软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(4):962−986 [doi: 10.3724/SP.J.1001.2012.04175]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

云计算环境下的分布存储关键技术*

王意洁, 孙伟东+, 周 松, 裴晓强, 李小勇

(国防科学技术大学 计算机学院 并行与分布处理国家重点实验室,湖南 长沙 410073)

Key Technologies of Distributed Storage for Cloud Computing

WANG Yi-Jie, SUN Wei-Dong+, ZHOU Song, PEI Xiao-Qiang, LI Xiao-Yong

(National Key Laboratory of Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: wd.sun@qq.com

Wang YJ, Sun WD, Zhou S, Pei XQ, Li XY. Key technologies of distributed storage for cloud computing. *Journal of Software*, 2012,23(4):962−986. http://www.jos.org.cn/1000-9825/4175.htm

Abstract: Considered as the next generation computing model, cloud computing plays an important role in scientific and commercial computing area and draws great attention from both academia and industry fields. Under cloud computing environment, data center consist of a large amount of computers, usually up to millions, and stores petabyte even exabyte of data, which may easily lead to the failure of the computers or data. The large amount of computers composition not only leads to great challenges to the scalability of the data center and its storage system, but also results in high hardware infrastructure cost and power cost. Therefore, fault-tolerance, scalability, and power consumption of the distributed storage for a data center becomes key part in the technology of cloud computing, in order to ensure the data availability and reliability. In this paper, a survey is made on the state of art of the key technologies in cloud computing in the following aspects: Design of data center network, organization and arrangement of data, strategies to improve fault-tolerance, methods to save storage space, and energy. Firstly, many kinds of classical topologies of data center network are introduced and compared. Secondly, kinds of current fault-tolerant storage techniques are discussed, and data replication and erasure code strategies are especially compared. Thirdly, the main current energy saving technology is addressed and analyzed. Finally, challenges in distributed storage are reviewed as well as future research trends are predicted.

Key words: cloud computing; data center; distributed storage; fault-tolerance; data center network; replication; erasure code; energy saving

摘 要: 云计算作为下一代计算模式,在科学计算和商业计算领域均发挥着重要作用,受到当前学术界和企业界的广泛关注.云计算环境下的分布存储主要研究数据在数据中心上的组织和管理,作为云计算环境的核心基础设施,数据中心通常由百万级以上节点组成,存储其上的数据规模往往达到PB级甚至EB级,导致数据失效成为一种常态

* 基金项目: 国家重点基础研究发展计划(973)(2011CB302601); 国家自然科学基金(60873215); 湖南省自然科学杰出青年基金(S2010J5050); 高等学校博士学科点专项科研基金(200899980003)

收稿时间: 2011-05-24; 修改时间: 2011-08-31; 定稿时间: 2011-12-31; jos 在线出版时间: 2012-02-06

CNKI 网络优先出版: 2012-02-06 17:04, http://www.cnki.net/kcms/detail/11.2560.TP.20120206.1704.001.html

© 中国科学院软件研究所 http://www.jos.org.cn

行为,极大地限制了云计算的应用和推广,增加了云计算的成本.因此,提高可扩展性和容错性、降低成本,成为云计算环境下分布存储研究的若干关键技术.针对如何提高存储的可扩展性、容错性以及降低存储的能耗等目标,从数据中心网络的设计、数据的存储组织方式等方面对当前分布存储的关键技术进行了综述.首先,介绍并对比了当前典型的数据中心网络结构的优缺点;其次,介绍并对比了当前常用的两种分布存储容错技术,即基于复制的容错技术和基于纠删码的容错技术;第三,介绍了当前典型的分布存储节能技术,并分析了各项技术的优缺点;最后指出了当前技术面临的主要挑战和下一步研究的方向.

关键词: 云计算;数据中心;分布存储;容错;数据中心网络;复制;纠删码;节能

中图法分类号: TP316 **文献标识码**: A

信息技术的发展极大地促进了社会和科学的发展与进步,同时,各行各业不断推进的信息化又给信息技术带来了巨大的挑战,推动着信息技术不断向前发展.随着科学、商业和日常生活中信息化程度的不断加深,产生的数据量越来越庞大,在高能物理、生物学、天文学、气候建模、气象预报和地震预测等科学计算领域以及Web 搜索和社会网络等商业计算领域中尤为突出.

云计算是随着计算、存储以及通信技术的快速发展而出现的一种崭新的共享基础资源的商业计算模型,被誉为“革命性的计算模型”(张亚勤,未来计算在“云-端”,http://blog.sina.com.cn/s/blog_596ccc870100aps1.html).云计算不同于传统的以个人计算机为中心的本地计算,它以互联网为中心,通过构建一个或多个由大量(百万级以上)普通机器和网络设备连接构成的数据中心,把海量的数据存储到数据中心上,向上层的服务和应用提供安全、可靠、快速、便捷、透明的数据存储和计算服务.数据中心是云计算的基础,企业依靠数据中心进行业务操作,服务提供商依靠数据中心提供服务而盈利,内容提供商依靠数据中心提供有用的内容并获得利润.云计算环境下的分布存储技术主要研究数据在数据中心上的存储、组织和管理,并向上层应用提供安全的、可靠的、可扩展的、高效的数据存储服务;为了提供更好的数据存储服务,还需要关注数据中心网络的构建;为了推广应用,还需要关注硬件基础设施与设备运行的能耗问题.

1 引 言

随着信息技术的发展,在科学计算、商业计算等众多应用领域中产生了规模巨大的数据,而且数据量仍在快速增加,呈海量形式发展.

在科学计算方面,如高能物理(http://www.ipac.caltech.edu/2mass/)、天文学(http://www.sdss.org/)、生物学[1]、地球科学(http://sedac.ciesin.columbia.edu/ddc/observed/index.html)等领域都产生了规模庞大的数据,据估计,每年的数据规模达到若干 PB[2].在商业计算方面,Web 搜索、社会网络等需要处理的数据规模也非常庞大,例如,Google 和 Facebook 等应用产生的数据达到了 PB 甚至 EB 级.按照摩尔定律,处理器的速度每 18 个月就会翻一番,光纤技术的发展也使得数据在网络上的传输速度大大加快.但是,数据存取受限于存储介质的机械运动,使得数据存取成为制约信息技术发展的主要瓶颈.在云计算环境下,海量数据集中存储在若干数据中心上,数据的规模扩大,也使得数据存取的瓶颈更加严重.为了提高数据存取的速度,海量数据一般被分布存储到数据中心不同的节点上以支持并行存取.在云计算环境下,海量数据存储的组织和管理在可扩展性、容错性以及成本控制方面表现出了更高的需求.

云计算是一个为用户提供可配置的、共享基础资源的计算模型,它使得用户能够在云服务提供商很少参与的情况下,方便、实时地访问网络、存储、计算等资源[3].云计算提供商通过把大量的节点和网络设备连接在一起,构建一个或若干个大规模的数据中心,然后以数据中心为基础向用户提供各种层次的服务,例如基础设施服务、平台服务、存储服务和软件服务等[4].云计算具有超大规模、高可扩展性、高可靠性、虚拟化、按需服务和价格低廉等特点,能够很好地满足海量数据存储的要求.

在云计算环境下,海量数据被存储到同一个数据中心的不同节点上,甚至不同数据中心的节点上,但是数据的位置和组织方式对用户是透明的,用户只需要通过服务商提供的一套简便的使用接口(如数据访问接口)向数

© 中国科学院软件研究所 http://www.jos.org.cn

据中心存取数据即可.数据的存储、组织、管理以及可靠性、可用性保证均由云提供商负责.云计算使得用户不必构建自己的数据中心,降低了用户的成本.他们只需要根据自身需求支付一定的费用,就能够方便地把数据存储到数据中心上.在处理时向数据中心提交任务,最后获得结果.

虽然云计算被誉为一个“革命性的计算模型”,但是它和传统的 P2P 计算、网格计算是一脉相承的.P2P 强调把分散在互联网上的各种资源组织起来提供服务,但是受限于网络因素,性能较低,而且节点的高度动态性降低了数据的可用性;网格技术强调的是分布在不同位置的各个组织和团队之间的资源共享.与 P2P 技术相比,云计算环境下数据中心内部和数据中心之间的网络状况更好,节点也更加稳定;与网格计算相比,云计算的规模更加庞大[5].云计算环境虽然构建成本和管理成本较高,但是用户只需要按需付费,成本较低,服务提供商则保证了数据的可靠性、可用性.对用户而言,极大地降低了海量数据管理带来的负担.分布存储技术是云计算的基础,主要研究如何存储、组织和管理数据中心上的大规模海量数据.由于面临的数据规模和用户规模更加庞大,在可扩展性、容错性以及成本控制方面面临着更加严峻的挑战.

- 可扩展性

传统的提高可扩展性的方法一般通过冗余的磁盘预留的方式实现,这种方法可以在一定程度上保证有足够的存储空间.但是,云计算环境下的现代数据中心的节点规模动辄几万甚至几十万;此外,数据中心上存储的数据以 PB 甚至 EB 计,而且数据中心的规模和存储的数据规模也会随着应用的拓展快速增加.因此,任何一个云服务提供商的数据中心都不可能通过磁盘预留的方式在建立之初就完全规划好.比如:Google 目前部署在全球的数据中心有 36 个,单个数据中心的计算机节点将达到数百万个[6];微软宣称将在全球建设超过 20 个数据中心(http://www.circleid.com/posts/microsoft_supersize_cloud_data_centers/),并于 2009 年 9 月在芝加哥建成世界最大的模块化数据中心,包括 220 个集装箱,每个集装箱有 1800~2500 台机器(http://www.datacenterknowledge.com/archives/2008/10/20/microsoft-pue-of-122-for-data-center-containers/),而且微软的服务器数量每 14 个月就会增长一倍,超过了摩尔定律的增长速度.庞大的规模和快速的增长对分布存储的可扩展性提出了更高的要求,不但要求数据中心网络具有良好的可扩展性,而且数据的组织结构也必须具有良好的可扩展性,以适应不断拓展的应用需求.

- 容错性

传统的通过高性能服务器、专用的存储设备或者 RAID 技术等提高容错性的方法成本高昂,难以满足追求利润的云计算提供商,而且庞大的节点规模和数据规模极大地提高了失效的概率.在云计算环境下,失效成为一种常态行为.例如,Google 公司在 2006 年 3 月的一份报告[7]中指出:在其数据中心内,平均每个 MapReduce 作业运行过程中就有 5 个节点会失效,在一个拥有 4 000 个节点的运行 MapReduce 作业的数据中心内,平均每 6 个小时就会有一个磁盘失效(http://wiki.apache.org/hadoop/HadoopPresentations).失效会给云服务提供商以及用户带来巨大的损失.2008 年 2 月 15 日,亚马逊公司的服务宕机事件使得几千个依赖亚马逊 EC2(弹性云计算)和 S3(云存储)的网站受到影响;2009 年 3 月,Google Docs 出现故障(http://blogs.wsj.com/digits/2009/03/08/1214/),随后,美国电子隐私信息中心请求联邦商务委员会介入调查,以确定 Google 的云计算服务对隐私和安全的保障(http://cloudstoragestrategy.com/2009/03/trusting-the-cloud-the-ftc-and-google.html).频繁的失效行为及其带来的巨大损失,使得容错成为云计算环境下分布存储面临的一个亟待解决的重要挑战.在云计算环境下提高分布存储的容错性,不但要研究节点之间的互联关系,提高物理拓扑结构的容错性,还要研究存储在节点上的数据的组织和管理,提高数据的容错性.

- 成本控制

传统的分布存储因为其节点和数据的规模较小,对能耗的考虑较少,而且企业一般都愿意用成本来换取效率和可靠性等.在云计算环境下,分布存储的规模巨大,能耗开销也很大.为了使得设备能够正常运转,能耗还要包括制冷设备的能耗.在 24×7 的不间断运行模式下,能耗成为构成数据中心存储开销的一个重要组成部分.在美国,2000 年~2005 年之间,数据中心的能耗翻了一番.研究人员调查还发现,一台服务器 4 年的能耗基本上等于其硬件的成本(http://arstechnica.com/business/news/2009/10/datacenter-energy-costs-outpacing-hardware-prices),

© 中国科学院软件研究所 http://www.jos.org.cn

而且降低能耗能够提高磁盘等硬件设备的运行寿命,进而降低数据中心的成本.云计算提供商作为依靠服务盈利的企业,降低能耗进而降低成本是一个必须追求的目标,而且节约能耗可以节约能源,促进环境保护,节能技术已成为分布存储设计中与效率、容错并列的关键技术之一.

针对云计算环境下分布数据存储面临的几个主要挑战,研究了解决这些挑战的若干关键技术,包括数据中心网络拓扑的构建技术、数据复制技术、数据编码技术以及分布存储中的节能技术等.综述了各项关键技术当前的研究现状,分析指出了现有技术存在的问题,并展望了未来研究的发展方向.

本文第 1 节概述云计算环境下分布存储研究的背景和面临的挑战.第 2 节讨论现有的数据中心网络拓扑结构,并对其进行分类和比较分析.第 3 节阐述云计算环境下分布存储中采用的两种容错技术,基于复制的容错技术和基于纠删码的容错技术,并对比分析它们在存储效率、容错能力、修复成本等方面的优缺点和面临的挑战.第 4 节介绍云计算环境下分布存储中节能技术的意义,并阐述常用的一些节能技术及其研究现状.第 5 节分析展望未来研究的趋势.最后对文章的内容进行总结.

2 数据中心网络

数据中心是云计算环境下分布存储的基础,云计算环境下的分布存储研究数据在数据中心上的组织和管理.为了提高可靠性和可用性,一般需要为一个数据对象创建若干个副本,或者以编码的形式提供一些冗余数据.数据对象及其副本或者冗余数据往往分布在数据中心不同的节点上,因此,数据的存取效率及可靠性与数据中心中节点的结构紧密相关.数据中心网络(data center network,简称 DCN)主要研究构成数据中心中的各个节点之间的物理连接结构,即如何组织和连接数据中心中的各个服务器节点以及连接设备等,从而更加方便地为上层的各种应用和服务提供良好的接口.

