式存储系统中广泛使用。
3. 我们将HACFS作为对HDFS的扩展，并通过对产品和LRC系列代码的两个案例研究来证明其有效性。我们将HACFS部署在一个具有实际工作负载的集群上，并将其与生产中使用的三种常用的sin- gle代码系统进行比较。HACFS系统总是保持低的存储开销，同时将退化的读延时提高到25% - 46%，重构时间为14 - 44%，网络和磁盘流量在重建过程中减少了19 - 45%。
4. **系统设计**

2.1在HACFS系统中的适应性编码

HACFS系统被实现为HDFS中HDFS raid模块[3]的扩展。我们将扩展到hdfs - raid作为图5中的三个阴影组件。自适应编码模块维护了基于erasi的数据的系统状态，并管理用于摄入和存储数据的状态转换。它还与erasure编码模块接口，后者实现了不同的编码方案。



HACFS结构

系统状态：HACFS的自适应编码模块管理系统状态。系统状态跟踪与每个erasdata编码的数据文件关联的以下文件状态:文件大小、最后修改时间、读取计数和编码状态。文件大小和最后的修改时间是由HDFS维护的属性，由HACFS使用来计算文件的总数据存储和写入年龄。自适应编码模块还跟踪一个文件的读取计数，这是HDFS客户端读取文件的总数。文件的编码状态代表了它是三路复制或删除编码方案。文件状态可以更新为从HDFS客户端向文件发出的创建、读或写操作。

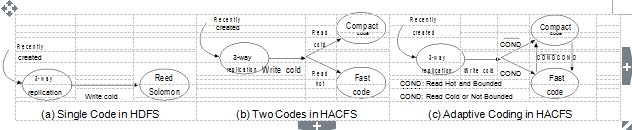
自适应编码模块还维护了一个全局状态，即用于数据和语法的总存储。在HDFS集群中，复制数据文件中的每个块都被复制到三个不同的节点上，两个副本用于奇偶存储。

相反，在纠删码数据文件中的每个块都只有一个副本。每个erasUNK编码的文件被分成不同的纠删码条纹，在HDFS集群中分布的每个条纹中都有块。每个eras定编码的数据文件也有一个相关的奇偶校验文件，其大小由编码方案决定。当自适应编码模块启动对eras编码文件的状态转换时，系统的全局状态会周期性地更新。状态转换对应于一个文件的编码状态的变化，并通过使用以下面向erasUNK编码模块的交互面来调用。

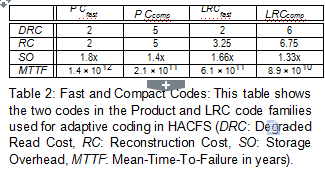
编码接口。如表1所示，HACFS系统的删除编码模块导出了编码数据的四个主要的in - terfaces:编码、解码、upcode和downcode。编码操作需要一个数据文件和编码方案作为输入，并为数据文件中的所有块生成一个奇偶校验文件。解码操作是在一个被降级的读取块失败或作为一个磁盘或节点失败的重建作业的一部分。它还需要一个文件中缺失或被损坏的块的索引，并使用输入编码方案在条纹中的剩余数据和奇偶校验块中重新构造丢失的块。

状态转换。HACFS系统扩展了hdfs - raid，以使用不同的读取频率来使用不同的文件删除编码方案，从而实现了快速代码的低恢复成本和紧凑代码的低存储开销。我们首先描述在hdfs raid中使用的基本状态机，然后详细介绍HACFS扩展。如图6(a)所示，最近在hdfs - raid中创建的文件被归类为基于最后修改时间的写入hot，因此是三路复制。hdfs - raid进程(如图5中的RaidNode)定期扫描文件系统以选择写冷文件来删除代码。然后，它安排好几个mapre -导出作业，用reed - solomon代码对所有这些候选文件进行编码[3]。在编码后，这些文件的复制级别降低为1，编码状态更改为reed - solomon。由于HDFS只支持对文件的ap - pends，所以一个块永远不会被覆盖，而且这些文件只在被删除后才被读取。

图6(b)显示了HACFS系统的第一个扩展。它复制了类似于hdfs raid的热文件。此外，HACFS还为文件中的数据块读取访问。所有的写冷文件都基于他们的阅读计数，并编码成两种不同的删除码。读取高读取计数的热文件用快速代码进行编码，这具有较低的恢复成本。使用低读计数的冷文件被编码成紧凑的代码，它的存储开销很低。



downcode操作将文件从com - pact转换为快速代码。其结果是，它降低了未来被降级读取到文件的恢复成本，这是一种非常紧凑的代码，但最近才被访问。如图4所示，Hadoop工作负载的数据访问倾斜导致一小部分读取热文件和大量读取冷文件。这种倾斜允许HACFS通过编码/将读取的热文件编码为快速的代码，并通过编码/ up编码来降低存储开销，从而降低存储开销。



