计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20170834
56(4):767-778, 2019

纠删码存储系统中基于网络计算的高效故障重建方法

唐英杰　王　芳　谢燕文
(武汉光电国家研究中心(华中科技大学)　武汉　430074)
(信息存储系统教育部重点实验室(华中科技大学)　武汉　430074)
(深圳华中科技大学研究院　广东深圳　518000)
(tangyingjie@mail.hust.edu.cn)

An Efficient Failure Reconstruction Based on In-Network Computing for Erasure-Coded Storage Systems

Tang Yingjie, Wang Fang, and Xie Yanwen
(*Wuhan National Laboratory for Optoelectronics* (*Huazhong University of Science and Technology*), *Wuhan* 430074)
(*Key Laboratory of Information Storage System* (*Huazhong University of Science and Technology*), *Ministry of Education*, *Wuhan* 430074)
(*Shenzhen Huazhong University of Science and Technology Research Institute*, *Shenzhen*, *Guangdong* 518000)

Abstract　Nowadays, the scale of distributed storage systems is getting increasingly larger. No matter whether the storage devices are disks or solid-state drives, the system is always faced with the risk of data loss. Traditional storage systems maintain three copies of each data block to ensure high reliability. Today, a number of distributed storage systems are increasingly shifting to the use of erasure codes because they can offer higher reliability and lower storage overhead. The erasure codes, however, have an obvious shortcoming in the reconstruction of an unavailable block, because they need to read multiple disks, which results in a large amount of network traffic and disk operations and ultimately high recovery overhead. In this paper, INP (in-network pipeline), an effective failure reconstruction scheme based on in-network computing that utilizes SDN (software defined networking) technology is presented in order to reduce the overhead of recovery without sacrificing any other performance. We use the global topology information for network from SDN controller to establish the tree of reconstruction, and transmit data according to it. The switches do part of the calculation that can reduce the network traffic, therefore to eliminate the bottleneck of the network, and to enhance the recovery performance. We evaluate the recovery efficiency of INP in different network bandwidths. Compared with the common erasure code system, it greatly reduces the network traffic and in a certain bandwidth, the degraded read time is the same as that of normal reading.

Key words　distributed storage system; erasure code; software defined networking (SDN); recovery overhead; in-network computing

收稿日期:2017-11-03;**修回日期**:2018-03-05

基金项目:国家自然科学基金项目(61772216);武汉应用基础研究计划项目(2017010201010103);深圳市科技计划项目(JCYJ20170307172248636);中央高校基本科研业务费专项资金;国防预研项目(31511010202)
This work was supported by the National Natural Science Foundation of China (61772216), the Wuhan Application Basic Research Project (2017010201010103), the Fund from Science, Technology and Innovation Commission of Shenzhen Municipality (JCYJ20170307172248636), the Fundamental Research Funds for the Central Universities, and the National Defense Preliminary Research Project (31511010202).

通信作者:王芳(wangfang@mail.hust.edu.cn)

摘 要 目前分布式存储系统的规模越来越大,不论存储设备是磁盘还是固态盘,系统都始终面临着数据丢失的风险.传统分布式存储系统大多采用基于三副本的高可靠性技术,但为了追求较低的存储开销,大量系统正在转向基于纠删码的可靠性方法.但是在纠删码方案下,重建故障数据需要读取多个存储设备,这将导致大量的网络传输和存储 I/O 操作,增大系统恢复开销.为了能够在不损失其他性能的同时降低恢复开销,利用软件定义网络(software defined networking, SDN)技术,提出一种基于网络计算的高效故障重建方案——网络流水线(in-network pipeline, INP),其中 SDN 控制器利用网络的全局拓扑信息构造重建树,系统依据重建树进行数据传输,并在交换机上完成部分计算,减少向后传输的网络流量,从而消除网络瓶颈,提升恢复性能.测试评估了不同网络带宽下 INP 的恢复效率.实验结果表明:与传统的纠删码系统相比,INP 总是能大幅减少网络流量,并且在一定带宽条件下,能够接近正常读的时间开销.

关键词 分布式存储系统;纠删码;软件定义网络;恢复开销;网络计算

中图法分类号 TP309.3

随着大数据时代的来临,数据的爆炸式增长使得存储系统的规模不断增加[1-3],但是存储设备的可靠性却一直没有得到显著提高.Google 的研究发现[4],传统机械磁盘的年更换率为 2%~9%,而固态盘 (solid-state drive, SSD)在 4 年内的整盘更换率为 4%~10%,比磁盘具有更低的更换率,但坏消息是 SSD 在数据局部损坏方面远远高于磁盘.这些问题给数据的持久化存储带来了巨大的挑战.

传统的分布式存储系统,比如 GFS(Google file system)[5]和 HDFS(Hadoop distributed file system)[1]采用多副本策略来保证系统可靠性.其中三副本策略可以容忍任意 2 个副本数据的丢失,在数据恢复过程中只需要从剩余可用副本拷贝数据即可,并且在并发访问中,3 个副本可以同时提供服务、分担负载.但是随着存储规模越来越大,对每份数据维护 3 个副本不仅在存储开销上代价高昂,而且大大增加系统管理员的负担.因此大量的数据中心开始部署纠删码[2,6-10],以较小的存储开销来保证足够高的系统可靠性.

目前制约纠删码系统的主要因素是恢复开销问题.纠删码的冗余策略是将文件分成若干数据块,然后通过计算产生指定数量的校验块,数据块和校验块共同形成了一个称为条带的结构.纠删码系统的存储开销和容错能力均取决于校验块的数量,不过校验块与副本不同,在正常情况下,校验块没有任何作用,无法分担负载.但是当出现设备故障、节点失效等异常情况时,就可以通过条带上剩余可用的数据块以及校验块进行数据恢复,不过在这个过程中,恢复 1 个数据块往往需要读取数倍的数据,产生大量的 I/O 请求以及网络流量,最终导致高的降级读延迟和显著的数据重建时间.

纠删码系统的恢复开销一般认为来自于 3 个方面:编解码运算、存储 I/O 以及网络传输.但是随着软硬件技术的发展,通过利用 CPU 的新指令[11],可以大大提高编解码效率,因此性能瓶颈主要集中在网络和存储上[12-20].

在本文中,我们将优化方向着眼于数据重建路径,针对数据恢复过程中网络流量过大的问题,利用软件定义网络(software defined networking, SDN)[21]的技术,在交换机上进行部分计算[22],使得数据恢复的解码计算尽量靠近数据存储节点,从而避免数据在网络中大范围传输,克服了恢复节点端的网络瓶颈问题[18-20].因为整个解码过程中,数据流在网络中聚合、计算,然后向下一个交换机传输,所以我们将这种基于网络计算的故障重建方案称为网络流水线(in-network pipeline, INP).