依据数据中心中担任数据包的路由转发功能的节点类型,可以把数据中心网络分为 3 种:以交换机为中心的结构(switch-centric)、以服务器为中心的结构(server-centric)以及混合结构(hybrid).以交换机为中心的结构是指数据中心上的各个服务器通过交换机连接到一起,数据包的路由转发功能由交换机完成,服务器不担任数据路由转发的功能,只负责数据的存储和处理;以服务器为中心的结构是指通过为每个服务器安装多个网卡,然后通过网线把这些服务器直接连接到一起的结构.在这种结构中,没有交换机等数据转发设备,服务器不但负责数据的存储和处理,还要负责数据包的转发,担任交换机的角色;混合结构是以交换机为中心的结构和以服务器为中心的结构的一种混合,其中不但有交换机,也有部分服务器担任数据的路由转发功能.

2.1 以交换机为中心的结构

传统的企业级数据中心大都采用以交换机为中心的数据中心网络构建,这种结构采用交换机把服务器连接起来,数据包的转发功能全部由交换机承担,服务器只负责数据的存储和处理.

2.1.1 传统的树型结构

以交换机为中心的结构一般被连接成一种三层的树型结构,分别是边缘层、聚合层与核心层.一个机架的服务器一般被连接到一个机架交换机(top-of-rank switch,简称 ToR)上,构成边缘层;边缘层交换机通过一定的结构与聚合层交换机连接,完成带宽的汇聚和均衡;最后,聚合层交换机与核心层路由设备连接,为用户从外部访问数据中心提供路由.图 1 所示即为 Cisco 公司的一种典型的数据中心网络的架构示意图.

树型结构简单直观,操作方便,易于连接和实现,通过增加机架和相应的交换机能够比较方便地扩展,但也存在许多问题:(1) 链路带宽容量有限:由于带宽聚合的原因,从服务器向上层的路由器移动,需要的带宽越来越大,需要在上层使用带宽和性能更高的交换机实现.即便如此,上面的访问路由器等设备在吞吐量较大时仍然无法满足下层链路的带宽需求,因此即使下层有服务器空闲,上层链路也可能因为带宽不足而无法分配其他的服务给空闲的服务器,限制了数据中心的负载转移的能力;(2) 灵活性差、下层服务器利用率低:在此结构中,节点的 IP 地址和拓扑结构紧密相关,重新配置 IP 不方便且代价高,因此,一个服务往往被分配给单一的第 2 层域.为了保证系统能够适应失效和需求的扩充,初始时往往要在第 2 层域中预留很多的资源,导致服务器资源利用率低下;(3) 交换机资源浪费严重:机架交换机之上的交换机设备,都是通过 1:1 的设备冗余来提高系统的可靠性,

© 中国科学院软件研究所 http://www.jos.org.cn

以防止其中一个交换机失效以后连接其上的所有服务器失效.虽然这种结构提高了网络的可靠性,但因为要保证其中一个失效以后,另外一个的替换不会显著地降低系统的效率,需要让这些设备及其链路的容量提高 50%,因而导致资源的利用率极其低下;(4) 无法满足通信需求:这种结构中有限的聚合层链路无法满足下层的服务器通信需求.

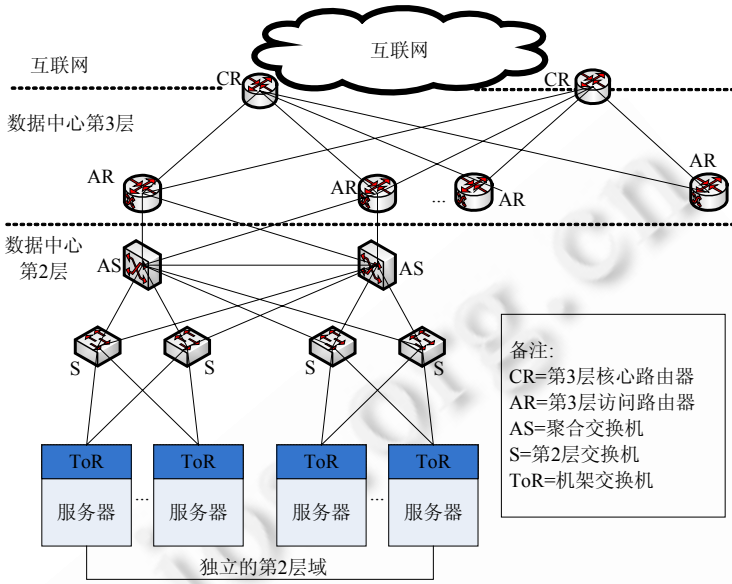


Fig.1 Traditional hierarchy tree topology of DCN

图 1 传统的分层树型 DCN 结构

2.1.2 改进的树型结构

针对上面分析的传统树型结构中聚合层链路的性能瓶颈以及资源浪费等问题,研究人员提出了几种改进的树型结构.

Fares 等人[8]在 2008 年基于树型结构提出了一种胖树型(fat tree)的数据中心网络结构.胖树结构把聚合层和核心的交换机全互连,同时把边缘层和聚合层的交换机全互连,如图 2 所示.通过这种全连接,胖树结构用冗余的普通交换机在树的上层获得了较高的带宽,解决了普通的树型结构在核心层的带宽瓶颈问题,使得每一层都具有相同的聚合带宽,而且具有良好的容错特性;但是要求核心层交换机具有较大的连接度数,降低了网络的可扩展性.胖树结构也将节点的 IP 地址和位置进行绑定,但仍无法解决数据中心的灵活性、敏捷性问题,导致数据中心的资源不能自由地分配给任意服务使用.Portland[9]的结构与胖树相似,但是通过引入 48 位的伪 MAC 地址(pseudo MAC,简称 PMAC)来标识主机的位置,分离了 IP 地址与主机位置的绑定,系统直接使用 PMAC 进行路由,与主机直接相连的出口/入口交换机负责 AMAC 与 PMAC 的报文头重写,使之对上层应用透明.但是,这种集中式的路由机制需要设置上百台的专用服务器集群提供 ARP 查询服务,严重制约了数据中心网络的灵活性,提高了运营和维护成本.

微软研究院的 Greenberg 等人提出了一种称为 VL2(virtual layer 2 networking)[10]的树型结构.VL2 扩展了传统的树型结构,通过将传统树型结构中的核心层和聚合层的交换机全连接形成 Clos[11]网络结构,解决了核心层交换机数量少、大量数据转发带来的带宽瓶颈问题,其拓扑结构如图 3 所示.同时,VL2 还使用了一种扁平式的编址方式,通过名称分离了主机的应用地址(application address,简称 AA)和定位地址(locator address,简称 LA).部署时,服务和应用能够以名称的方式部署在数据中心上的任意位置;运行时,通过映射的方式获取真正的服务器 IP 地址,提高了资源的利用率.Clos 网络在聚合层提供的冗余链路,使得 VL2 在转发数据包时,能够采用一种随机的链路选择算法让流量更加均匀地分配到各个核心交换机上.但是,在 VL2 中每个机架交换机只通过两条

© 中国科学院软件研究所 http://www.jos.org.cn

10GigE 的上行端口与聚合层交换机连接,容错性较差.对于吞吐率要求较高的应用,随着机架中主机数量的增多,链路带宽仍然可能成为性能瓶颈.另外,传统结构中有限的服务器之间的链路容量问题、资源浪费问题以及扩展性问题在 VL2 中依然存在.

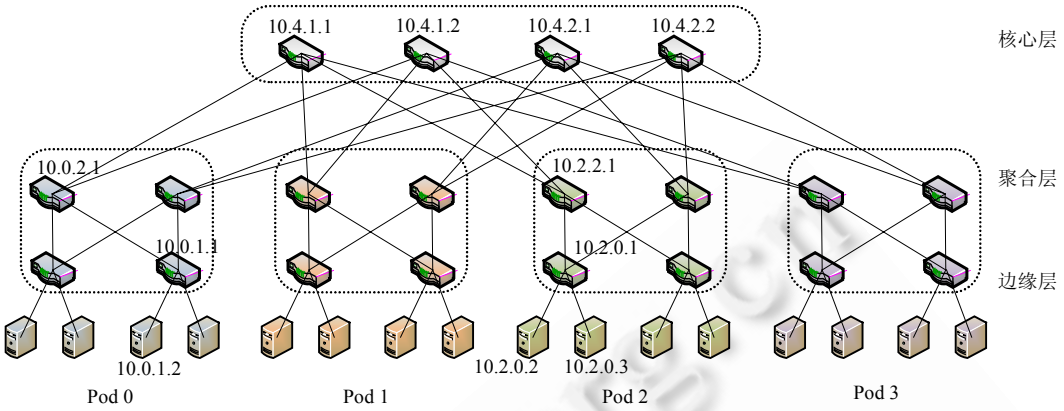


Fig.2 Example topology structure of Fat Tree

图 2 胖树型网络拓扑结构示意图

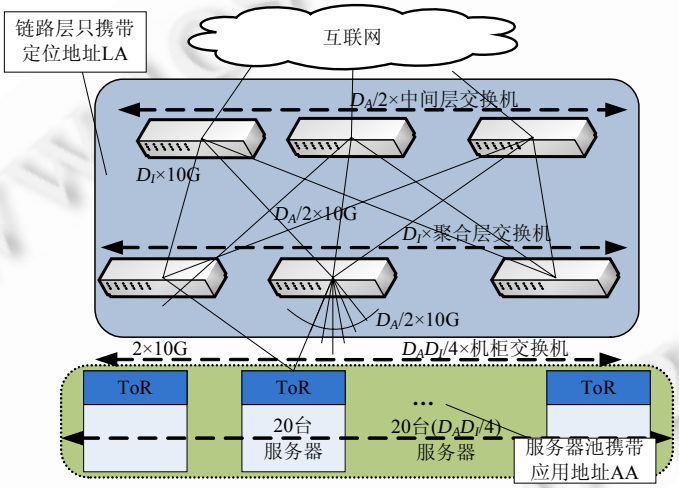


Fig.3 Example topology structure of VL2

图 3 VL2 拓扑结构示意图

2.2 以服务器为中心的结构

以服务器为中心的结构通过为每台服务器安装多个网卡实现服务器的互联,没有交换机和路由器等数据转发设备,数据转发工作全部由服务器完成.

微软研究院的 Libdeh 等人提出了一种称为 CamCube 的数据中心网络结构[12,13],CamCube 是一个纯粹由服务互相连接构成的数据中心网络结构,没有任何交换机和路由器设备.服务器之间通过多个网卡直接相连,数据包的路由转发工作完全由服务器承担.其中一个服务器节点在三维的每个方向都与其他两个服务器直接相连,构成一个三维的环状结构(3D-torus).CamCube 的坐标结构类似于 CAN[14],但在 CamCube 中,网络的物理拓扑和逻辑拓扑完全一致,其结构如图 4 所示.CamCube 向外提供了节点的坐标空间,以及距离 1 跳的邻居节点之间发送和接收数据的接口.Camcube 的设计思想是,通过更加底层、灵活的路由接口,应用可以根据需求自主开发更

© 中国科学院软件研究所 http://www.jos.org.cn

加高效的路由算法,消除因为网络的不透明性带来的性能损耗.最后的实验验证了这种设计的有效性.

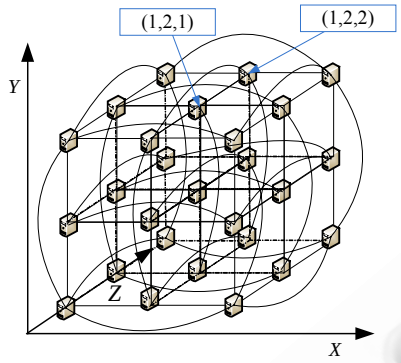


Fig.4 Example topology structure of CamCube

图 4 CamCube 拓扑结构示意图

CamCube 结构和线路连接都比较简单,但是拥有很高的链路冗余.网络中没有交换机设备,不存在树型结构中的单点瓶颈,从而可以让服务器直接与网络底层交互,上层应用可以根据需求开发更加高效的路由算法.但是,所有的数据转发工作由服务器完成,需要占用部分服务器的计算资源,对服务器的负载压力较大,会导致服务器的计算效率下降.而且每个服务器安装的网卡数目是有限的,目前最多是为每台机器安装两个网卡,每个网卡有 6 个端口(Hotlava systems. http://www.hotlavasystems.com),因此节点的规模数目有限.

CamCube 的路由路径比较长 $O(N^{1/3})$,由此带来了性能的损失和成本的增加.因此,Popa 等人提出了基于 De Bruijn 图的以服务器为中心的拓扑结构[15].De Bruijn 图的网络直径是 $O(\log N)$,因此能够有效地降低网络的路由开销.该结构中每个机架内的服务器被连接成相同的 De Bruijn 图结构,然后对机架内的节点编号,那些相同编号的不同机架内的服务器又被连接成一个 De Bruijn 图结构.他们的实验结果表明,与 CamCube 的三维环相比,De Bruijn 结构能够提高路由性能、降低成本.

2.3 混合结构

混合结构通过交换机连接服务器节点,同时又为每个服务器安装多块网卡,让服务器同时参与数据包的路由和转发功能.可以设计出更加独特和适用于特定应用场景的网络结构.