2.2产品编码

这一部分描述的产品的编码接口的产品编码在HACFS中的应用。

编码和解码：表七显示了一个有条纹的两行五列的数据块的重构，pcf ast的编码操作从HDFS集群中的不同位置检索10个数据块，生成两个水平的、五个垂直的和一个全局奇偶校验。在每一行中传输五个数据块并在其上执行XOR操作，从而生成水平的平价。垂直奇偶校验只在列中重新调整两个数据块传输。全球奇偶性可以构建为两个水平的位元的XOR。一个产品代码的解码操作是在一个块失败上调用的。在P Cf ast代码的任何数据或奇偶性块中，一个单一的故障只需要从同一列传输两个块来重建它。结果,P Cf ast代码可以实现复苏两块转移成本很低的成本高的存储开销为了八个奇偶校验块十个数据块,选择P Ccomp或P C(6×5)紧凑的代码(参见图7),它提供了一种降低存储开销成本高1.4 x和复苏的五块转移(见表2)。此外,快速和紧凑的产品代码的可靠性比三方复制。我们为HACFS系统选择了一个1.5 x的存储绑定，因为它接近于pccomp代码的存储开销。这个绑定还与谷歌代码(0)所规定的实际限制相匹配[2]，它使用reed - solomon RS(6,3)代码，类似地为恢复成本进行了优化。

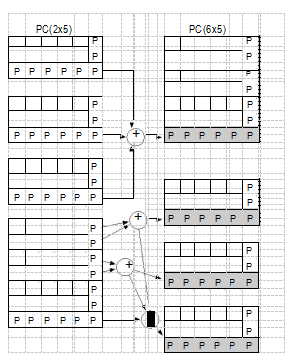


图7:产品代码- Upcode和Downcode op - erations:图中显示了代码的Upcode和Downcode操作，以及产品代码的奇偶校验块。在输出代码中，阴影的水平奇偶校验块只计算，其余的块与输入代码保持不变。

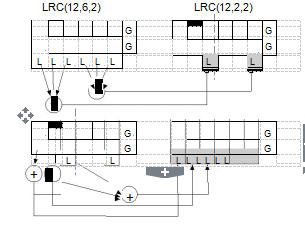
Upcoding Downcoding。图7从P Cf ast到P Ccomp，从P Ccomp向下编码到P Cf ast编码。Upcode是一种非常高效的产品代码转换操作。从P Cf ast到P Ccomp代码，Alldata和垂直奇偶校验块保持不变。此外，它只执行XOR在三个P Cf ast代码的旧水平和全局奇偶校验块上，以计算P Ccomp代码的新水平和全局奇偶校验块。因此，upcode操作不需要从三个代码到PC f ast的数据块的任何网络传输，都可以在P Ccomp代码中计算新的参数

downcode操作将pccomp代码转换为三个P Cf ast代码。只有水平和全局的参数在P Ccomp代码和三个P Cf ast代码之间变化。然而，在前两个P Cf ast代码中，计算水平和全局参数需要在pccomp代码的两个水平行的数据块上进行net - work传输和XOR操作。在第三个P Cf ast代码中，水平和全局的参数是由旧的P Ccomp代码和前两个P Cf ast代码的那些新计算出来的。这种优化节省了两个水平行数据块的网络传输。类似于up - code操作，数据和垂直奇偶校验块在重新sulting 3 P Cf ast代码保持不变的P Ccomp代码，不需要任何网络传输。

2.3.LRC编码  
编码和解码：图8显示了LRCf ast或LRC(12、6、2)的结构，其中有12个数据块，6个本地参数，以及两个全局参数。LRC代码的编码操作通过执行XOR在一组数据块上进行计算。在LRCf ast中，每个列中的两个数据块组成了一个不同的组，两个全局参数通过执行一个reed - solomon编码来进行计算[24]。全球范围的reed - solomon编码具有类似于Microsoft Azure Storage[15]中使用的LRC代码构造的属性，用于最突出的单块故障场景。LRCf ast代码的解码操作类似于数据和本地奇偶校验块的产品代码。对于LRCf ast的数据或本地奇偶校验块中的任何单个故障都需要从同一列进行两个块传输来重建。然而，在一个全局奇偶性块中失败需要所有12个数据块来使用reed - solomon decoding来重建它。然而，LRCf ast代码的存储头是1.66x，对应于12个数据块所需的8个奇偶块。