本文的主要贡献有 4 个方面:

1) 提出了 INP 这种基于网络计算的高效故障重建方案,通过利用 SDN 的技术,在交换机上执行部分解码运算,从而减少网络流量,提高降级读以及数据重建的性能.

2) 通过对解码过程进行分析,说明 INP 方案可以适用于多种常见的纠删码,比如 RS 码(Reed-Solomon code)[23-24]、PC 码(product code)[25]以及 LRC 码(local reconstruction code)[13]等,并且 INP 可以和现有的优化技术[12,16,18-20]相结合,进一步提升恢复性能.

3) 扩展了 SDN 控制器的功能模块,使其能够根据交换机的实时负载情况,灵活地选择出合适的交换机参与到解码计算中.

4) 以 RS 码为例,评估了 INP 与传统纠删码系统在降级读操作中的网络流量大小.另外在不同的网络带宽下,测试了正常读、传统降级读以及基于 INP 方案的降级读这 3 种读取方式的性能,结果显示基于 INP 方案的降级读方式可以在一定带宽限制下得到和正常读接近的性能.

1 相关工作

目前,分布式存储系统的规模越来越大,故障更加频繁,存储系统往往通过存储冗余数据以保障在故障发生时,数据不丢失,持续可用.而今,越来越多的分布式文件系统采用纠删码的冗余数据方案,而不是多副本.纠删码在提供足够的可靠性同时,大大降低了存储开销,但代价是高昂的恢复开销.这使得降级读和数据重建的完成时间大大增加,前者会导致用户感受到明显的访问时延,而重建窗口的扩大则会降低系统的可靠性,一旦在数据重建阶段再次发生节点故障,可能会导致数据无法恢复.在不牺牲系统可靠性以及存储开销的前提下,为降低恢复开销,我们提出了一种基于网络计算的故障重建方案,利用交换机的计算能力,在网络中实现数据合并,从而大大减少网络中的数据流量,提高恢复性能.

现在实际生产系统中,比如 Google 的 ColossusFS[8]、Facebook 的 HDFS[7]等,分别使用 2 种不同的 RS 编码:RS(6,3)和 RS(10,4).RS 编码的优势在于可以编码任意 k 个数据块,产生任意 m 个校验块,k 和 m 都可以完全由管理员自定义,一旦发生数据丢失,只需要任意 k 个块(不管是数据块还是校验块)都能完全恢复出所有数据,即可以容忍任意 m 个块的丢失.但 RS 编码会导致比较高的恢复开销,以 RS(10,4)为例,10 个数据块编码产生 4 个校验块,存储开销是 1.4 倍,如果发生数据丢失,则需要从 10 个节点上读取数据,恢复开销是 10 倍,可以看到 RS 编码是在存储开销和恢复开销之间做一个权衡.但是有研究发现,超过 98% 的数据丢失都是单数据块丢失,因此有很多针对单块恢复的优化工作,其中 Microsoft Azure Storage[2,13] 使用一种称为 LRC 的编码,它编码 12 个数据块,产生 2 个全局校验块,另外每 6 个数据块 1 组编码产生 2 个局部校验块,如果出现单块丢失,只需要读取 6 个块即可进行恢复,这种 LRC(12,2,2)编码具有 1.33 倍的存储开销和 6 倍的单块恢复开销,但缺点是牺牲了部分可靠性,不能容忍任意 4 个块的丢失.

关于纠删码的优化研究大致可以分为 2 类:1)一种新的编码方案,比如再生码[22]、阶梯码[14]等;2)系统级的优化,比如双纠删码系统[16],利用负载中存在的数据访问倾斜,对频繁访问的少量热数据使用恢复开销较小的编码,对大量冷数据使用存储开销较小的编码,从而获得低的存储开销和恢复开销.这些新型编码或多或少都是在存储性能和恢复性能上做一些权衡.另外还有针对数据重建方案的优化[17],传统的数据重建方案如图 1 所示.当系统检测到块丢失或者节点故障,则会在某些可用节点上启动重建任务,以一种并行的方式恢复丢失数据.图中数据块 D_1 丢失,需要从其他节点上读取相应块进行恢复,当数据通过网络传输到恢复节点时,恢复节点端的网络带宽将成为瓶颈,从而限制了恢复性能.

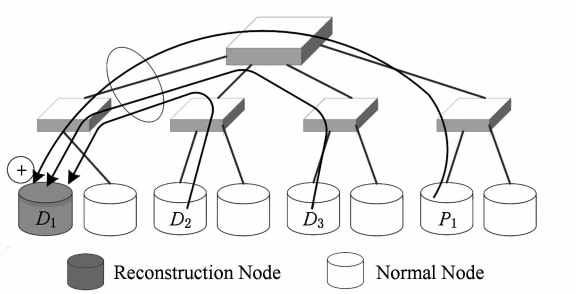

Fig. 1 The traditional erasure code data reconstruction scheme

图 1 传统的纠删码数据重建方案

文献[17]中提出一种新的恢复方案,如图 2 所示,它充分利用各个节点的带宽和计算能力,采用流水线的方式,将前一个节点的数据传输到下一个节点上进行计算,然后将计算结果继续向后续节点传输,直到到达恢复节点,这种方式能比较好地解决恢复节点的瓶颈问题,但它并没有减少传输过程中的网络流量.

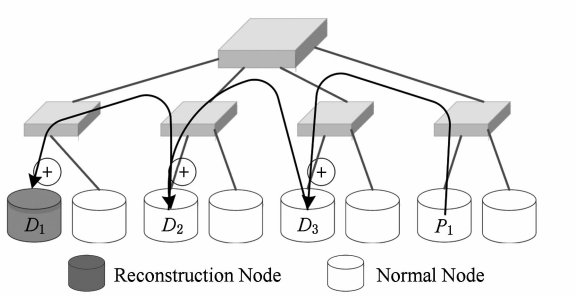

Fig. 2 The erasure code data reconstruction scheme based on pipeline

图 2 基于流水线的纠删码数据重建方案

我们的想法得益于 SDN 技术的快速发展. SDN 是一种新型的可编程网络架构,其最核心的概念是将网络设备的控制面和数据面分离,并强调控制面和数据面的可编程性. 其中可编程控制面技术已经日趋成熟,现在已经有大量的开源控制器实现方案,控制器由一系列基本网络服务模块构成,并通过特定的接口(比如 OpenFlow 协议标准)与交换机进行通信,以此实现拓扑获取、流量统计、路由控制等功能. 在此基础上,用户可以很容易地根据需求开发扩展功能,并对外提供访问接口. 而关于可编程数据面的研究,目前正在逐步完善,其中 P4[26] 语言已经被设计出来,旨在实现交换机功能的动态配置.

在这样的背景下,我们运用 SDN 的技术,提出了基于网络计算的高效故障重建方案 INP,由可编程控制器面实现网络的拓扑管理和调度功能,由可编程数据面实现数据处理功能. 如图 3 所示,通过在交换机上对数据进行合并计算,不仅能够解决网络瓶颈问题,还能大大减少网络中的流量,从而提升恢复性能.