2.3.1 DCell 结构

微软亚洲研究院的 Guo 等人[16]于 2008 年提出一种以服务器为中心的数据中心网络结构 DCell.DCell 的设计目标不同于 VL2,它除了考虑网络带宽能力以外,更多地考虑了对多种错误的容错性.DCell 是一个分层的、递归定义的网络结构,高层的 DCell 网络由多个低层 DCell 网络组成,如果将第 k 层 DCell 网络看作是一个虚拟节点,那么同一层的所有节点实现全连接.最底层的 $DCell_0$ 由 n 个服务器连接到一台交换机上构成,n+1 个 $DCell_0$ 网络构成 $DCell_1$ 网络,以此类推.当 n=4 时,$DCell_1$ 拓扑结构如图 5 所示.每个 $DCell_i$ 网络都需要与同一层次其他节点相连,连接规则为节点[j,k–1]与节点[k,j]建立连接,其中,元组的第 1 个元素表示同层的 $DCell_i$ 之间的编号,第 2 个元素表示 $DCell_i$ 内某一个节点的编号,比如,[k,j]表示编号为 k 的 $DCell_i$ 中的第 j 个节点.

DCell 网络使用自上而下的方法扩展网络,即在设计规划时确定网络的层次(最大规模),然后按照从高层节点到低层节点的顺序增加服务器,每台服务器有多块网卡(3 块~4 块),网卡数即为网络层次数.第 1 层具有 n 个节点的 $DCell_k$ 的节点总数为$(n+k)!/(n-1)!$,当 n=4 时,$DCell_4$ 的节点总数为 6 720.当需要加入服务器时,DCell 网络会在每一个已有服务器上增加一块网卡,相应地扩充每一个层次的节点数目(既增加 n 的大小),并按照规则相连,构成节点规模更大的拓扑网络.

DCell 网络采用递归定义的分布式路由策略,报文在 DCell 网络中依据服务器和虚拟节点之间的连接关系进行传递,根据目的地址自行确定下一跳的路径,不需要在服务器中查询路由选择表.DCell 网络大量的冗余链

© 中国科学院软件研究所 http://www.jos.org.cn

路可以实现比树型结构更好的聚合带宽,能够较好地支持数据密集型计算一对多和多对多的通信要求.DCell网络还考虑了服务器失效、链接失效以及机架失效的情况,并依据报文绕过失效设备,通过容错路径到达目的服务器的策略,分别使用局部重路由策略、局部链接状态策略以及 Jump up 策略等解决相应的失效问题.DCell网络的路由协议实现位于第 2 层之上、第 3 层之下,可以完全兼容现有的 TCP/IP 协议及基于此类协议的应用.

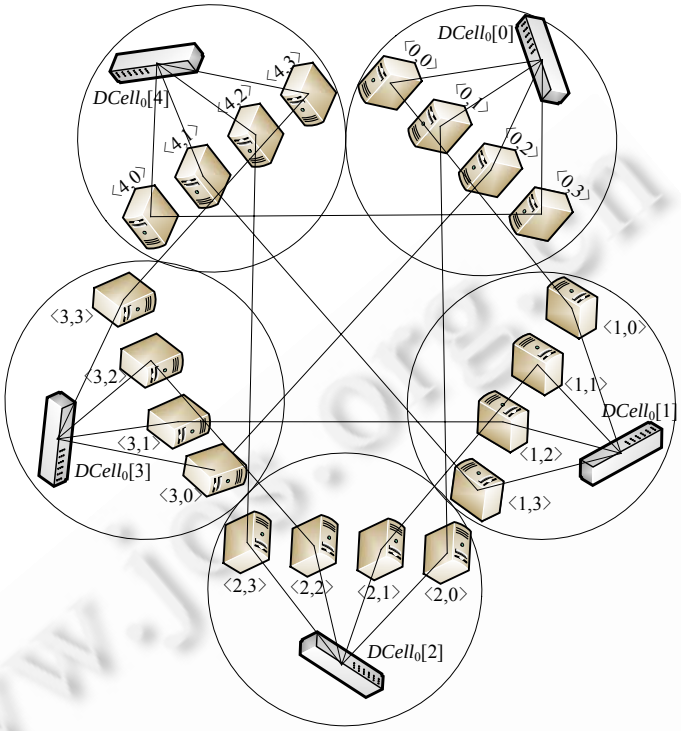


Fig.5 Example topology structure of $DCell_1$ with n=4

图 5 n=4 时的 $DCell_1$ 拓扑结构示意图

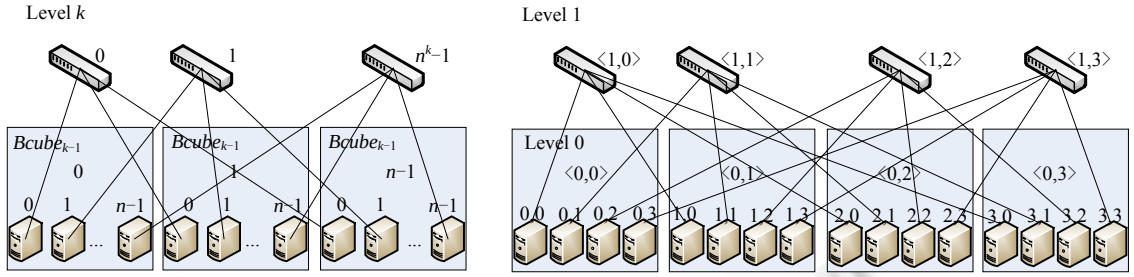
在 DCell 中拥有大量的冗余链路,每台服务器都需要安装 3~4 块网卡,虽然提高了容错性,但却增加了成本.现在的普通服务器一般拥有两个网卡端口:一个用于网络连接,另外一个备用.针对这种情况,Li 等人提出了一种 DCell 的变形结构 FiConn[17],通过部分的性能损失换取成本的降低.FiConn 不再保证 DCell 中同层虚拟节点之间的全连接结构,只利用备用端口实现同层之间是连通的,且与一层保持联通,大大降低了冗余链路的数目以及对网卡和交换机的要求,因此无需再为每个服务器安装多个网卡,高层的交换机的端口数也大为降低,节省了数据中心的构建成本.

2.3.2 BCube 结构

针对由集装箱组成的、模块化的数据中心(modular data centre,简称 MDC)应用的特点,Guo 等人在 2009 年提出了一种称为 BCube[18]的类似超立方体的数据中心网络结构,如图 6 所示.模块化数据中心把包含约有数千台规模的服务器、网络连接设备以及各种基础设施构建于一个封闭的环境中,要求部署环境空间有很高的利用率,因此对拓扑结构的物理互联特性提出很高的要求.同时,模块化数据中心网络在部署启用以后,对网络设备和服务器的修复和维护都比较困难,因此网络必须具有很高的容错性;而且要求在部分服务器和网络设备失效的情况下,网络的带宽性能下降幅度较小甚至持续保持.BCube 也是以服务器为中心的数据中心网络,采用递归的方式定义.$BCube_0$ 由 n 台服务器连接到一个 n-口的交换机组成,$BCube_1$ 由 n 个 $BCube_0$ 通过 n 个 n-口的交换机连接构成.以此类推,$BCube_k$ 由 n 个 $BCube_{k-1}$ 通过 n^k 个 n-口交换机构建而成.BCube 中任意两台主机之间都有 k+1 条并行的路径,但路径长度并不相同,BCube 实现了 one-to-x 数据传输模式线性的加速比,加速比取决于服务

© 中国科学院软件研究所 http://www.jos.org.cn

器网卡的数目.



(a) $BCube_k$ 拓扑示意图 (b) 当 n=4 时,$BCube_1$ 拓扑结构

Fig.6 Example topology structure of $BCube_k$ and $BCube_1$

图 6 $BCube_k$ 和 $BCube_1$ 拓扑结构示意图

BCube 关注集装箱内部的连接结构,但是单个集装箱的规模无法满足实际的数据中心需求,因此,Guo 等人提出了集装箱之间的连接结构 MDCube[19].MDCube 把各个由 BCube 构建的集装箱采用普通的光纤连接在一起,构建规模更加庞大的数据中心.在 MDCube 中,每个 BCube 集装箱被视为一个虚拟的节点,这些节点之间互相连接形成一个 HyperCube 结构网络.

2.4 分析与比较

在网络结构方面:以交换机为中心的结构通过把服务器连接到交换机上,构成层次式的树型结构,简单直观,连接方便;而以服务器为中心的结构需要为每个服务器安装多个网卡实现网络的互联,为了满足一些性质,构建结构更加复杂的网络拓扑;混合结构同时采用交换机和服务器提供数据的路由转发功能,因此能够构造出更加自由、灵活和适应特定应用场景的网络结构.

在路由开销方面:以交换机为中心的结构的路由开销都由交换机等专门的路由设备承担,服务器只负责数据的存储和处理,因此数据转发不会占用服务器的 CPU 资源.为了使下层节点资源得以充分利用,树型结构对上层交换机的带宽以及处理能力提出很高的要求;但是以服务器为中心的结构和混合结构都需要服务器参与数据的路由转发,因此需要占用 CPU 资源.

在可扩展性方面:以交换机为中心的结构通过增加交换机的端口以及层次,可以使得数据中心的规模不断扩大;但是在以服务器为中心的结构和混合结构中,受每个服务器安装的网卡数目的限制,数据中心的节点规模有限.

在构建成本方面:混合结构因为具有能够同时结合交换机结构和服务器结构的优点,因此在同样的性能条件下具有更低的成本[15].但是,以服务器为中心的结构的成本可能随着未来 CPU 技术或者网卡技术的发展而有所降低,比如在 CPU 中集成专门的用于网络处理的内核,或者提升网卡的处理能力使得报文不用经过 CPU 便可直接转发.

3 数据容错技术

数据容错技术是分布存储研究领域的一项关键技术,良好的容错技术不但能够提高系统的可用性和可靠性,而且能够提高数据的访问效率.数据容错技术一般都是通过增加数据冗余来实现的,以保证即使在部分数据失效以后也能够通过访问冗余数据满足需求.冗余提高了容错性,但是也增加了存储资源的消耗.因此,在保证系统容错性的同时,要尽可能地提高存储资源的利用率,以降低成本.目前,常用的容错技术主要有基于复制(replication)的容错技术和基于纠删码(erasure code)的容错技术两种.

基于复制的容错技术简单直观,易于实现和部署,但是需要为每个数据对象创建若干同样大小的副本,存储

© 中国科学院软件研究所 http://www.jos.org.cn

空间开销很大;基于纠删码的容错技术则能够把多个数据块的信息融合到较少的冗余信息中,因此能够有效地节省存储空间,但是对数据的读写操作要分别进行编码和解码操作,需要一些计算开销.当数据失效以后,基于复制的容错技术只需要从其他副本下载同样大小的数据即可进行修复;基于纠删码的技术则需要下载的数据量一般远大于失效数据大小,修复成本较高.

3.1 基于复制的容错技术

基于复制的容错技术对一个数据对象创建多个相同的数据副本,并把得到的多个副本散布到不同的存储节点上.当若干数据对象失效以后,可以通过访问其他有效的副本获取数据.基于复制的容错技术主要关注两方面的研究:(1) 数据组织结构;(2) 数据复制策略.数据组织结构主要研究大量数据对象及其副本的管理方式;数据复制策略主要研究副本的创建时机、副本的数量、副本的放置等问题.

3.1.1 数据组织结构

数据组织结构主要研究如何组织和管理大量的数据对象及其副本,是数据复制技术的一个重要研究内容.现有的组织结构主要有两种:一种是基于元数据服务器的组织结构,另一种是基于 P2P 的组织结构.

- 基于元数据服务器的组织结构

基于元数据服务器的组织结构采用统一的元数据服务器(meta-data server,简称 MDS)存储数据及其副本的元数据信息,比如副本的位置信息、版本信息、副本与数据对象之间的映射信息以及一些系统的属性、特征、状态等信息.在这种结构中,通过把管理信息存储到一个或者若干个元数据服务器上完成对数据的集中式管理.当用户访问数据时,首先与元数据服务器交互获取数据对象的位置、版本等信息,然后把数据写入到相应的位置或者从相应的位置读取数据块.

在传统的基于网格和 P2P 的分布式存储中,基于元数据服务器的副本组织结构应用广泛[20,21],它们均通过元数据服务器分离元数据的读写过程和数据的读写过程,以提高数据的容错性和读写效率.而在 P2P 应用中,为了降低分布在互联网上的各个节点访问元数据服务器的延迟,一般把网络分割成簇,然后在每个簇内构建元数据服务器,比如,Weil 和 Zhu 等人[22,23]都通过构建元数据服务器集群,把用户的访问分配给距离较近、负载较轻的元数据服务器,以提高访问效率.

在云计算环境中,基于元数据服务器的组织结构也得到了部署和应用,比如 Google 公司的 GFS[24]以及开源的 HDFS[25,26].HDFS 是开源平台 Hadoop(http://hadoop.apache.org/)的底层文件系统,实现了 GFS 的大部分功能,具有与 GFS 类似的结构,图 7 为 HDFS 的结构及工作过程示意图.