我们使用在Azure[15]中使用的LRCcomp或LRC(12、2、2)代码作为自适应编码在HACFS中的代码。LRCcomp代码的存储空间较低，因为本地的不太好。然而，它的两个本地参数中的每一个都与一个由六个数据块组成的组相关联。因此，恢复丢失的数据块或局部奇偶性需要在LRCcomp代码中从其组传输六个块。全球范围需要12个数据块传输来恢复。因此，LRCcomp代码的恢复成本也较低，其重构成本与LRCf ast代码相似。两个LRC代码都比三种方法更可靠。我们为HACFS系统选择了一个带有LRC代码的1.4 x的存储绑定，因为它接近LRCcomp代码的存储开销，并且低于HACFS和产品代码。

图8通过计算LRCcomp代码中的新局部参数，并从LRCf ast代码中保存全局参数，显示了LRC的upcode操作。LRCcomp代码中的两个本地参数在LRCf ast代码中作为一个XOR超过三个局部的参数。因此，HACFS系统只需要6个网络传输来计算两个新的本地参数。在upcode操作中使用LRCcomp代码。downcode操作通过LRCcomp代码的各个列中的数据块来计算LRCf ast代码中的三个新的本地参数中的两个。第三个本地奇偶校验是通过执行一个XOR在两个新的本地参数和LRCcomp代码中旧的本地奇偶校验值来计算的。总的来说，下行代码的操作需要十个块的传输来计算新的本地参数。全局参数保持不变，也不需要在downcode操作中进行任何网络工作传输。



图八LRC代码- Upcode和Downcode op - erations:图中显示了为LRC代码提供数据和奇偶校验块的Upcode和Downcode操作。阴影块仅在这些操作期间计算，其余块保持不变。

1. **实现**

我们已经将HACFS作为扩展到Hadoop分布式文件系统(HDFS)中的HDFS raid[3]模块。在HDFS集群中，HDFS - raid模块受到了Facebook的支持，以支持分布式存储的单一删除代码。我们的HACFS的实现跨越了近2k行代码，包含在HDFS raid模块中，并且不需要对其他HDFS组件(如namen - ode或DataNode)进行修改。

纠删码在HDFS：HDFS raid模块覆盖了在HDFS上的擦除编码，并作为RaidNode进程运行。RaidNode进程周期性地查询需要编码的新数据文件的NameNode，以及需要重新覆盖的损坏文件。RaidNode启动一个MapReduce作业来计算与用于编码操作的dif -ferent DataNodes数据文件相关的奇偶校验文件。解码操作被调用作为降级读取或重新构建阶段的一部分。

从HDFS客户端读取的数据要求从DataNode中使用block con-帐篷。被降级的读取可能发生在DataNodes(比如CRC检查错误)上的失败。在这些情况下，HDFS客户端查询RaidNode，以获得恢复所需的erasin -code stripe中可用块的位置。然后客户端检索这些块并执行解码，以恢复失败的块。已恢复的块用于服务读取请求，但它不会被写入HDFS，因为大多数退化的读取都是由瞬时故障引起的，这些故障并不一定表示数据丢失。

HACFS和两个删除代码。上文展示了HACFS具体实现的三个主要组件纠删编码模块、系统状态和自适应编码模块。此外，我们还实现了一个故障注入器来触发降级读取和数据重建。

HDFS raid只支持单个的reed - solomon erasure代码来编码数据。作为HACFS删除编码模块的一部分，我们实现了两个新的代码系列:产品和LRC代码。HACFS中的删除编码模块输出与HDFS raid相同的编码/解码。此外，删除编码模块还为在HACFS自适应编码模块中实现的扩展状态机提供了两个新的upcode / downcode交互面。upcode操作要么合并三种快速编码，要么将LRC代码的快速代码分解成更小的新协议代码。下行编码执行重复的步骤序列。这两个操作都更改数据文件的编码状态，并将其复制级别降低到一.