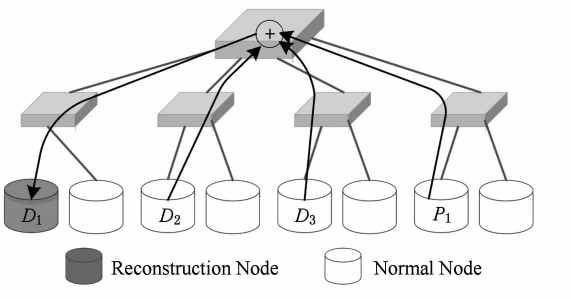

Fig. 3 The erasure code data reconstruction scheme based on in-network computing

图 3 基于网络计算的纠删码数据重建方案

设计 INP 方案主要是为了实现 3 个目标:

1) 更快的降级读. 当客户端访问到不可用的数据块时,能够在很短的时间内完成解码操作,使客户端感受不到太大的延时.

2) 更短的重建时间. 数据重建完成得越快,系统可靠性越高,在新的重建方案下,针对可能发生的多数据块重建,如何保证高并发非常值得思考.

3) 尽可能小地影响交换机正常工作. 因为交换机计算能力有限,在不影响其正常工作的前提下,如何调度交换机也是一个非常重要的工作.

2 设 计

本节首先介绍 INP 方案的整体结构、包括 SDN 控制器、OpenFlow 交换机网络等;之后讨论 INP 在降级读过程中的 I/O 详细原理;最后分析解码计算对交换机性能的影响.

2.1 INP 方案结构

INP 方案的整体结构如图 4 所示,主要由 2 部分组成:SDN 网络和 Hadoop 集群. 其中 SDN 网络又分为 SDN 控制器和 OpenFlow 交换机群. SDN 控制器维护着整个交换机网络的拓扑关系,并负责交换机的调度. 交换机通过 OpenFlow 协议与 SDN 控制器通信,并在数据重建过程中执行部分计算任务.

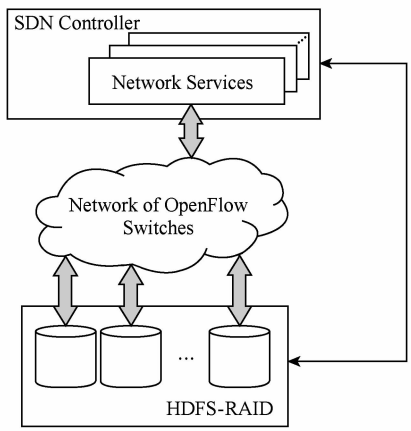

Fig. 4 The architecture of INP

图 4 INP 方案结构

HDFS-RAID(redundant array of independent disks):INP 方案中的 Hadoop 集群是基于 HDFS-RAID 的一个扩展. HDFS-RAID 是 Facebook 基于第 1 代 Hadoop 开发的支持纠删码的系统,它在 HDFS 的基础上实现了一个 RAID 方案,HDFS-RAID 将存储在 HDFS 中的数据块编码产生若干校验块,并删除原来的副本数据块. 一旦发现数据丢失,就会在集群中启动重建任务执行数据重建操作,如果在数据重建期间,客户端需要访问丢失的数据,而此时系统中不存在副本数据,这将导致正常的读取操作抛出异常,从而开始执行降级读操作. 降级读本质上也是一种数据重建操作,与一般意义上的数据重建相比,降级读操作的主要区别在于它不会将重建的数据块写回相应的节点,而只是将其返回给访问的客户端. 在 2.2 节中,我们将主要讨论降级读操作,因为从原理和性能上来说,降级读操作和数据重建操作是大体一致的.

1) SDN 控制器. 在 INP 方案中,我们对 SDN 控制器的功能进行了扩展,当 HDFS-RAID 集群进行降级读或者数据重建操作时,集群中的某个节点

会首先向 SDN 控制器发出请求,控制器利用自身对交换机网络的全局视角,选择出合适的交换机参与解码计算,并将选择结果以一种树型的结构返回给集群,集群凭借该结构与相应交换机建立连接.关于 SDN 控制器对交换机的选择策略,基于交换机的性能考虑,在 SDN 控制器中维护了交换机的状态信息,以此作为评估标准,避免出现热点问题.交换机选择策略的设计将在 2.3 节中详细介绍.

2) OpenFlow 交换机.在 INP 方案中,交换机被用来作为聚合节点,通过在交换机上执行一些简单计算,将来自不同节点的数据流合并,然后将计算结果继续向后转发,从而达到减少网络流量的目的.为了尽可能小地影响交换机正常工作,在整个数据重建过程中,交换机只需要进行异或操作.具体的原理将会在 2.3 节中介绍.

2.2 INP 中的降级读

本节讨论 INP 方案中完整的数据读取流程.如图 5 所示.1)首先用户通过 HDFS 客户端向 HDFS 中的数据块 D_1 发出读取请求.2)但此时数据块 D_1 因为某些故障模式导致数据丢失,而系统可能尚未发现该故障或者正处于数据重建过程中,所以访问的最终结果是抛出异常.3)HDFS 客户端捕获到读取异常,了解到数据块丢失,从而启动降级读操作.在后文的后续说明中,均采用的是 RS(3,2)编码,即 3 个数据块通过编码产生 2 个校验块,在这种编码模式下,如果 1 个块丢失,则需要从剩余 4 个可用块中任意选出 3 个块进行数据重建,在图 5 中,当数据块 D_1 丢失时,选择从数据块 D_3 以及校验块 P_1 和 P_2 中读取数据用于数据重建.而在 INP 架构中,从指定块读取数据之前,HDFS 客户端会向 SDN 控制器发出请求,在请求消息中,HDFS 客户端告知

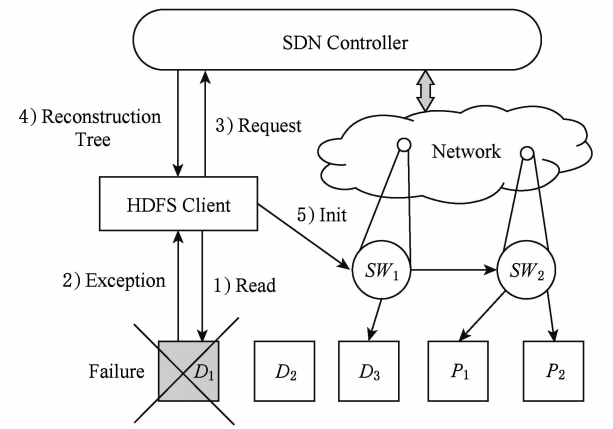

Fig. 5 Degraded reading process of INP

图 5 INP 降级读流程

SDN 控制器需要参与到降级读操作当中的数据块和校验块所在节点位置,SDN 控制器不仅维护着整个 OpenFlow 交换机网络的拓扑信息,还掌握着边缘设备(Hadoop 集群节点)与交换机的连接关系,当 SDN 控制器收到 HDFS 客户端的请求后,它根据交换机拓扑信息和节点位置选择出合适的交换机用于计算.4)SDN 控制器将交换机和节点的连接关系通过一种树型结构返回给 HDFS 客户端,我们将这种树型结构命名为重建树,如图 6(a)所示.5)HDFS 客户端收到重建树后,尝试进行初始化,与各交换机和节点建立连接,在图 5 中 HDFS 客户端向交换机 SW_1 发出建立连接请求并等待响应,SW_1 收到请求后按照重建树结构向交换机 SW_2 和节点 D_3 发出连接请求并等待响应,以此类推.当各节点确认自身状态,同意建立连接,就会向上一级交换机发出确认信号,当 HDFS 客户端收到所有的确认信号,就表示连接建立成功,从而可以开始读取数据,进行解码运算.