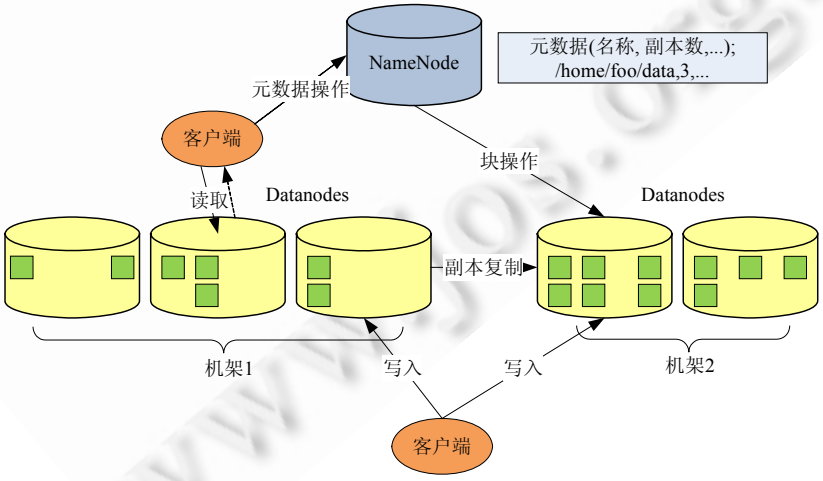


Fig.7 Architecture of HDFS

图 7 HDFS 结构示意图

© 中国科学院软件研究所 http://www.jos.org.cn

在 HDFS 中,有唯一的被称为管理节点(NameNode)的元数据服务器,负责元数据的存储和管理,数据则存放在数据节点(DataNode)上,每个数据节点定期地发送心跳信息给管理节点报告自己的状态.心跳信息中还包括其他的一些供管理节点决策的信息.HDFS 针对数据块进行复制,所以首先要把数据分割成固定大小的数据块,然后再以块为对象进行复制.数据节点通过心跳信息定期地发送自己拥有的数据块列表信息,使得管理节点能够掌握数据对象的最新分布状态.当用户读取数据时,首先通过管理节点获取数据的块列表以及每个块的副本列表及其所在的数据节点位置,并选择一个就近的数据节点读取数据;数据写操作是针对每个数据块进行的,首先通过管理节点获取需要创建的副本数目以及分配给每个副本的数据节点的位置,然后执行数据写入操作,并在数据写入完成以后把每个数据块的块列表信息以及数据块的副本信息和版本信息等记录到管理节点上.如果管理节点检测到的某个数据块的副本数目小于复制因子,就会启动复制过程以增加副本的数目.

基于元数据服务器的组织结构简单、管理方便,但是对数据的所有访问都需要通过元数据服务器,因此元数据服务器容易成为系统扩展的性能瓶颈,且存在单点失效的可能.Google 为了降低元数据服务器的负载,在 GFS 中只存储了数据块的位置映射信息和操作日志两种类型的元数据,即使这样,当元数据的访问比较频繁时,其单一的元数据服务器结构依然对性能造成了很大的影响[24,27].为了提高性能,减小单点失效的可能,改进的方案是构建由多个元数据服务器组成的元数据服务器集群.通过这种改进,能够分散单个元数据服务器的负载,减小单个服务器失效对系统的影响范围.

- 基于 P2P 的组织结构

基于 P2P 的组织结构采用比较成熟的 P2P 技术,把所有的节点按照 P2P 的方式组织,各个节点的角色是对等的,没有清晰的客户端和服务器划分.数据在存储时按照分布式哈希表(distributed Hash table,简称 DHT)的方式存储到节点上,访问时通过计算哈希值获得数据的存放位置.

传统的采用 P2P 组织结构的方法把数据的副本存放在负责数据映射关键字节点的若干个后继节点上[28],通过分布式 Hash 取消了集中的中央元数据服务器,把元数据的存储和管理分布到整个系统中.

在云计算环境中,采用 P2P 结构方式管理元数据的系统有 Amazon 公司的 Dynamo[29]和 Facebook 的 Cassandra[30].Cassandra 与 Dynamo 的结构类似,其结构如图 8 所示.Dynamo 采用一致哈希[31]的方法把数据分布到不同的节点上.一致哈希 Hash 函数的值域(哈希空间)构成一个封闭的环,通过随机地给每个节点在哈希空间上赋予一个值,Dynamo 把节点构成一个环,而这些值则表示节点在环上的位置.Dynamo 环上的每个节点负责管理自己及其前一个节点之间的哈希值空间区域,每个数据都由一个唯一的 Key 标识,当数据要插入到 Dynamo 中时,首先对 Key 进行哈希操作,得到一个哈希值(这个值在 Dynamo 环上表示一个位置),然后沿着环顺时针查找,直到找到满足节点的哈希值大于等于该数据哈希值的第 1 个节点,该节点被称为该数据的协调节点(coordinator).协调节点不但存储落在自己范围之内的数据(通过对 Key 的哈希值),而且负责对其管理的每个数据对象复制 N–1 个副本,并把这些副本存放到之后的 N–1 个后继节点上.

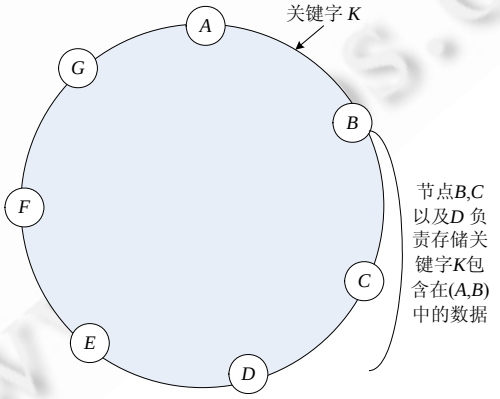


Fig.8 Ring structure of Cassandra

图 8 Cassandra 的环状结构

© 中国科学院软件研究所 http://www.jos.org.cn

基于 P2P 的组织结构不需要统一的中央服务器,解决了元数据服务器的单点失效和性能瓶颈问题.但是因为没有全局的信息作为指导,副本的放置会带来负载不均衡的问题,而且协调节点的失效会导致其负责管理的数据对象不可用.

3.1.2 数据复制方法

数据复制方法与多个因素相关,比如应用需求、网络状况、存储空间和数据访问模式等.数据复制方法对于数据的容错性和访问效率以及存储空间的利用率等至关重要,对于复制方法的研究主要包括复制策略以及副本的放置策略两个方面.

- 复制策略

复制策略主要关注创建副本的时机以及创建副本的数目,常见的复制策略可以包括静态复制策略和动态复制策略两种.

静态复制策略在数据进入时就创建指定数目的副本,然后依据副本放置策略把副本分布到节点上.在 GFS 和 HDFS 中,都是由配置参数确定副本的数目.静态复制策略简单,但却不能依据环境的变化做出动态的调整,容易造成资源浪费.

动态复制策略依据网络状况、存储空间、用户需求等动态地创建或者删除副本,在存储空间紧张时删除部分副本以节省存储空间;当存储资源丰富时,为频繁访问的数据增加副本以提高效率,均衡节点负载.动态复制策略可参考文献[32−34],Facebook 的 Cassandra 系统也通过动态复制迁移副本以均衡节点的负载.但是动态复制策略在动态创建或者迁移副本时需要执行大量额外的操作,特别是频繁的数据传输会带来巨大的网络开销.

- 放置策略

放置策略最基本的目的在于提高数据的容错性,使得用户在部分副本失效以后仍然能够通过其他的副本获得数据.但是创建的副本传输到放置节点上,需要占用大量带宽,消耗很长时间.因此,良好的放置策略不但要考虑容错性,更要考虑复制效率,使得副本能够快速地放置到节点上.基于复制的容错技术创建的多个副本,可以支持并行的数据访问,能够极大地提高数据的访问效率.传统的针对提高容错性的副本放置策略有顺序放置和随机放置策略,最新的放置策略在保证容错性的同时,针对提高副本放置的效率和访问的效率展开研究.

顺序放置策略把副本按照一定的顺序依次放置到候选节点上.这种策略的思想是:若一个放置策略可能产生的排列越多,那么当一个导致多个节点失效的随机错误发生时,就越容易使得多个副本失效.因此,如果把一个数据的所有副本按照一定的顺序放置分布到各个节点上,则产生的排列数目有限,因此在随机失效模式下就能够给存储带来较高的可靠性.顺序放置策略一般在链状的反集群结构[35]中加以应用[36],或者在一些分布式哈希表结构中得到应用[29,37,38].顺序放置策略简单而且容易实现,但在实际中,失效往往是相关的,比如网络的失效会导致整个机架不可访问,而断电则会导致整个数据中心不可访问.

随机放置策略在数据的可放置节点集合中随机地选择若干个节点,然后把副本放置到这些随机选择的节点上.现代的数据中心的副本放置大多采用这种放置策略.比如 GFS,Cassandra 等系统.从理论上来讲,随机放置能够降低关联失效对可靠性的影响,同时还能够均衡节点的负载.但是这种理论上的均衡是在节点的同构性和数据访问的同构性假设条件下得到的,在实际中,因为每个节点的存储能力不同、计算能力不同、而数据的受欢迎程度也不同,一些数据可能会更加频繁地被访问,因此这种策略并不能很好地均衡节点的负载.

为了节省副本创建和传输的时间,HDFS 的设计人员把第 2 个和第 3 个副本放置到相同的机架上.为了提高数据访问的效率,Chandy 等人[39]则把副本放置在距离用户较近的节点上,使得访问数据时能够较快地获取数据.而 Ding 等人[34]则依据用户的访问模式,对那些经常访问的数据创建较多的副本,并把副本放置到用户访问密集的区域.

3.2 基于纠删码的容错技术

基于复制的容错技术存储开销巨大,要提供冗余度为 k 的容错能力,就必须另外创建 k 个副本,存储空间的开销也增大了 k 倍.基于编码的容错技术通过对多个数据对象进行编码产生编码数据对象,进而降低完全复制带来的巨大的存储开销.RAID[40]技术中使用最广泛的 RAID5 通过把数据条带化(stripping)分布到不同的存储

© 中国科学院软件研究所 http://www.jos.org.cn

设备上以提高效率,并采用一个校验数据块使之能够容忍一个数据块的失效.但是随着节点规模和数据规模的不断扩大,只容忍一个数据块的失效已经无法满足应用的存储需求.纠删码(erasure-coding)技术[41]是一类源于信道传输的编码技术,因为能够容忍多个数据帧的丢失,被引入到分布存储领域,使得基于纠删码的容错技术成为能够容忍多个数据块同时失效的、最常用的基于编码的容错技术.

3.2.1 基于纠删码技术的容错原理

采用纠删码进行容错时,首先要把待存储的数据对象分割成若干大小相等的数据块,然后对这些数据块进行编码,得到一些编码后的编码块,读取数据时只要获得任意足够数量的编码后的数据块,就可以解码得到原始数据.用 k 表示编码前数据块的个数,b 表示每个数据块包含的比特数,k'表示一个不小于 k 的整数,n 是编码后的数据块个数,则定义纠删码为一个四元组(n,k,b,k').这个定义表示通过纠删码编码以后,用户在获得编码后的任意 k'个文件块都可以解码还原产生原文件.这个定义可简化表示为(n,k),其中,n 表示数据块和冗余块的总数,k 表示要获取的数据块的最少数目.已有很多研究在提高容错能力和降低编码复杂度等方面提出了很多编码方法,依据编码方式的不同,这些方法可以分为 Reed-Solomon 码(简称 RS 码)、奇偶阵列码(parity array code)、奇偶校验码(parity-check code)和低密度奇偶校验码(low-density parity-check code,简称 LDPC 码)等类型[42].

虽然在拥有相同容错能力的前提下,基于纠删码的容错技术的存储开销更低[43,44].但是,当数据块失效以后,基于复制的容错技术只需下载一块同样大小的数据就可以完成修复过程,而基于纠删码的容错技术则需要下载至少 k 个同样大小的数据块才能解码恢复原始数据,要占用更多的网络带宽资源,给数据中心中本来就比较紧张的带宽资源[45]带来了巨大的压力,也给数据的读取带来很大的性能损失,极大地限制了基于纠删码的容错技术的应用和推广.因此,降低基于纠删码的容错技术的带宽修复成本,成为目前针对基于纠删码的容错技术研究的一个热点问题.下面详细讨论关于纠删码修复的最新研究进展.

3.2.2 基于纠删码技术的容错修复

容错修复是指为了保证一定的冗余度,在节点失效以后重新构建冗余数据的过程.已有较多的工作针对纠删码较高的修复成本问题展开研究,根据所采用的优化方法的不同,可以把降低纠删码修复成本的方法分为两种:一种是基于度数限制的优化方法,另一种是基于网络编码的优化方法.

- 基于度数限制的优化方法

在纠删码理论中,数据块的出度定义为与该数据块有计算关系的所有冗余块的数目;冗余块的入度定义为与该冗余块有计算关系的所有数据块的数目.基于纠删码的容错技术的修复成本取决于数据块和冗余块的度数分布情况,度数越高,修复过程中参与计算的数据块和冗余块数目越大,则修复成本越高.

基于度数限制的优化方法在编码时通过限制纠删码数据块和冗余块的度数以达到降低纠删码修复成本的目的.Hafner 等人[46]提出的 WEAVER 码通过固定数据块和冗余块的度数来降低修复成本.WEAVER 码把数据块和冗余块均存放于同一节点,通过对同一节点上的数据块按照一定的规则进行异或操作得到冗余块.在计算过程中,为了降低修复成本,限制了每个数据块和冗余块的度数,在某种程度上降低了修复成本.然而,WEAVER 码的存储空间利用率并不高,为了获得较高的读取性能,往往需要用到两倍于原始数据的存储空间,存储开销较大.Gallager[47]提出了一种通过固定数据块的出度和冗余块的入度,并且限制它们之间的计算关系的低密度奇偶校验码(LDPC 码),降低了修复带宽.这种纠删码编码快、耗时少,但存储空间利用率低.而 Luby 和 Plank 等人[48,49]提出了一种基于概率密度函数来计算数据块和冗余块的度数的纠删码,在读取和修复性能上虽然有一定的优化,但结构不规则,译码过程具有概率性,不能保证译码一定成功,不易于在大规模分布存储上实现和部署.

- 基于网络编码的优化方法

网络编码是一种应用于通信领域的、融合编码和路由于一体的信息交换技术,在存储转发时,通过允许对接收的多个数据包进行编码信息融合,增加单次传输的信息量,提高网络整体性能.

Dimakis 等人[50]于 2007 年首先提出了一种称为再生码(regenerating code)的纠删码,与基于度数限制方法的纠删码不同,再生码并不限制数据块和冗余块的度数,而是通过选择特殊的编码系数来构造生成矩阵,在需要

© 中国科学院软件研究所 http://www.jos.org.cn

修复时,把存储在同一节点的多个数据块数据融合,从而降低需要传输的数据量,达到节省带宽成本的目的.