我们还在HDFS外实现了一个故障注入器，以模拟DataNodes上不同的块、磁盘和节点故障模式。故障注入器从DataNode上的本地文件系统中删除一个块，它被HDFS DataNode检测为一个丢失的块，当一个HDFS客户端试图访问它时触发一个降级读取。错误注入器通过删除目标DataNode给定磁盘上的所有数据，然后重新启动相应的DataNode进程来模拟磁盘故障。通过杀死目标DataNode进程本身来注入节点故障。在磁盘和节点失败中，NameNode更新丢失块的列表，然后RaidNode启动MapReduce作业进行重构。

1. **评价**

从三个不同的方面去评价HACFS的设计技术，降级读取的延迟，重建时间，过量存储。

4.1降级读取延迟

HACFS采用了一种可恢复性快速编码(P Cf ast和LRCf ast)和存储效率的com- pact代码(P Ccomp和LRCcomp)的组合。图9显示了在y轴上被降级的读取延迟，以及在x轴上存储的5个不同工作负载的存储过度。从HDFS客户端到可用数据块读取的normal - mal可以占用1.2秒，因为如果该块重新设置，则需要一个localdisk读取和一个网络传输。相比之下，降级的读取可能需要多个网络传输，并且在3个编码系统中需要16 - 21秒。这些系统不适应工作负载，只使用一个代码。因此，它们的降级读取延迟和存储过头在所有5个工作负载中是相同的。自适应编码HACFS减少退化读了5 - 10秒的延迟三个工作负载(CC1,CC4和facebook),有更高比例的热数据读取en-coded快速代码(85 - 90%,见表3)。图9中的两个阴影框表明HACFS适应不同工作负载的特点。然而，HACFS总是优于三个单一编码的系统，因为它们都需要更多的块来读取和传输到网络来解码一个错误的块。

HACFS系统在工作负载(CC1、CC2和FB)上的存储开销较低，而使用com- pact代码的冷文件(85 - 95%)的百分比更高。HACFS的最低存储开销是由两个阴影区域的左边界显示的，它们分别以1.33x和1.4 x作为紧凑代码(分别是LRCcomp和P Ccomp代码)。此外，HACFS还将产品代码的存储开销和LRC代码的1.4倍限制在了存储开销上。因此，工作负载(CC3和CC4)很少有冷文件，它们仍然不能超过在两个阴影区域的右边缘标记的这些存储开销。如果我们不强制任何存储开销界限，这两种工作负载在降级读延迟中减少了6 - 20%，就会进一步受益。

4.1重构时间

图10显示了三个single代码系统和HACFS系统的重构时间，当一个带有100gb数据的磁盘失败时，它具有自适应的编码。recon- struction作业在不同的DataNodes上启动映射任务，以重新创建来自丢失磁盘的数据和奇偶校验块。重构一个编码为com- pact代码的冷文件的时间比快速代码的时间要长。带有产品代码的HACFS系统超过了所有5个工作负载的三个单一的代码系统。它比三个正弦信号系统的重建时间要少10 - 35分钟。这是因为fast和com- pact产品代码的重构速度快于两个reed - solomon代码和LRCcomp代码。

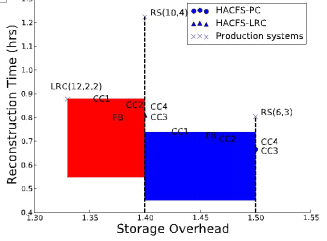
****

图10:重建时间:图中显示重建时间，从两个HACFS系统和三个单一的代码系统中恢复数据丢失

使用更快的LRCf ast代码进行重构的HACFS系统执行的LRCcomp代码具有最低的存储开销。然而，使用LRC代码的HACFS系统在所有工作负载中通常比在疣体中使用的RS(6,3)代码更糟糕。这是因为在HACFS中使用的两个LRC代码都有一个比RS(6,3)代码更高的恢复成本。

带有产品代码的HACFS系统总是比3个sin- gle代码系统更少的重构块:CC3和CC4工作负载的8个块和CC1、CC2和FB工作负载(包含超过85%的冷文件)之间的8个块，基于数据倾斜分布。与产品代码相比，带有LRC代码的HACFS系统需要更多的块进行全局奇偶恢复。结果，它的重建流量接近于RS(6,3)和LRCcomp代码，每丢失数据块有10个块。

1. **结论**

我认为该论文的主要贡献在于设计并实现了一个新的erasUNK编码的存储系统，通过使用几种不同的纠删码，能很好地动态适应工作负载。详细地介绍了系统三个不同的重要组件纠删编码模块，系统状态，自适应编码模块。通过对比这种新型系统相对于原来的单编码系统在过量存储，降级读取延迟，重构时间上都有很大的进步，为我们利用纠删码解决分布式系统存储的容错问题提供了一种新的思路。