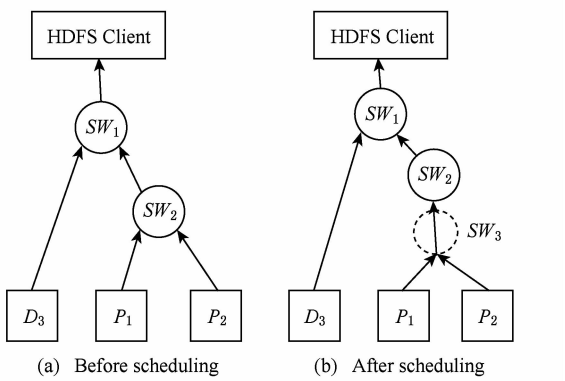

Fig. 6 Structure of reconstruction tree

图 6 重建树结构

2.3 交换机性能分析

在 INP 方案中,交换机在整个数据重建操作中具有至关重要的作用,它是使得网络流量减少的关键设备,在本节中将重点讨论针对交换机性能设计的调度策略以及交换机在解码运算中的工作原理.

1) 交换机调度策略.在实际应用场景中,多用户并发访问或者多数据块重建是非常常见的.以用户访问为例,当不同用户同时引发降级读时,SDN 控制器会根据网络拓扑选择出合适的交换机进行解码运算,这时可能会出现某台交换机被多个降级读所共享的情况.为了避免出现交换机热点问题,对 SDN 控制器功能进行扩展,实现了 3 步调度策略:①如图 7 所示,首先选择出各节点到 HDFS 客户端的最短路径,各路径交点为解码交换机,从而得到如

图 6(a)所示的重建树结构;②如果选择出的交换机的连接数(参与降级读的数量)超过某个阈值,则进行向上调度,即从当前交换机沿数据流方向(图 7 中箭头所指方向)选择出空闲交换机用于解码,所得重建树如图 6(b)所示;③如果最短路径上所有交换机均不符合,则放弃最短路径,直接从高层交换机中选择出空闲交换机.

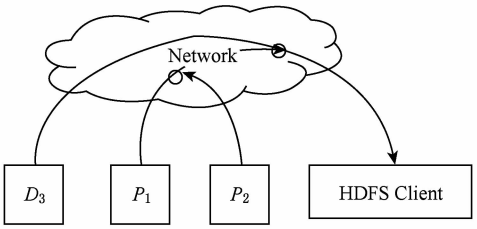

Fig. 7 Shortest-path strategy

图 7 最短路径策略

2) RS 解码运算的分解. 在 2.2 节的讨论中,已经提到了基于网络计算的重建方案的核心思想,即在交换机上进行部分计算,使得网络中传输的流量减少. 考虑到交换机本身的计算能力,为了尽量小地影响其基本的路由转发功能,在重建方案的设计中,在交换机上只需要进行非常简单的异或运算. 具体的工作原理可通过式(1)来解释.

$$D_1=(a_1 \quad a_2 \quad a_3)\times\begin{pmatrix}D_3\\P_1\\P_2\end{pmatrix}=$$

$$a_1\times D_3+a_2\times P_1+a_3\times P_2. \tag{1}$$

RS 码的解码过程可以表示为 2 个矩阵相乘,其中 a_1,a_2,a_3 组成了系数矩阵,D_3,P_1,P_2 是用于解码的数据块和校验块. 将式(1)左边的矩阵运算展开,得到式(1)右边的表达式,可以看到解码操作的实质就是若干系数各自乘上对应的块,然后将结果累加. 需要注意的是,式(1)中的乘法和加法运算都是伽罗华域中的运算,伽罗华域中的乘法运算非常复杂,涉及到查表等操作,而加法运算非常简单,就是异或操作,因此在 INP 的设计中,当从各节点读取数据时,首先在各节点上对原始数据 D_3,P_1,P_2 执行乘法运算,即式(1)中的乘法项 $a_1\times D_3,a_2\times P_1,a_3\times P_2$,然后将计算结果发送给对应的交换机,在交换机上只需要将来自不同节点的数据进行异或合并即可. 另外通过式(1)也可以分析出降级读过程中的流量变化,假设需要降级读取 1 GB 的数据,按照传统的数据重建方案,HDFS 客户端所在的网络将接收到 3 GB 的数据,而 INP 方案中,交换机将来自不同节点的数据进行异或合并,大大减少了向后传输的网络流量,理论上 HDFS 客户端收到的数据量应该为 1 GB,从而解决数据恢复中的瓶颈问题.

2.4 编码扩展性分析

2.3 节中已经详细说明了 RS 码在 INP 方案下的解码过程,可以发现 INP 方案的有效性依赖于交换机上的数据合并过程. 在本节中我们将分析一些常见纠删码的解码过程,从而证明 INP 方案可以运用在多种纠删码系统中.

1) LRC 码. LRC(12,2,2)的编码结构如图 8 所示,12 个数据块使用 RS 编码的算法产生 2 个全局校验块,另外每 6 个数据块通过异或产生 1 个局部校验块. 在解码过程中,如果出现单块故障,则只需要通过式(2)进行异或计算即可,显然可以在交换机上进行数据合并减少网络流量;如果出现多块故障,则需要用到全局校验块,该解码过程和 RS 完全一致.

$$d_4=d_1+d_2+d_3+d_5+d_6+L_1. \tag{2}$$

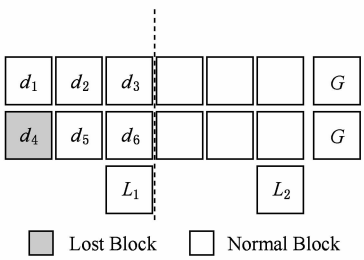

Fig. 8 LRC code

图 8 LRC 编码结构

2) PC 码. PC(2,5)的编码结构如图 9 所示,每行 5 个数据块通过异或产生 1 个局部校验块,每列 2 个数据块通过异或产生 1 个局部校验块,所有的数据块异或产生 1 个全局校验块. 其解码过程类似于 LRC 的单块恢复,每块数据的恢复都涉及单次异或运算,因此 PC 码也可以部署在 INP 方案中.