再生码的基本原理如图 9 所示,每个节点存放两个数据块,后两个节点上存放的冗余块分别由前两个节点上的数据块计算而来.当某个节点失效时,先在各个节点上进行一次组合计算,将数据融合,然后把结果加上融合过程中使用的计算系数传到修复后数据块要存储的节点.这样,修复数据块 A_1 和 A_2 只需传送 3 个块大小的数据量.如果不经计算融合而直接传送数据,则要传送 6 个块大小的数据量.

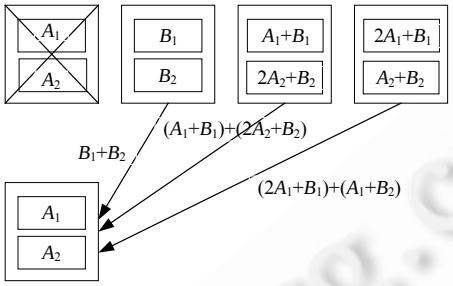


Fig.9 Repairing a (4,2)-MDS code, when node 1 fails

图 9 (4,2)-再生码修复一个失效节点的过程

再生码可以对数据块进行精确修复,但对冗余块却只能做到功能性修复.即修复后的冗余信息与原始冗余信息不一致,但可以提供同等程度的容错能力.Wu 等人[51,52]提出了确定性再生码(deterministic regenerating code),并从概率统计的角度证明了确定性再生码的存在性.确定性再生码通过有限域上的基于概率统计方法的随机系数选择,获得一组满足特定要求的系数,构造出能够精确修复冗余块的再生码.

采用网络编码的方法来降低修复成本,可以在一定程度上减少修复过程中传输的数据量.但为了满足一定的编码要求,如精确修复等,系数所在的有限域要足够大才能保证系数存在性;而且编码系数的选择方法不规则,不易于实现,目前这种编码方法仍处于理论上的探索阶段.

3.2.3 基于纠删码的容错技术的开源实现及实验平台

由于纠删码技术能够容忍多个数据帧的丢失,因此被引入到分布存储领域成为一种重要的数据容错技术.研究人员已经提出了各种类型的纠删码策略[53],同时也有许多研究人员实现了这些纠删码算法,并公布了他们的代码库,比如 Plank 等人实现的 Jerasure[54]、LUBY 实现的 Cauchy Reed-Solomon(http://www.icsi.berkeley.edu/~luby/)、Python Software Foundation 发布的 Zfec(http://pypi.python.org/pypi/zfec)以及 Partow 实现的 Reed-Solomon 码 Schifra(http://www.schifra.com/downloads.html).2009 年,Plank 等人[55]对一些常见的开源纠删码实现进行了评测和对比,他们不仅比较了各个开源的纠删码实现,而且比较了各种已有的纠删码的效率,同时还测试了各个参数对纠删码效率的影响,为研究人员在分布存储中研究基于纠删码的容错技术提供了良好的借鉴.

HDFS 虽然在最初的实现中采用的是基于复制的容错技术,但是作为具有良好结构的开源分布存储系统,为纠删码的研究和测试提供了良好的平台.微软研究院的 Zhang 等人[45]通过修改 HDFS,使其支持纠删码的容错方案.Fan 等人[56]则在 HDFS 加入一个后台进程监控数据节点上的数据块,并对那些生命周期超过一定期限的数据块,采用纠删码的容错方案替换副本方案,以节省存储空间.

3.3 分析与比较

作为分布存储中最常使用的两种容错技术,基于复制的容错技术易于实现和部署,且支持并行的访问以改善数据的读取效率,因此在实际系统中得到了广泛的应用;基于纠删码的容错技术虽然能够节省存储空间,但却需要编码和解码,计算开销较大,而且实现较为复杂,特别是容错修复需要占用较大的网络带宽,与实际应用之间尚有较大距离.另外,分布式存储作为云计算环境下比较基础的服务,需要为各种应用提供数据存取服务,不同的应用具有不同的特性,对容错的要求各异,因此各种应用在容错技术以及各种参数的选择方面也有很大的不同.本节首先对两种技术进行对比,然后介绍各种实际应用对容错的影响.

© 中国科学院软件研究所 http://www.jos.org.cn

3.3.1 技术比较

一般认为,基于纠删码的容错技术能够节省存储空间,提高容错能力.假定原始数据的大小为 n MB,被分割成 n 块大小为 1MB 的数据块(block),然后再对每个块进行复制和编码.纠删码的编码参数为(n,k),即编码后的数据块为 n 块,每一块的大小仍然为 1 MB,获取其中的任意 k 个子块可以得到原始的大小为 k MB 的数据块.则此纠删码在不考虑其他因素的情况下,能够在 $n–k$ 个数据块失效时仍然保持数据的可用性;而为了拥有相等的容错能力,复制技术需要为每个数据块创建 $n–k$+1 个副本(包括原始数据块).表 1 给出了这种情况下基于复制的技术与纠删码技术的比较,从表中可以看出,纠删码在容错能力和存储空间上占有绝对优势.

但是,这种理论上的理想状况在实际环境中很难达到,因为在实际的分布存储中采用基于纠删码的容错技术,还需要考虑各种特定的应用背景和需求,例如数据的访问模式、节点的负载均衡、失效修复等情况.Rodrigues 等人[44]在 PlanetLab,Overnet, Farsite 等多个平台下的实验模拟的研究结果表明,纠删码的优势并不是在每个平台上都能够发挥出来的,在某些特殊的情况下,其效果还比不上基于复制的容错技术.Lin 等人[57]经过深入的研究发现,纠删码的优势并不如想象的那么明显,在节点可用性很低的情况下,纠删码的成本甚至要高于对整个文件进行复制的成本.基于纠删码的容错技术还有一些内在的缺陷,比如:在下载延迟上受限于 k'个数据块中的最近副本的最大延迟,而基于复制的技术则只需下载最近的副本;纠删码也无法直接读取下载数据块的其中一个子块,要获取某一个子块,必须下载多个数据块,再经解码得到相应的子块;对于在服务器端的一些诸如关键字搜索、内容查找等操作,也是基于纠删码的容错技术所无法满足的.

Table 1 Comparison between replication and erasure code

表 1 基于复制的容错技术与基于纠删码的容错技术对比

	副本技术	纠删码技术
存储空间	$k(n–k+1)$ MB	n MB
修复带宽	1 MB	k MB
容错能力	$(n–k)$ 块	$(n–k)$ 块

3.3.2 应用分析

云计算环境下的分布式存储为各种分布式应用向上提供数据存取服务[4],比如 Google 的 Web 搜索应用,Amazon 的电子商务应用以及微软的 Hotmail 邮件服务等,这些应用的特性各异,对容错的要求也不尽相同.

基于复制的容错技术的实现简单、易于部署,可以提供更高的访问效率,在 Web 搜索、电子商务、在线社交网络等领域得到了普遍应用,比如在 Google 公司的 GFS[24]、Amazon 公司的 Dynamo[29]和 Facebook 的 Cassandra[30]以及 Hadoop 的 HDFS[26]中都采用基于复制的容错技术提高系统的容错性;基于纠删码的容错技术实现复杂,修复成本较高,因此在实际的分布存储中应用较少,Weatherspoon 等人[43]在基于 P2P 的分布存储系统 OceanStore 上采用了基于纠删码的容错技术,以实现对归档数据进行容错,节省存储空间.Fan 等人[56]通过对雅虎 M45 集群应用 7 个月的追踪观察发现,大多数的数据访问操作发生在数据创建后的较短的一段时间内,因此他们修改了 HDFS,使其通过一个后台进程监控写入的数据块,当数据块被写入一段时间后通过用编码块替换副本块,采用基于纠删码的容错技术替换基于复制的容错技术,以节省存储空间,并在此基础上测试了延迟编码的时间与带来的性能损耗之间的关系.其结果表明,当延迟时间大于 1 个小时以后,性能的损耗几乎可以忽略不计,此时采用基于纠删码的容错技术能够有效地降低存储开销,而延迟带来的磁盘临时额外开销仅为 12%左右.

当采用基于复制的容错技术时,不同的应用也有不同的选择.在数据的组织方式方面,Google 的 GFS[24]采用元数据服务器的方式组织和管理大量的 Web 搜索数据,而 Amazon 的电子商务应用[29]和 Facebook 的社交网站应用[30]中存储的多是键-值对数据,因此他们均采用一致哈希的方式组织数据以获得更高的效率;在冗余块的大小设置方面,Google 的 GFS 选择了较大的 64MB 的数据块,这样可以减小数据块的数量,进而减小其初始设计时单一元数据服务器的负载.

为了消除应用的相关性,Kossmann 等人[58]提出了一种灵活的可配置的模块化分布存储系统 Cloudy,通过采用一种通用的 DPI 数据模型表示数据,使得用户能够根据自身的需求修改模块和参数,使之适应特定的应用场

© 中国科学院软件研究所 http://www.jos.org.cn

景.但是 Cloudy 仍然不能解决所有问题,不同的应用仍需针对应用特性研究相关的技术,开发不同的模块.

4 节能技术

在云计算环境下,构成分布存储的底层数据中心规模庞大,也使得分布存储的成本非常高昂.成本不仅包括硬件设施的购买成本,还包括 IT 设备的电能消耗、制冷设备及其电能消耗等其他开销.为了应对不断增大的能耗,Microsoft 公司和 Google 公司构建的新的数据中心甚至不得不选择距离发电站较近的地方(http://www.gorgebusiness.com/2005/google.htm).不断增长的能耗开销不但增加了 IT 系统的运行成本,更加剧了“温室效应”,给人类居住生活的环境造成了巨大破坏.因此,不论从云计算提供商降低成本以追求更高利润的角度,还是从降低能耗以保护环境的角度来看,降低能耗都是云计算领域的一个研究热点.

数据存储是云计算的重要组成部分,是各种云计算服务的基础,在云计算的整个能耗组成中占有很大比例,一些大规模数据中心上的存储系统的能耗占到整个数据中心能耗的 27%~40%[59,60].因此,在云计算环境下,研究分布存储中的节能技术具有很大的现实意义和应用前景.

4.1 能耗模型

云计算环境下的分布存储应用一般部署在大规模的数据中心上,要节省数据中心的能耗,最直接的方法就是降低单个计算机节点的能耗,但是降低能耗往往伴随着性能的损失.为了更好地研究性能和能耗的关系,需要建立单个计算机的能耗模型.现有的单个计算机的能耗模型有两种:一种是比例模型,另一种是两段模型.

比例模型简单直观,认为能耗与计算机上的硬件设备(CPU、磁盘、交换机等)的利用率成正比,当设备空闲时其能耗可以忽略[61].比例模型并不能准确地描述设备的能耗情况,实际上,只要计算机处于开机状态,就会消耗一定的电能.动态频率和电压调整(dynamic frequency and voltage scaling,简称 DVFS)方法[62,63]是一种典型的受比例模型驱动的能耗节约方法,它动态地调整 CPU 的频率和电压,使其能够动态地适应负载的变化.但是,CPU 的能耗在整个计算机的能耗中只占 25%[64],在以数据为中心的分布式存储中,CPU 的能耗比例更低[65],因此,降低计算机的能耗需要考虑各个设备组件.Google 的两位工程师提出与能耗成比例的计算[66],2011 年,韩国的研究小组[67]从视频应用出发,研究了固态盘的管理方式,使磁盘能够在不同的负载下按照不同的转速运行,进而实现能耗和性能成比例,以降低能耗.

两段模型认为,计算机的能耗由两部分组成:固定能耗和可变能耗.固定能耗由风扇、机械驱动、二极管以及其他一些只要计算机打开就运行的设备的能耗构成;可变能耗则随着 CPU,磁盘等运行速度的提高而增加.比例模型和两段模型都认为,计算机能耗的峰值一般出现在其负载很高的时候.但是比例模型认为,计算机空闲时的能耗很低甚至可以忽略,而两段模型则认为,计算机空闲时的负载是不可忽略的.Rivoire 的研究[68]显示,一个 8 核的 Xenon 处理器在空闲时的能耗达到了所有核都运转时的 60%,说明两段模型能够更加准确地表示计算机的能耗情况.两段模型说明,DVFS 等使得计算机处于空闲状态的技术并不能消除无用的能耗,动态的机器启动和挂起(vary on vary off,简称 VOVO)[69]技术则通过动态控制的方法使得节点上的部分组件在没有任务的情况下关闭,达到降低单个计算机能耗的目的;或者把数据中心上的部分节点在没有任务的时候挂起,达到降低整个数据中心能耗的目的[70].

4.2 技术分类

目前,针对如何减少分布存储中的能源消耗问题,已产生了不少研究成果.从软、硬件的角度来分,可将现有技术分为硬件节能技术和软件节能技术[59].硬件节能技术主要通过降低构成分布存储的硬件设备的能耗,以达到降低分布存储能耗的目的;软件方法则通过对一些软件的使用而对存储等资源进行有效的调度,以达到降低分布存储能耗的目的.

4.2.1 硬件节能技术

按照关注的硬件层次区分,硬件节能技术可以分为两种:一种是从构成分布存储的各个计算机部件的角度出发,通过采用新的体系结构或者硬件技术,以降低单个计算机节点以及整个分布存储的能耗,如基于 ARM 体

© 中国科学院软件研究所 http://www.jos.org.cn

系结构的低能耗 CPU、支持随机访问的大容量闪存硬盘(flash disk)等;另一种是从数据中心的角度出发,用低能耗、低性能的硬件设备替换高能耗、高性能的硬件设备来构建数据中心,以达到降低分布存储能耗的目的.

- 降低硬件设备能耗

硬件技术的改进主要通过新的体系结构,比如CPU和磁盘的体系结构,以降低组件的能耗.硬件体系结构的研究一直是硬件研究和设计人员关注的重点,这里简单介绍若干与能耗相关的主要研究.