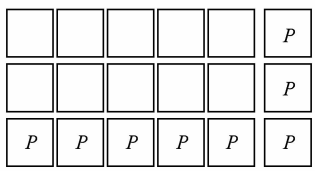

Fig. 9 PC code

图 9 PC 编码结构

3) SD(sector-disk)码[15]. SD 码是一种可以修复扇区错误的纠删码,其结构如图 10 所示. 与传统纠删码相比,SD 码将扇区作为最小编码单元,每行采用 RS 编码算法产生 2 个局部校验块,所有扇区

经过编码产生 2 个全局校验块,其解码过程与 RS 码相同,区别只在于 SD 码重建粒度更小,所以 INP 方案也可以兼容用于扇区修复的纠删码.

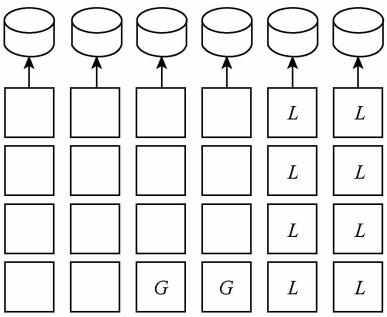

Fig. 10 SD code

图 10 SD 编码结构

3 实 现

3.1 传统纠删码系统

HDFS-RAID 中实现纠删码功能的核心模块主要是 DRFS(distributed raid file system)和 RaidNode,其中 DRFS 是在 HDFS 基础上实现的 RAID 方案,而 RaidNode 进程主要负责数据编码以及数据重建工作. RaidNode 进程会周期性地查询 NameNode,来获取需要编码和需要进行重建的文件条目.

DRFS 客户端是在 HDFS 客户端的基础上进行封装,它的大部分功能其实都是由下层的 HDFS 客户端实现的,在读取数据时,如果出现块丢失等异常情况,DRFS 客户端将会捕获到这些异常,并且定位到异常发生的位置,进而检索同一条带上的可用块用于解码操作,得到请求的数据,这就是传统的降级读操作.

3.2 基于网络计算的纠删码系统

在 INP 方案中,当客户端读取到某个丢失块时,客户端同样会捕获异常,与传统降级读不同的是客户端之后会采取的措施,即开始执行 2.2 节所描述的降级读操作. 在整个 INP 方案的实现中,SDN 控制器选择的是基于 Java 语言的 Floodlight,OpenFlow 交换机则采用 Open vSwitch 虚拟交换机. 在客户端捕获到读取异常之后,首先向条带上的其余数据块和校验块发起访问,确定有足够多的可用块,并获取这些块所在节点的 IP 地址. 因为 Floodlight 采用 REST(representational state transfer)风格的编程接口对外提供服务,所以客户端将节点的 IP 地址通过 HTTP 请求的方式发送给 Floodlight,Floodlight 收到请求后,根据维护的交换机负载信息,对交换机进行调度. 当成功选择出合适的交换机后,更新交换机负载信息,并将交换机和节点的连接方式以重建树的结构发送回客户端. 然后客户端开始计算式(1)中的系数矩阵,也称为解码矩阵,在与交换机以及各节点建立连接的过程中,将系数矩阵以及文件偏移等信息发送到对应节点. 当节点接收到客户端的连接请求之后,它会去打开指定文件,根据文件偏移定位数据,如果在这个过程中发生异常,会向客户端发送一个值为 false 的确认,只要有 1 个节点不能正常连接,客户端收到的确认就是 false,只有当所有的节点以及交换机都做好数据传输准备,客户端才会收到值为 true 的确认.

连接成功建立之后,客户端会采用流水线的方式向节点发出读取请求. 为了提升读取效率,客户端上建立了双缓冲区,在连接建立后客户端启动 1 个子线程向缓冲区中写数据,写线程会一直循环检查 2 个缓冲区,一旦发现某个缓冲区为空,就会向服务器发出读取数据请求. 采用双缓冲区这样的方式,是为了读线程在读某个缓冲区时,写线程可以去填满另一个缓冲区,当读线程读完前一个缓冲区时,可以无等待地切换到另一个缓冲区,而同时写线程会去填满之前被读完的缓冲区.

当写线程读取到 1 个块的块尾时,表示已经成功读取 1 个丢失的块,这时客户端就应该与交换机以及各节点断开连接,在这个过程中客户端会向 Floodlight 发送一个断开连接的信号,收到此信号的控制器就会更新交换机的负载信息. 如果客户端还需要读取文件的下一个块,且该数据块也发生丢失,则需要重新建立连接,重复整个流程. 因为即使是文件中连续的 2 个块,也可能编码在不同的条带上.

4 测 试

4.1 测试环境

如图 11 所示,在 1 台配置有 2TB 磁盘、12 GB 内存以及 2 个 4 核 2.4 GHz Intel Xeon E5620 CPU 的物理机上构建了 1 个由 11 个 Docker 容器和 1 台虚拟交换机组成的小集群,其中 8 个容器作为 DataNode 服务器节点,另外 3 个容器分别作为 NameNode、HDFS 客户端以及交换机的计算节点. 容器内运行 Ubuntu 14.04.2 操作系统,Docker 版本为 1.7.1,SDN 控制器是 Floodlight-0.90,交换机使用 Open vSwitch 2.0.2 来模拟.

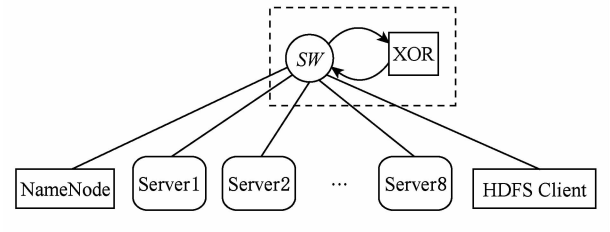

Fig. 11 Experimental cluster

图 11 测试集群

因为虚拟交换机本身并不具备数据解码中的计算功能,而修改其源代码代价太大.因此我们为每个虚拟交换机绑定 1 个计算节点,如图 11 所示,当交换机接收到服务器节点传来的数据时,首先将数据转发给对应的计算节点,在计算节点上完成异或合并运算,然后再发送回交换机,并继续向后传输.但是采用这种方式,会给测试引入不必要的开销,在 4.3 节我们将分析这种方式对测试结果造成的影响,并合理估计这部分额外开销.

另外关于纠删码的设置,我们采用的编码是 RS(3,2),即每 3 个数据块编码产生 2 个校验块,5 个块一起构成 1 个条带,基于这种编码,每次降级读取 1 个块需要连接另外 3 个可用块.另外设置 HDFS 块大小为 128 MB,使用 HDFS-RAID 默认的块分布方案将编码后的数据块和校验块平均分布在集群中,确保同一纠删码条带上的任意 2 个块不在同一节点上.