Gurumurthi 等人[71]认为,磁盘消耗的能耗遵守比例模型,其消耗的电能随着转速的加快而增加.基于此想法提出了一种具有多级转速的磁盘结构,使得磁盘在低负载状态以较低的转速运行,高负载时以较高的转速运行.Hamilton[72]提出了一种基于 Athlon 处理器的低能耗服务器机架结构.

- 降低数据中心能耗

近年来,低能耗芯片的发展异常迅猛,但是这些芯片和市场上的主流芯片相比性能较低.一个自然的想法是研究如何把这些性能较低、能耗也较低的芯片应用到数据中心上,以在单位能耗上产生更高的性能.

CMU 的一个小组[73]采用 500MHZ 的处理器和快速闪存盘构建集群,并在其上测试单位焦耳内执行的查询数.结果发现,采用低能耗节点的集群的能耗有效性要比普通机器构建的集群能耗有效性高 6 倍.韩国的一个研究小组[74]采用低功耗的组件搭建了一个 Hadoop 平台,虽然性能略有降低,但其结果表明,这种模式使得能耗节省了 113 倍.为了防止性能的大幅下降,他们在构建时加入了部分通用节点,以便能够在必要时把数据转移到这些通用节点上以满足任务的性能需求.

在数据密集型的分布存储应用中,数据吞吐量很大,但是磁盘操作的速度远远低于 CPU 的处理速度.根据 Amadal 定律,数据的存储访问会成为整个系统的瓶颈.针对这个问题,Szalay 等人[75]采用能耗较低、效率较低的 CPU 和访存效率较高的固态磁盘构建数据中心以提高能耗的有效性.他们的实验结果表明,这样的搭配方式能够有效地提高数据中心的效率,而能耗则基本保持不变,从而提高了能耗的有效性.

Lim 等人[76]将低能耗机器应用于面向 Internet 服务的数据中心内,在研究和分析了其性能参数的基础上,提出了一种专为数据中心计算环境而设计的服务器体系结构.该结构使用低能耗机器构成大规模数据中心,以更少的能耗提供相同的服务能力.

4.2.2 软件节能技术

软件节能技术通过一定的软件策略,在很少的性能损失,甚至不影响性能的前提下,使数据中心内的部分节点进入低能耗模式或者被挂起状态,达到降低整个存储数据中心能耗的目的.目前的软件节能技术主要关注两方面的研究:一个是节点管理技术,另一个是数据管理技术.

- 节点管理技术

节点管理技术主要研究如何选择分布存储中的部分节点(或者磁盘)为应用提供服务,并让其他节点进入低能耗模式或者关闭状态,从而达到降低能耗的目的.

磁盘的能耗更符合比例模型,因为其主要能耗来源于磁盘的电机,当磁盘在执行高速的数据存取访问时,消耗的电能远高于磁盘在空闲状态时的能耗,与转速的平方成正比[77].Gurumurthi 等人[71]提出的磁盘动态转速(dynamic rotations per minute,简称 DPRM)策略使得磁盘可以在不同的访问频率下以不同的转速运转,从而在满足性能要求的同时尽可能地降低磁盘的能耗.云计算环境下的分布存储拥有大量的存储节点,每个节点上都有若干磁盘.在这个环境下,每个节点的数据访问频率不同,为能耗提供了很大的优化空间.Harnik 等人[78]针对实际系统的研究发现,数据访问具有很强的周期性,在一天中某一大片的时间(比如夜晚)数据内存访问的频率较小,此时,系统的大部分节点处于空闲状态,因此可以在这些空闲时间关闭大量空闲节点以节省能耗.

节点管理技术选择一些数据访问频率较低或者空闲状态的节点使其进入低能耗或者挂起状态.为了保证数据的可用性、容错性以及效率等方面的需求,要求未被关闭的活跃节点(或者称其为存活节点)上存储所有数据对象的至少一个副本.因此,需要在所有的存储节点集合中找到一个子集,并使其覆盖所有的数据,这个问题被称为完全覆盖(full coverage)问题.完全覆盖问题可以用图论上的二部图或者超图来抽象描述[78],理论证明,这是一个非常困难的问题,比如对于基于复制的容错技术来讲,当数据的冗余度 d=2 时,完全覆盖问题是一个 NP

© 中国科学院软件研究所 http://www.jos.org.cn

完全问题.对于更一般意义上的 d,查找完全覆盖将更加难以实现,只能通过启发式或者近似的算法来实现.

完全覆盖查找策略和原始的数据放置策略紧密相关,因此,不同的数据放置策略可能需要不同的完全覆盖查找策略.Pinheiro 等人[79]把数据副本分为主副本和从副本,并让主副本集中到存储节点的一个子集中,而从副本分布到其他的不同节点上,从而可以让存储主副本的节点构成覆盖子集.但是这种策略需要修改原始的数据放置策略,并且不能很好地利用冗余带来的负载均衡和效率优势.更一般的完全覆盖查找策略有随机查找策略和启发式查找策略:(1) 随机查找策略:这种策略随机地选取若干个子集,并在这些子集中选择拥有最好覆盖的一个[78].随机查找策略简单,容易实现,但是找到的覆盖集合可能不是最优的,甚至可能没有覆盖所有的数据对象;(2) 启发式查找策略:这种策略通过多次循环往一个子集中不断地加入一个节点,在每一次的循环中按照某种贪心策略加入当前的最佳候选节点[78,80].

有时,为了保证数据的可用性和容错性,可能没有满足条件的完全覆盖.因此,IBM 实验室的 Harnik 等人[78]引入辅助节点用来集中存储那些没有被覆盖的被关闭节点上的数据对象,以找到一个更小的完全覆盖集合,从而节省更多的能耗.Pinheiro 等人[79]让那些访问频繁的数据所在的存储节点优先进入覆盖集合,以保证这些频繁访问的数据能够被尽可能地高效访问,从而提高低功耗模式下的数据访问效率.

- 数据管理技术

节点管理中被关闭节点的选择与数据的管理技术紧密相关,目前已有的数据管理技术可以分为 3 种:基于静态数据放置的数据管理技术、基于动态数据放置的数据管理技术和基于缓存预取的数据管理技术.

基于静态数据放置的数据管理技术[81−87]根据一定的数据放置策略将数据散布到系统中各个磁盘节点上以后就不再改变其分布位置,它利用系统中的冗余磁盘(如存放数据副本或纠删码冗余信息的磁盘),在保证一定容错性能的前提下,使得在一定的时间内一部分磁盘不向外提供数据存取服务,进而可以关闭或者挂起该部分磁盘,以达到节省能耗的目的.比如,斯坦福大学的研究小组[87]发现,Hadoop 能耗效率较低的主要原因是其节点的大量空闲运转时间,因此他们通过改善副本的放置策略来使得其中的一些空闲节点得以关闭,从而节省能耗.其实验结果显示,这种策略能够带来 9%~50%的能耗优化.Pinheiro 等人[86]通过把原始数据和冗余数据分开存放到不同的存储节点上,从而使得对数据的访问集中到原始数据所在的存储节点上,因此可以把冗余节点关闭以节省能耗.基于静态数据放置的数据管理技术将数据一次性散布到系统中,在后续阶段不会进行数据迁移.但其灵活性差,数据进入系统时不合理地放置策略可能导致节点负载不均衡、利用率低下等问题.

基于动态数据放置的数据管理技术[88−92]根据数据访问模式或访问频度动态调整数据存放的位置,通过把访问频度高的数据迁移到部分磁盘上,使得其余部分磁盘在较长的时间内没有任何访问请求,从而可以使其进入低能耗状态,达到降低能耗的目的.基于动态数据放置的数据管理技术可以根据数据的访问频度等统计特性动态调整数据的布局,其灵活性高.但缺点在于,数据迁移过程会占用大量的网络带宽,特别是在数据密集型应用领域,移动数据成本较大.而且采用 VOVO 方法把一些节点挂起以后,如果存储在这些节点上的数据是唯一副本,则会导致数据不可访问.比如,瑞士的一个研究小组[92]在 Hadoop 的基础上提出了一种任务的调度与数据的存放位置相互感知的方法,使得任务的调度和数据的存放节点相互依赖,从而避免了把存放将要访问的数据节点挂起,或者调度那些数据不可访问的任务,因此避免了性能的损失,达到休眠部分空闲 Hadoop 节点的目的.

基于缓存预取的数据管理技术[93]采用内存中的数据缓存思想,通过统计磁盘中数据的访问频率等特征,采用一定的统计预测模型,提前将磁盘中的数据取到内存或其他低能耗辅助存储设备上,以此将数据访问转移到内存或低能耗辅助存储设备上,使原磁盘进入低能耗模式.如 Zhu 等人[93]提出的一种能耗敏感的缓存替换算法,针对数据读取操作进行优化,将近期频繁读取的数据预取到内存中以减少对磁盘的数据访问频率,使其进入低能耗模式.缓存预取技术需要对数据的访问信息进行统计和分析,而特定的应用对数据的访问模式各异,而且数据的预取很少能够针对数据读操作和写操作同时进行优化,因此其适用范围较小.在分布存储中,即使频繁访问的数据也具有很大的规模,内存空间可能无法满足缓存的需要.因此,一些低能耗的存储设备,比如固态硬盘(solid state disk,简称 SSD)可以作为缓存使用,比如 Szalay 等人[75]设计的基于固态盘的模型,而 Harnik 等人[78]的辅助节点上的磁盘就可以用能耗更加有效的固态磁盘来代替.

© 中国科学院软件研究所 http://www.jos.org.cn

4.3 典型案例

Hadoop 是一个广受欢迎的开源分布式计算平台,但其本身并没有降低能耗的模块,因此有很多关于改进 Hadoop 能耗的研究工作.Standford 的研究小组[87]首次针对 Hadoop 的能耗问题展开研究,指出 Hadoop 能耗效率较低的主要原因是其中有大量节点长时间运转在空闲状态.他们同时设计了一种采用基于静态数据放置的数据管理技术选取节点,使得部分节点进入低能耗状态,以达到节省能耗的目的.而 Berkeley 大学的研究小组[94,95]则通过对实际系统的追踪分析了 Hadoop 的低能耗有效性,并提出了一整套基于能耗测量的框架,为 Hadoop 上的能耗研究提供了很好的基础.他们通过调整 Hadoop 的参数来改进其能耗有效性,并通过重放工作过程来验证他们的方法.Oppenheim[96]通过修改 Hadoop 平台,在其中引入一个能耗管理模块以关闭部分空闲节点,从而节省 Hadoop 的能耗.这些工作采用的都是软件节能技术,通过修改数据的放置策略,从而把数据的读写集中到部分节点上,进而关闭其余的空闲节点以达到降低能耗的目的.Hadoop 为分布存储的节能研究提供了一个良好的实验平台.

微软剑桥研究院的研究小组设计开发了一个能耗有效(energy efficient)的分布式存储系统 Sierra[97].Sierra 具有与 GFS 类似的架构,但是拥有一个独立的磁盘(节点)调度模块(gear schedule service),用于启动或者关闭磁盘.Sierra 系统采用了如下几种软件节能技术以节省能耗:(1) 通过能耗感知的基于静态数据放置的数据管理技术,以保证即使有节点被关闭,也能够满足每个对象在任何时刻都是可用的;(2) 通过预测调度机制挖掘数据访问的高峰期和空闲期,以启动或者关闭部分节点;(3) 通过基于动态数据放置的主动的数据迁移技术,以保证数据在任意时刻都是可用的;(4) 通过短期的版本控制仓库,以保证数据的写操作在任意时刻都是一致的、可行的.Sierra 通过对 Windows Hotmail,Windows Messager 以及运行于剑桥的一个文件下载服务器等实际运行系统的数据访问模式进行跟踪统计,发现数据的访问具有明显的周期性,每天的数据访问都有基本固定的高峰期和空闲期,因此在空闲期关闭部分节点以节省能耗.他们在 Hotmail 上重现访问请求,测试结果发现,Sierra 能够节省 23%的能耗.

4.4 分析与比较

降低硬件设备能耗的优点是节能效果好,不需要复杂的管理组件;但是需要新的体系结构和硬件设计,灵活性差,不能根据需求动态调整.对于已经大规模部署的实际应用系统而言,大批量地替换低能耗硬件成本过高,因此可推广性较差.而采用低功耗的设备构建数据中心则比较灵活,从能耗有效性的角度出发,在保持性能的同时尽可能地降低数据中心整体的能耗,为数据中心的构建提供了有益的指导.

软件节能技术是目前云计算环境下分布存储节能技术的研究热点,软件节能技术不需要改变现有的硬件设施,采用一定的节点管理技术和数据管理技术,通过改变数据的放置位置或者数据预取和缓存技术,创造时机让部分磁盘节点进入低能耗模式或关闭部分磁盘节点来达到节能的目的.与基于硬件方法的节能技术相比,其可推广性强、灵活性好、变更成本小,因此得到了更加广泛的关注和研究.但是,现有的研究大多是针对基于复制的容错技术而展开,在基于纠删码的容错存储方案下,关于节能的研究仍然很少.

5 未来研究展望

综合云计算环境下数据分布存储面临的挑战、研究热点以及应用前景,结合我们目前的研究工作,认为未来的研究可以从以下几个方面展开:

(1) 针对存储应用进行优化的数据中心网络结构与路由策略的研究

不同类型的应用具有不同的特点,数据的访问模式也随着应用的不同有很大的变化.针对数据中心上的一些键-值对访问频繁的应用特点,Libdeh 等人[12]提出了最新的 CamCube 结构,该结构对键-值访问进行了路由优化.未来的工作可以针对存储应用的访问模式和特点,设计一种能够有效组织和管理数据的数据中心网络拓扑结构及相应的路由策略,使之能够有效地均衡链路负载,提高系统的吞吐率.例如,通过对实际部署的系统进行统计,挖掘系统的数据访问模式,然后针对具体的访问模式,优化底层的数据中心网络拓扑,把频繁访问的数据

© 中国科学院软件研究所 http://www.jos.org.cn

存放到网络性能更好的节点,使得频繁访问的数据能够得到更加高效的访问效率.