在 4.2~4.4 节中,我们将分别从网络流量、降级读延迟以及带宽竞争力这 3 个方面说明 INP 方案的有效性.

4.2 流量测试

通过第 2 节分析可知,影响降级读性能的一个重要因素是网络流量,在我们的测试集群中,网络瓶颈主要出现在交换机与客户端相连的链路上.因为测试过程中客户端降级读取 1 GB 的数据,而编码方案采用的是 RS(3,2)编码,即每次需要读取 3 倍的数据用于解码操作,所以理论上客户端的降级读会从服务器上获取 3 GB 的数据,传统降级读方式中 3 GB 的数据将全部流经交换机到达客户端,在客户端上进行解码得到丢失的数据.而基于网络计算的降级读方式因为在交换机上已经完成了计算,因此理论上只需要向客户端传输 1 GB 数据.我们通过调用 Floodlight 控制器的接口可以获取到交换机各端口的输入输出流量,经过简单处理后的结果如图 12 所示,其中 HDFS-RAID 表示传统降级读,INP 表示基于网络计算的降级读.

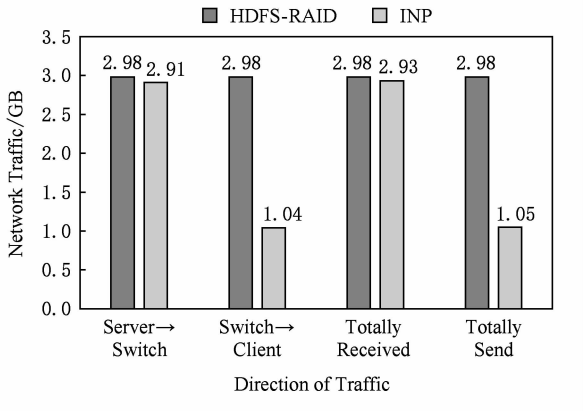

Fig. 12 Comparison of network traffic

图 12 网络流量对比

第 1 组数据测的是交换机从各服务器节点读取的数据量,可以看到在传统降级读过程中,需要从集群中读取 2.98 GB 的数据,与理论分析值 3 GB 基本相符,另外 INP 方案中的降级读从服务器获取了 2.91 GB的数据,略少于传统降级读,但认为是在测试误差范围内.

我们比较关注的是第 2 组数据:交换机转发给客户端的数据量.图 12 中数据显示 INP 方案中交换机转发给客户端的数据量与传统降级读的比例为 1:3,基本与理论值相符.

后面 2 组数据测试的是交换机的总的接收数据量和发送数据量,统计这 2 组数据主要是为了说明在降级读中,交换机接收的流量几乎都来自服务器,而转发的流量都是送往客户端.

测试中我们采用的是一种恢复开销相对较小的编码,每次恢复 1 个数据块只需要读取 3 倍的数据量.而在 Google ColossusFS 中使用的是 RS(6,3),Facebook HDFS-RAID 使用的是 RS(10,4),如果在 INP 方案中部署这 2 种编码,能够减少的网络流量会更多.

4.3 降级读延时

在本节中,我们将分别在不同网络带宽下对正常读、传统降级读和 INP 方案中的降级读的时间进行比较.测试结果如表 1 和图 13 所示,其中 HDFS 表示正常读.

关于网络带宽的设置,采用的办法是在交换机到客户端的出口端口上创建 1 个 QoS(quality of service)队列,并设置最大最小速度,以此来限制网络带宽.需要注意的是所有的网络限制都是针对交换机到客户端的出口带宽,其余链路不做限制,以此来分析网络流量所引起的瓶颈问题,同时说明 INP 方案的有效性.

Table 1 Time Required to Read 1 GB Data at Different Network Bandwidths

表 1 不同网络带宽下读取 1 GB 数据所需时间

Bandwidth /Mbps	Time Required to Read 1 GB Data/s		
	HDFS-RAID	INP	HDFS
50	542.34	191.93	186.41
100	263.84	96.55	93.44
200	132.14	51.92	46.35
400	67.26	32.52	25.67
500	57.08	29.72	19.68
1 000	29.78	19.50	10.39
2 000	16.21	15.33	6.28
10 000	8.79	14.30(9.82)	1.65

Note: The value in parentheses is the correction data, because the simulation method used in our experiments has a large impact on the result at 10 000 Mbps.

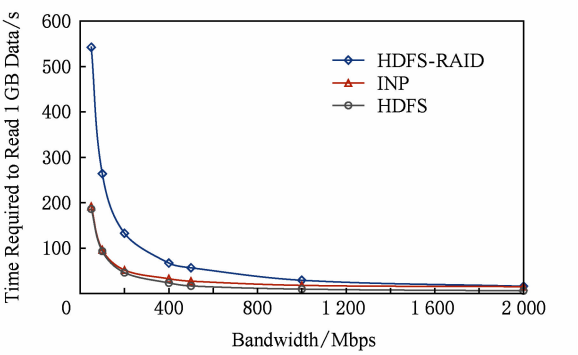

Fig. 13 Time required to read 1 GB data at different network bandwidths

图 13 不同带宽下读取 1 GB 数据所需时间对比

在整个测试部分,我们以正常读取 1 GB 数据的时间作为基准来分析传统降级读和基于 INP 方案的降级读的性能. 在 10 000 Mbps 下,测得客户端正常读取 1 GB 的数据需要 1.65 s,基于流水线原理,读取时间取决于瓶颈带宽和瓶颈链路上的网络流量,在传统降级读中,客户端所在链路需要传输 3 倍于正常读的流量,因此理论上传统降级读读取 1 GB 的数据需要 3 倍的正常读时间,而实际测试结果是 8.79 s. 这主要是因为降级读除了网络传输开销,还存在存储 I/O 以及解码计算开销.

再观察 10 000 Mbps 下 INP 方案的测试结果. 因为 4.2 节实际测得交换机最后转发到客户端的流量和正常读是一致的,即 INP 方案中瓶颈链路带宽以及瓶颈链路传输的网络流量均与正常读相同,所以 INP 方案中的网络传输时间理论上也应该与正常读一致,不过根据 4.1 节所述,因为硬件设备受限,所以在仿真测试中为交换机绑定 1 个计算节点,按照这种方式,在 10 000 Mbps 下,虽然消除了客户端的网络瓶颈,但是计算节点处的链路反而成为了瓶颈,而且交换机转发给计算节点的数据和计算节点回送给交换机的计算结果均在同一链路上传输,所以该段链路上的网络流量是正常读的 4 倍,为了消除这部分开销,我们测试了 3 GB 数据在万兆链路下的网络传输时间,以此来近似估计交换机与计算节点之间的额外开销,如表 1 所示,我们列出了 INP 方案的原始测试结果和修正之后的结果,消除开销之后 INP 方案读取时间为 9.82 s. 但即使是修正之后的数据,也远远大于正常读,甚至要比传统降级读时间略长,不仅没有减少恢复开销,反而增大. 分析 INP 方案的额外时间开销,不仅包括存储 I/O 和解码计算,另外在 INP 方案中,客户端在读取数据之前还需要访问 SDN 控制器,并等待控制器返回重建树信息,之后还要与交换机、服务器建立连接,这些操作都存在一定的时间开销. 而且由于客户端链路的网络带宽达到 10 000 Mbps,数据传输速度较快,导致网络开销并不是影响恢复性能的主导因素. 这就使得 INP 方案性能反而没有传统降级读性能好. 由于集群规模的限制,我们在测试中只能采用条带较短的编码方案,但是考虑到实际系统中往往会部署恢复开销更大的编码,这时网络流量将会成倍增加,这些都将有利于 INP 方案的性能发挥.

当我们将带宽设置为 2 000 Mbps 时,交换机与计算节点之间的链路不再是网络瓶颈,因此不用对测试结果进行修正. 测试结果显示 INP 方案性能优于传统降级读,且随着带宽资源越来越紧张,INP 方案的优势也越来越明显,甚至接近正常读时间. 如图 13 所示,在 1 000 Mbps 下,INP 方案可以减少 50% 的降级读开销;而在 200 Mbps 下,INP 方案已经达到正常读的性能. 这主要是因为随着带宽降低,降级读的恢复开销主要取决于网络传输时间,而 INP 方案大大减少了网络流量,因此可以获得非常大的性能提升. 而且目前用于测试的 INP 方案只是一个简单原型,在性能优化上还有一些工作有待完成.

4.4 网络竞争力

在 4.3 节的测试中,评估的都是某种读取方式独享整个网络带宽时的性能. 而在真实应用场景里,往往是既有正常读,也有降级读,2 种读取方式并行执行. 在这部分,将考量正常读+正常读、正常读+传统降级读、正常读+INP 降级读这 3 种共享读取模式下各读取方式的性能表现. 设置这组测试的目的

主要是为了评估降级读与正常读产生的数据流在网络中的竞争力表现.

如图 14 所示,在 1 000 Mbps 下,第 1 组数据显示的是独占模式下各读取方式的性能,即 4.3 节中测得的数据.第 2 组数据表示的是当网络中存在 2 个客户端同时读取数据时正常读的性能表现.结果符合预期,当网络中流量成倍增加时,正常读性能成倍降低,图 14 中显示 2 个正常读操作的时间都是原来独占网络带宽时的 2 倍.再观察第 3 组数据,当网络中同时存在正常读和传统降级读时,正常读的完成时间接近于独占网络带宽时的 3 倍,这是因为当网络中存在降级读时,网络流量大大增加,严重占用了网络带宽,实验结果显示在这种共享模式下,正常读分配有 1/3 的带宽,传统降级读占用了剩下的 2/3,但是考虑到传统降级读的流量是正常读的 3 倍,所以说明正常读的网络竞争力会略强于传统降级读.第 4 组数据测的是正常读和 INP 降级读的共享模式,其中正常读的时间相比于独占带宽增加了 1 倍,即正常读在混合模式下占用了一半的带宽,这与第 2 组数据情况相同,因此可以认为 INP 方案在网络竞争力方面与正常读表现一致.

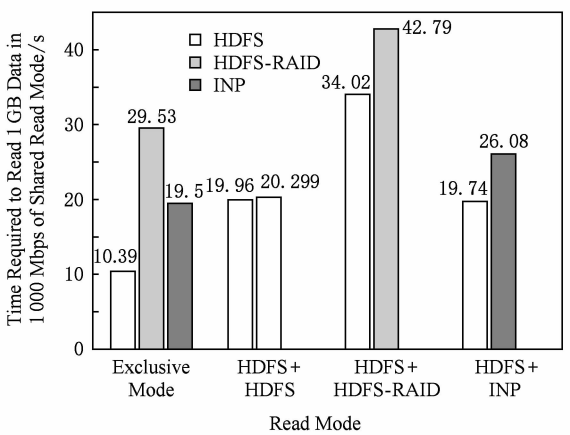

Fig. 14 Shared read mode in 1 000 Mbps

图 14 1 000 Mbps 下的共享读取模式

另外我们还分别在 500 Mbps 下和 100 Mbps 下对 3 种混合读取模式进行了测试,如图 15 和图 16 所示.在这 2 种带宽下,正常读和 INP 降级读的混合模式中,正常读均占有一半的带宽,与 1 000 Mbps 下表现一致.特别是 100 Mbps 下的测试结果,非常直观地显示了 INP 降级读具有和正常读一样的网络竞争力.如图 16 所示,因为在该带宽下,INP 降级读已经具有接近正常读的时间开销,所以图 16 中第 4 组数据和第 2 组数据表现完全吻合,这也说明了可以将 INP 降级读等价于正常读来看待.

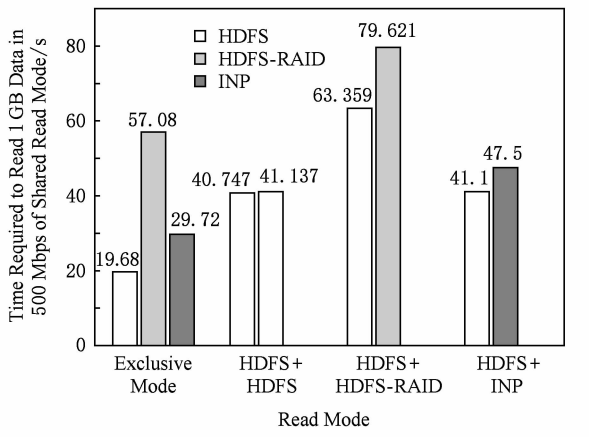

Fig. 15 Shared read mode in 500 Mbps

图 15 500 Mbps 下的共享读取模式

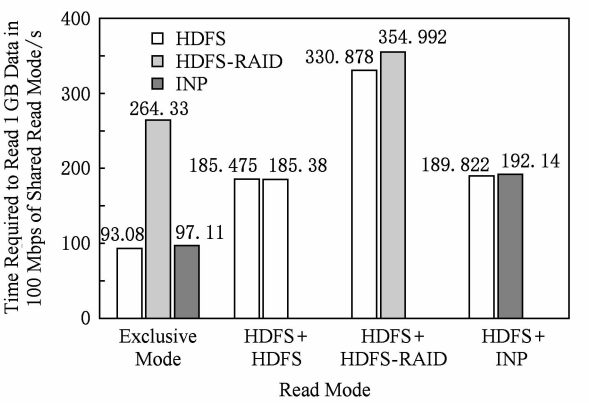

Fig. 16 Shared read mode in 100 Mbps

图 16 100 Mbps 下的共享读取模式

4.5 未来工作

在本次测试过程中,我们从网络流量、降级读延迟以及网络竞争力说明了 INP 方案的有效性,它能够大大减少降级读过程中的网络流量,从而提升恢复性能,甚至可以在一定带宽下接近正常读水平,而且我们可以预见到,在恢复开销更大的编码系统上,我们的方案会获得进一步的性能提升.