(2) 降低数据中心网络成本的研究

随着数据中心规模的扩大和数据量的增加,数据中心的构建成本越来越高.硬件工艺和技术的不断进展,使得采用低端交换机替代高端的交换机,以降低数据中心网络的构建成本成为可能.为了提高吞吐率和性能,在构建数据中心网络时往往采用了大量的冗余设备,带来了大量的能耗开销,可以通过关闭部分不用的交换机以降低数据中心网络的能耗成本.通过对数据中心分布存储应用的数据访问模式及流量特征的研究,使得数据中心网络中一些冗余的链路和交换机能够在不影响效率和容错性能的情况下被关闭,达到优化数据中心网络成本的目的.

(3) 基于纠删码的数据放置技术的研究

目前,云计算环境下分布存储应用中的数据放置策略都比较简单,比如机架无关(rack unware)、机架相关(rack aware)、数据中心相关(datacenter aware)等策略,或者顺序放置、随机放置等策略.这些策略大都针对基于复制的容错技术,实际上,不同放置策略对基于纠删码的容错技术的容错性以及访问效率同样有很大的影响,但是,现有的工作很少有针对基于纠删码的数据放置策略的研究.针对基于纠删码的数据放置策略展开研究,设计结合数据中心网络结构特征和具体应用特点的数据块放置策略.

(4) 基于纠删码的节能技术研究

软件节能技术是目前云计算环境下分布存储的一个研究热点.基于磁盘管理的技术取得了一定的成果,但是这些技术仍然存在缺陷.而且,现有的研究工作都是针对基于复制的容错技术展开的.基于纠删码的容错技术有其自身的特点,未来的工作可以针对纠删码数据块放置的特点,在降低动态数据放置技术的数据迁移成本以及提高静态技术的负载均衡能力方面进行更加深入的研究,以提出节能效果更加明显的数据放置技术.

(5) 分布存储系统的开发与部署测试研究

目前,部署并应用的分布存储系统主要有各大企业的云计算存储平台,比如 Google 的 GFS、Amzon 的 S3 等.但是,在学术界影响最大的是开源的 Hadoop 系统,它在底层为分布存储研究提供了一个良好的平台.比如,张哲等人[45]以及 Fan 等人[56]就是在 Hadoop 的基础上,针对基于纠删码的容错技术在分布存储中的作用展开研究.在理论研究的基础上部署一个实际的分布存储系统,为理论和算法研究提供验证是必须的.通过对系统的观察和测试,不但可以发现研究中存在的问题,而且可以挖掘系统的运行特征,促进更进一步的理论研究.

6 总 结

云计算作为下一代的计算模式,在科学计算以及商业计算等众多领域应用广泛.数据中心作为云计算的基础,需要解决海量数据环境下分布存储在可扩展性、容错性和低成本等方面的问题.为此,需要深入研究数据中心物理网络拓扑的构建技术,提高数据容错性的技术以及降低能耗的各种节能技术等.在云计算环境下构建一个拥有良好的可扩展性、容错性和低成本的分布存储数据中心,涉及到各种技术方法,这些技术是目前分布存储领域内的研究热点,但与实际的部署应用之间仍然存在很大的距离.

本文研究了在云计算环境下构建分布存储面临的挑战,以及解决这些挑战需要的各种关键技术,综述了这些关键技术的最新研究进展,对各项技术依据不同的标准进行了分类,并在分类的基础上,分析对比了一些相关的技术方法,指出了这些技术存在的问题.最后展望了未来研究的方向.

References:

[1] Kouzes RT, Anderson GA, Elbert ST, Gorton I, Gracio DK. The changing paradigm of data-intensive computing. Computer, 2009, 42(1):26−34. [doi: 10.1109/MC.2009.26]

[2] Moore R, Prince TA, Ellisman M. Data-Intensive computing and digital libraries. Communications of the ACM, 1998,41(11):56−62. [doi: 10.1145/287831.287840]

[3] U.S. Department of Commerce. The NIST definition of cloud computing. National Institute of Standards and Technology, 2011.

© 中国科学院软件研究所 http://www.jos.org.cn

[4] Chen K, Zheng WM. Cloud computing: System instances and current research. Journal of Software, 2009,20(5):1337−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[5] Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In: Proc. of the GCE 2008. Austin: IEEE, 2008. 1−10. [doi: 10.1109/GCE.2008.4738445]

[6] Markoff J. Sun and IBM to Offer New Class of High-End Servers. New York Times, 2007. http://www.nytimes.com/2007/04/26/technology/26compute.html

[7] Dean J. Experiences with MapReduce, an abstraction for large-scale computation. In: Proc. of the PACT 2006. Seattle: ACM Press, 2006. 16−20. [doi: 10.1145/1152154.1152155]

[8] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: Proc. of the SIGCOMM 2008. Seattle: ACM Press, 2008. 63−74. [doi: 10.1145/1402958.1402967]

[9] Mysore RN, Pamboris A, Farrington N, Huang N, Miri P, Radhakrishnan S, Subramanya V, Vahdat AA. PortLand: A scalable fault-tolerant layer 2 data center network fabric. In: Proc. of the SIGCOMM 2009. Barcelona: ACM Press, 2009. 39−50. [doi: 10.1145/1592568.1592575]

[10] Greenberg A, Hamilton JR, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S. VL2: A scalable and flexible data center network. In: Proc. of the SIGCOMM 2009. 2009. 51−62. http://research.microsoft.com/pubs/80693/vl2-sigcomm09-final.pdf [doi: 10.1145/1592568.1592576]

[11] Dally WJ, Towles BP. Principles and Practices of Interconnection Networks. San Francisco: Morgan Kaufmann Publishers, 2004. 550.

[12] Abu-Libdeh H, Costa P, Rowstron A, O'Shea G, Donnelly A. Symbiotic routing in future data centers. In: Proc. of the SIGCOMM 2010. New Delhi: ACM Press, 2010. 51−62. [doi: 10.1145/1851182.1851191]

[13] Costa P, Zahn T, Rowstron A, O'shea G, Schubert S. Why should we integrate services, servers, and networking in a data center? In: Proc. of the WREN 2009. Barcelona: ACM Press, 2009. 111−118. [doi: 10.1145/1592681.1592699]

[14] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of the SIGCOMM 2001. San Diego: ACM Press, 2001. 161−172. [doi: 10.1145/383059.383072]

[15] Popa L, Ratnasamy S, Iannaccone G, Krishnamurthy A, Stoica I. A cost comparison of data-center network architectures. In: Proc. of the ACM CoNEXT 2010. Philadelphia: ACM Press, 2010. [doi: 10.1145/1921168.1921189]

[16] Guo CX, Wu HT, Tan K, Shi L, Zhang YG, Lu SW. DCell: A scalable and fault-tolerant network structure for data centers. In: Proc. of the SIGCOMM 2008. Seattle: ACM Press, 2008. 75−86. [doi: 10.1145/1402958.1402968]

[17] Li D, Guo CX, Wu HT, Tan K, Zhang YG, Lu SW. FiConn: Using backup port for server interconnection in data centers. In: Proc. of the INFOCOM 2009. Rio de Janeiro: IEEE, 2009. 2276−2285. [doi: 10.1109/INFCOM.2009.5062153]

[18] Guo CX, Lu GH, Li D, Wu HT, Zhang X, Shi YF, Tian C, Zhang YG, Lu SW. BCube: A high performance, server-centric network architecture for modular data centers. In: Proc. of the SIGCOMM 2009. ACM Press, 2009. 63−74. http://research.microsoft.com/pubs/80693/vl2-sigcomm09-final.pdf [doi: 10.1145/1592568.1592577]

[19] Wu HT, Lu GH, Li D, Guo CX, Zhang YG. MDCube: A high performance network structure for modular data center interconnection. In: Proc. of the CoNEXT 2009. Rome: ACM Press, 2009. 25−36. [doi: 10.1145/1658939.1658943]

[20] Stockinger H, Rana OF, Moore R, Merzky A. Data management for grid environments. In: Hertzberger B, Hoekstra A, Williams R, eds. Proc. of the European High Performance Computing and Networks Conf. Amsterdam: Springer-Verlag, 2001. 151−160. [doi: 10.1007/3-540-48228-8_16]

[21] Guy L, Kunszt P, Laure E, Stockinger H, Stockinger K. Replica management in data grids. In: European Organization for Nuclear Research. Research Report, 2002. http://people.isb-sib.ch/Heinz.Stockinger/publications/datagrid-ggf2002.pdf

[22] Weil SA, Pollack KT, Brandt SA, Miller AEL. Dynamic metadata management for petabyte-scale file systems. In: Proc. of the 2004 ACM/IEEE Conf. on Supercomputing (SC 2004). Pittsburgh: ACM Press, 2004. [doi: 10.1109/SC.2004.22]

[23] Hua Y, Jiang H, Zhu YF, Feng D, Tian L. SmartStore: A new metadata organization paradigm with semantic-awareness for next-generation file systems. In: Proc. of the FAST 2009. Portland: ACM Press, 2009. 1−12. [doi: 10.1145/1654059.1654070]

[24] Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Proc. of the Symp. on Operating Systems Principles (SOSP 2003). Bolton: ACM Press, 2003. 29−43. [doi: 10.1145/945445.945450]

© 中国科学院软件研究所 http://www.jos.org.cn

[25] The apache software foundation. HDFS Architecture Guide, 2009. http://hadoop.apache.org/common/docs/current/hdfs_design.html

[26] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop distributed file system. In: Proc. of the IEEE 26th Symp. on Mass Storage Systems and Technologies (MSST). Lake Tahoe: IEEE, 2010. 1−10. [doi: 10.1109/MSST.2010.5496972]

[27] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. In: Proc. of the OSDI 2006. Seattle: USENIX Association, 2006. 205−218. http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/zh-CN//archive/bigtable-osdi06.pdf

[28] Muthitacharoen A, Morris R, Gil TM, Chen BJ. Ivy: A read/write peer-to-peer file system. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation. Boston: ACM Press, 2002. 31−44. [doi: 10.1145/1060289.1060293]

[29] Decandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W. Dynamo: Amazon's highly available key-value store. In: Proc. of the SOSP 2007. Stevenson: ACM Press, 2007. 205−220. [doi: 10.1145/1294261.1294281]

[30] Lakshman A, Malik P. Cassandra: A decentralized structured storage system. ACM SIGOPS Operating Systems Review, 2010, 44(2):35−40. [doi: 10.1145/1773912.1773922]

[31] Karger D, Lehman E, Leighton T, Panigrahy R, Levine M, Lewin D. Consistent Hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proc. of the STOC. El Paso: ACM Press, 1997. 654−663. [doi: 10.1145/258533.258660]

[32] Sashi K, Thanamani AS. A new replica creation and placement algorithm for data grid environment. In: Proc. of the 2010 Int'l Conf. on Data Storage and Data Engineering (DSDE). Bangalore: IEEE, 2010. 265−269. [doi: 10.1109/DSDE.2010.38]

[33] Gu QF, Chen B, Zhang YP. Dynamic replica placement and location strategies for data grid. In: Proc. of the 2008 Int'l Conf. on Computer Science and Software Engineering (CSSE). Wuhan: IEEE, 2008. 35−40. [doi: 10.1109/CSSE.2008.1328]

[34] Ding Y, Lu Y. Automatic data placement and replication in grids. In: Proc. of the HIPC. Kochi: IEEE, 2009. 30−39. [doi: 10.1109/HIPC.2009.5433229]

[35] Hsiao HI, Dewitt DJ. Chained declustering: A new availability strategy for multiprocessor database machines. In: Proc. of the 6th Int'l Conf. on Data Engineering (ICDE). Los Angeles: University of Wisconsin, 1990. 456−465. [doi: 10.1109/ICDE.1990.113499]

[36] Buyya R, Corte T, Jin H. Petal: Distributed virtual disks. In: Proc. of the High Performance Mass Storage and Parallel I/O: Technologies and Applications. Piscataway: Wiley-IEEE Press, 2002. 420−430. [doi: 10.1109/9780470544839.ch27]

[37] Dabek F, Kaashoek MF, Karger D, Morris R, Stoica I. Wide-Area cooperative storage with CFS. In: Proc. of the 18th ACM Symp. on Operating System Principles (SOSP). Banff: ACM Press, 2001. 202−215. [doi: 10.1145/502034.502054]

[38] Rowstron A, Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In: Proc. of the SOSP 2001. Banff: ACM Press, 2001. 188−201. [doi: 10.1145/502034.502053]

[39] Chandy JA. A generalized replica placement strategy to optimize latency in a wide area distributed storage system. In: Proc. of the DADC 2008. Boston: ACM Press, 2008. 49−54. [doi: 10.1145/1383519.1383525]

[40] Patterson DA, Gibson G, Katz RH. A case for redundant arrays of inexpensive disks (RAID). In: Proc. of the SIGMOD. Chicago: ACM Press, 1988. 109−116. [doi: 10.1145/50202.50214]

[41] Rizzo L. Effective erasure codes for reliable computer communication protocols. Computer Communication Review, 1997,27(2): 24−36. [doi: 10.1145/263876.263881]

[42] Xiao N, Shu JW, Liu F, Li MQ. Research on the state of art of storage technologies and future trend. In: Wang CH, ed. The Annual Report in 2008 on the Development of Computer Science and Technology. Beijing: China Machine Press, 2009. 12−48 (in Chinese).