但是我们在测试环境上还存在一些不足,比如单机性能有限,在单机上模拟集群,对于集群的很多特性无法表现出来,比如存储 I/O 操作,这也是一个影响恢复性能的重要因素;另外一个不足是目前没有在实际的 OpenFlow 交换机上去实现我们的功能,这也导致实验过程中引入了一些额外开销,虽然我们在测试中修正了这部分开销,但还是不能完全代表真实场景.关于测试部分,在代码实现上还存在不少优化空间,可以进一步提升 INP 方案的性能,另外目前 INP 方案尚未与更多编码方案进行对比测试.这些问题都将作为我们未来的研究方向.

5 总 结

本文提出一种新型的基于网络计算的高效故障重建方案 INP 来解决纠删码系统中恢复开销太大的问题,考虑到影响恢复开销的 3 个因素:较高的网络传输开销、解码计算复杂以及大量的磁盘读写. INP 方案选择从网络传输量这个主要因素上进行优化.

通过分析传统纠删码系统恢复数据时的网络情况,可以发现客户端所在链路经常需要传输大量的数据,这是由 RS 码的解码算法决定的,传统纠删码系统在解码时总是把所有需要的数据都传输到 1 个节点上进行计算,这就使得该节点所在链路成了整个网络的瓶颈,从而导致恢复开销增大.

因此我们利用 SDN,设计出一种可以在交换机上进行聚合计算的数据重建方案,以此来减少向后传输的网络流量,最终解决客户端处的网络瓶颈问题.

实验结果显示 INP 方案能够大大减少降级读过程中的网络流量,从而提升恢复性能,在 1 000 Mbps 下能够降低 50%的降级读开销,而且随着带宽资源紧缺,INP 方案中的降级读可以接近正常读水平.

参 考 文 献

[1] Apache. The Hadoop distributed file system [EB/OL]. [2017-05-16]. http://wiki.apache.org/hadoop/HDFS

[2] Huang Cheng, Simitci H, Xu Yikang, et al. Erasure coding in windows azure storage [C] //Proc of the 21st USENIX Conf on Annual Technical Conf. Berkeley, CA: USENIX Association, 2012: 15-26

[3] Schmuck F, Haskin R. GPFS: A shared-disk file system for large computing clusters [C] //Proc of the 1st USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2002: 231-244

[4] Schroeder B, Lagisetty R, Merchant A. Flash reliability in production: The expected and the unexpected [C] //Proc of the 14th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2016: 67-80

[5] Ghemawat S, Gobioff H, Leung S T. The Google file system [C] //Proc of the 19th ACM Symp on Operating Systems Principles. New York: ACM, 2003: 29-43

[6] Luo Xianghong, Shu Jiwu. Summary of research for erasure code in storage system [J]. 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[J]. 计算机研究与发展, 2012, 49(1): 1-11)

[7] Facebook. HDFS-RAID [EB/OL]. [2017-02-28]. https://wiki.apache.org/hadoop/HDFS-RAID

[8] Google. Colossus, successor to Google file system [OL]. [2017-01-29]. http://static.googleusercontent.com/media/research.google.com/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf

[9] Luse P, Greenan K. Swift object storage: Adding erasure codes [OL]. [2017-01-09]. http://www.snia.org/sites/default/files/Luse_Kevin_SNIATutorialSwift_Object_Storage2014_final.pdf

[10] IBM. IBM cloud object storage [EB/OL]. [2016-12-20]. https://www.ibm.com/cloud/object-storage

[11] Plank J S, Greenan K M, Miller E L. Screaming fast Galois field arithmetic using Intel SIMD instructions [C] //Proc of the 11th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2013: 298-306

[12] Rashmi K V, Shah N B, Gu Dikang, et al. A "hitchhiker's" guide to fast and efficient data reconstruction in erasure-coded data centers [C] //Proc of the 28th ACM Conf on SIGCOMM. New York: ACM, 2014: 331-342

[13] Sathiamoorthy M, Asteris M, Papailiopoulos D, et al. XORing elephants: Novel erasure codes for big data [J]. Proceedings of the VLDB Endowment, 2013, 6(5): 325-336

[14] Li Mingqiang, Lee P P C. STAIR codes: A general family of erasure codes for tolerating device and sector failures in practical storage systems [C] //Proc of the 12th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2014: 147-162

[15] Plank J S, Blaum M, Hafner J L. SD codes: Erasure codes designed for how storage systems really fail [C] // Proc of the 11th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2013: 95-104

[16] Xia Mingyuan, Saxena M, Blaum M, et al. A tale of two erasure codes in HDFS [C] //Proc of the 13th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2015: 213-226

[17] Huang Jianzhong, Liang Xianhai, Qin Xiao, et al. PUSH: A pipelined reconstruction I/O for erasure-coded storage clusters [J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(2): 516-526

[18] Rashmi K V, Shah N B, Gu Dikang, et al. A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster [C] //Proc of the 5th USENIX Conf on Hot Topics in Storage and File Systems. Berkeley, CA: USENIX Association, 2013: 8-8

[19] Khan O, Burns R, Plank J, et al. Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads [C] //Proc of the 10th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2012: 251-264

[20] Rashmi K V, Preetum N, Wang Jingyan, et al. Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth [C] //Proc of the 13th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2015: 81-94

[21] Nunes B A A, Mendonca M, Nguyen X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks [J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634

[22] Dimakis A G, Godfrey P B, Wu Yunnan, et al. Network coding for distributed storage systems [J]. IEEE Transactions on Information Theory, 2010, 56(9): 4539-4551

[23] Tamo I, Barg A. A family of optimal locally recoverable codes [J]. IEEE Transactions on Information Theory, 2014, 60(8): 4661-4676

[24] Reed I S, Solomon G. Polynomial codes over certain finite fields [J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 300-304

[25] Roth R. Introduction to Coding Theory [M]. Cambridge, UK: Cambridge University Press, 2006

[26] The P4 Language Consortium. P4factory [EB/OL]. [2016-12-25]. https://github.com/p4lang/p4factory

Tang Yingjie, born in 1995. MSc candidate. His main research interests include storage security and distributed storage system.

Wang Fang, born in 1972. PhD, professor. Member of CCF. Her main research interests include massive storage systems, parallel file systems, non-volatile storage, large scale graph data storage and processing.

Xie Yanwen, born in 1990. PhD candidate. Student member of CCF. His main research interests include reliable storage and distributed storage system.