[43] Weatherspoon H, Kubiatowicz JD. Erasure coding vs. replication: A quantitative comparison. In: Proc. of the IPTPS. Cambridge: Springer-Verlag, 2002. 328−337. [doi: 10.1007/3-540-45748-8_31]

[44] Rodrigues R, Liskov B. High availability in DHTs: Erasure coding vs.replication. In: Proc. of the IPTPS. Ithaca: Springer-Verlag, 2005. 226−239. [doi: 10.1007/11558989_21]

[45] Zhang Z, Deshpande A, Ma XS, Thereska E, Narayanan D. Does erasure coding have a role to play in my data center? Technical Report, MSR-TR- 2010-52, Microsoft Research, 2010.

© 中国科学院软件研究所 http://www.jos.org.cn

[46] Hafner JL. Weaver codes: Highly fault tolerant erasure codes for storage systems. In: Proc. of the 4th USENIX Conf. on File and Storage Technologies (FAST 2005). San Francisco: USENIX Association, 2005. http://static.usenix.org/events/fast05/tech/full_papers/hafner_weaver/hafner_weaver.pdf

[47] Gallager RG. Low-Density parity-check codes. IEEE Trans. on Information Theory, 1962,8(1):21−28. [doi: 10.1109/TIT.1962.1057683]

[48] Luby MG, Mitzenmacher M, Shokrollahi MA, Spielman DA. Efficient erasure correcting codes. IEEE Trans. on Information Theory, 2001,47(2):569−584. [doi: 10.1109/18.910575]

[49] Plank JS, Thomason MG. A practical analysis of low-density parity-check erasure codes for wide-area storage applications. In: Proc. of the DSN 2004. Florence: IEEE, 2004. 115−124. [doi: 10.1109/DSN.2004.1311882]

[50] Dimakis AG, Godfrey PB, Wainwright M, Ramchandran K. Network coding for distributed storage systems. In: Proc. of the INFOCOM 2007. Anchorage: IEEE, 2007. 2000−2008. [doi: 10.1109/INFCOM.2007.232]

[51] Wu YN, Dimakisy A, Ramchandrany K. Deterministic regenerating codes for distributed storage. In: Proc. of the Allerton Conf. on Control, Computing, and Communication. Urbana: Curran Associates, Inc., 2007. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.3806

[52] Wu YN, Dimakis AG. Reducing repair traffic for erasure coding-based storage via interference alignment. In: Proc. of the IEEE Int'l Symp. on Information Theory (ISIT). Seoul: IEEE, 2009. 2276−2280. [doi: 10.1109/ISIT.2009.5205898]

[53] Dimakis AG, Ramchandran K, Wu YN, Suh C. A survey on network codes for distributed storage. Proc. of the IEEE, 2011,99(3):476−489. [doi: 10.1109/JPROC.2010.2096170]

[54] Plank JS, Simmerman S, Schuman CD. Jerasure: A library in C/C++ facilitating erasure coding for storage applications (version 1.2). Research Report, Department of Electrical Engineering and Computer Science, University of Tennessee, 2008. http://web.eecs.utk.edu/~plank/plank/papers/CS-08-627.pdf

[55] Plank JS, Luo JQ, Schuman CD, Xu LH, Wilcox-O'hearn Z. A performance evaluation and examination of open-source erasure coding libraries for storage. In: Proc. of the FAST 2009. San Francisco: USENIX Assciation, 2009. 253−265. http://nisl.wayne.edu/Papers/Tech/code-pf-fast09.pdf

[56] Fan B, Tantisiriroj W, Xiao L, Gibson G. DiskReduce: RAID for data-intensive scalable computing. In: Proc. of the Petascale Data Storage Workshop (PDSW 2009). Portland: ACM Press, 2009. 6−10. [doi: 10.1145/1713072.1713075]

[57] Lin WK, Chiu DM, Lee YB. Erasure code replication revisited. In: Proc. of the 4th Int'l Conf. on Peer-to-Peer Computing (P2P 2004). Zurich: IEEE, 2004. 90−97. [doi: 10.1109/PTP.2004.1334935]

[58] Kossmann D, Kraska T, Loesing S, Merkli S, Mittal R, Pfaffhauser F. Cloudy: A modular cloud storage system. In: Proc. of the 36th Int'l Conf. on Very Large Data Bases. Singapore: VLDB Endowment, 2010. 1533−1536. http://www.comp.nus.edu.sg/~vldb2010/proceedings/files/papers/D05.pdf

[59] U.S. Environmental Protection Agency. EPA Report on Server and Data Center Energy Efficiency. 2007. http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency_study

[60] Battles B, Belleville C, Grabau S, Maurier J. Reducing data center power consumption through efficient storage. Research Report, NetApp, 2007. http://www.it-executive.nl/images/downloads/reducing-datacenter-power.pdf

[61] Dawson-Haggerty S, Krioukov A, Culler D. Power optimization—A reality check. Research Report, Berkeley: Computer Science Division, University of California, 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-140.pdf

[62] Miyoshi A, Lefurgy C, van Hensbergen E, Rajamony R, Rajkumar R. Critical power slope: Understanding the runtime effects of frequency scaling. In: Proc. of the ICS 2002. New York: ACM Press, 2002. 35−44. [doi: 10.1145/514191.514200]

[63] Wu Q, Juang P, Martonosi M, Peh LS, Clark DW. Formal control techniques for power-performance management. IEEE Micro, 2005, 25(5):52−62. [doi: 10.1109/MM.2005.87]

[64] Lefurgy C, Wang XR, Ware M. Server-Level power control. In: Proc. of the ICAC 2007. Jacksonville: IEEE, 2007. [doi: 10.1109/ICAC.2007.35]

[65] Fan XB, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. In: Proc. of the 34th Annual Int'l Symp. on Computer Architecture. San Diego: ACM Press, 2007. 13−23. [doi: 10.1145/1250662.1250665]

[66] Barroso LA, Holzle U. The case for energy-proportional computing. Computer, 2007,40(12):33−37. [doi: 10.1109/MC.2007.443]

© 中国科学院软件研究所 http://www.jos.org.cn

[67] Song M, Kim M. Solid state disk (SSD) management for reducing disk energy consumption in video servers. In: Proc. of the FAST 2011. San Jose: Usenix Association, 2011. http://static.usenix.org/event/fast11/posters_files/Song_M.pdf

[68] Rivoire S, Ranganathan P, Kozyrakis C. A comparison of high-level full-system power models. In: Proc. of the HotPower 2008. San Diego: Usenix Association, 2008. http://static.usenix.org/event/hotpower08/tech/full_papers/rivoire/rivoire.pdf

[69] Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP. Managing energy and server resources in hosting centers. In: Proc. of the SOSP 2001. Banff: ACM Press, 2001. 103−116. [doi: 10.1145/502034.502045]

[70] Meisner D, Gold BT, Wenisch TF. PowerNap: Eliminating server idle power. In: Proc. of the ASPLOS 2009. Washington: ACM Press, 2009. 205−216. [doi: 10.1145/1508244.1508269]

[71] Gurumurthi S, Sivasubramaniam A, Kandemir M, Franke H. Reducing disk power consumption in servers with drpm. IEEE Computer, 2003,36(12):59−66. [doi: 10.1109/MC.2003.1250884]

[72] Hamilton J. CEMS: Low cost, low power servers for Internet-scale services. In: Proc. of the Conf. on Innovative Data Systems Research (CIDR). Asilomar, 2009. http://www.mvdirona.com/jrh/TalksAndPapers/JamesHamilton_CEMS.pdf

[73] Vasudevan V, Franklin J, Andersen D. FAWNdamentally power-efficient clusters. In: Proc. of the HotOS XII. Monte Verita: Usenix Association, 2009. http://vijay.vasu.org/static/papers/hotos2009.pdf

[74] Kim HS, Shin DI, Yu YJ, Eom H, Yeom HY. Towards energy proportional cloud for data processing frameworks. In: Proc. of the 1st USENIX Workshop on Sustainable Information Technology (SustainIT 2010). San Jose: Usenix Association, 2010. http://static.usenix.org/event/sustainit10/tech/full_papers/kim.pdf

[75] Szalayy AS, Bell G, Huangz HH, Terzisy A, Whitey A. Low-Power amdahl-balanced blades for data intensive computing. ACM SIGOPS Operating Systems Review, 2009,44(1):71−75. [doi: 10.1145/1740390.1740407]

[76] Lim K, Ranganathan P, Chang J, Patel C, Mudge T, Reinhardt S. Understanding and designing new server architectures for emerging warehouse-computing environments. In: Proc. of the 35th Int'l Symp. on Computer Architecture. Beijing: IEEE, 2008. 315−326. [doi: 10.1109/ISCA.2008.37]

[77] Yin S, Alghamdi MI, Ruan XJ, Nijim M, Tamilarasan A, Zong ZL, Qin X, Yang YM. Improving energy efficiency and security for disk systems. In: Proc. of the HPCC 2010. Melbourne: IEEE, 2010. 442−449. [doi: 10.1109/HPCC.2010.26]

[78] Harnik D, Naor D, Segall I. Low power mode in cloud storage systems. In: Proc. of the IPDPS 2009. Rome: IEEE, 2009. 1−8. [doi: 10.1109/IPDPS.2009.5161231]

[79] Pinheiro E, Bianchini R. Energy conservation techniques for disk array-based servers. In: Proc. of the ICS 2004. Malo: ACM Press, 2004. 68−78. [doi: 10.1145/1006209.1006220]

[80] Sümer O. Partial covering of hypergraphs. In: Proc. of the SODA 2005. Vancouver: Society for Industrial and Applied Mathematics, 2005. 572−581. http://dl.acm.org/citation.cfm?id=1070512&dl=ACM&coll=DL&CFID=71057116&CFTOKEN=85304476

[81] Li D, Wang J. Eeraid: Energy efficient redundant and inexpensive disk array. In: Proc. of the ACM SIGOPS European Workshop. Leuven: ACM Press, 2004. [doi: 10.1145/1133572.1133577]

[82] Greenan KM, Long DDE, Miller EL, Thomas SJ, Wylie JJ. A spin-up saved is energy earned: Achieving power-efficient, erasure-coded storage. In: Proc. of the HotDep 2008. San Diego: USENIX Association, 2008. http://static.usenix.org/event/hotdep08/tech/full_papers/greenan/greenan.pdf

[83] Weddle C, Oldham M, Qian J, Wang AA, Reiher P, Kuenning G. Paraid: A gear-shifting power-aware raid. ACM Trans. on Storage (TOS), 2007,3(3):1553−1569. [doi: 10.1145/1289720.1289721]

[84] Li D, Wang J. Conserving energy in conventional disk based RAID systems. In: Proc. of the 3rd Int'l Workshop on Storage Network Architecture and Parallel I/Os (SNAPI 2005). 2005. 65−72. http://www.eece.maine.edu/snapi/snapi05/DongLi.pdf

[85] Yao XY, Wang J. Rimac: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. In: Proc. of the EuroSys. Leuven: ACM Press, 2006. 249−262. [doi: 10.1145/1217935.1217959]

[86] Pinheiro E, Bianchini R, Dubnicki C. Exploiting redundancy to conserve energy in storage systems. In: Proc. of the SIGMetrics/Performance 2006. Saint Malo: ACM Press, 2006. 15−26. [doi: 10.1145/1140277.1140281]

[87] Leverich J, Kozyrakis C. On the energy (in) efficiency of Hadoop clusters. ACM SIGOPS Operating Systems Review, 2010,44(1):61−65. [doi: 10.1145/1740390.1740405]

© 中国科学院软件研究所 http://www.jos.org.cn

[88] Colarelli D, Grunwald D. Massive arrays of idle disks for storage archives. In: Proc. of the Supercomputing 2002. Baltimore: ACM Press, 2002. 1−11. http://www.supercomputing.org/sc2002/paperpdfs/pap.pap312.pdf [doi: 10.1109/SC.2002.10058]

[89] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage. ACM Trans. on Storage (TOS), 2008,4(3):253−267. [doi: 10.1145/1416944.1416949]

[90] Storer MW, Greenan KM, Miller EL, Voruganti K. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In: Proc. of the FAST 2008. San Jose: Usenix Association, 2008. 1−16. http://www.ssrc.ucsc.edu/Papers/storer-fast08.pdf

[91] Zhu QB, Chen ZF, Tan L, Zhou YY, Keeton K, Wilkes J. Hibernator: Helping disk arrays sleep through the winter. In: Proc. of the 20th ACM Symp. on Operating systems principles (SOSP). Brighton: ACM Press, 2005. 177−190. [doi: 10.1145/1095810.1095828]

[92] Vasić N, Barisits M, Salzgeber V, Kostic D. Making cluster applications energy-aware. In: Proc. of the ACDC 2009. Barcelona: ACM Press, 2009. 37−42. [doi: 10.1145/1555271.1555281]

[93] Zhu QB, David FM, Devaraj CF, Li ZM, Zhou YY, Cao P. Reducing energy consumption of disk storage using power-aware cache management. In: Proc. of the HPCA 2004. Madrid: IEEE, 2004. 118−129. [doi: 10.1109/HPCA.2004.10022]

[94] Chen YP, Ganapathi AS, Fox A, Katz RH, Patterson DA. Statistical workloads for energy efficient mapreduce. Technical Report, UCB/EECS-2010-6, Berkeley: University of California, 2010. http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-6.html

[95] Chen YP, Keys L, Katz RH. Towards energy efficient MapReduce. Technical Report, UCB/EECS-2009-109, Berkeley: University of California, 2009.

[96] Oppenheim BM. Reducing cluster power consumption by dynamically suspending idle nodes [MS. Thesis]. San Luis Obispo: Facalty of California Polytechnic State University, 2010.

[97] Thereska E, Donnelly A, Narayanan D. Sierra: A power-proportional, distributed storage system. Technical Report, MSR-TR-2009-153, 2009.

附中文参考文献:

[4] 陈康,郑纬民.云计算:系统实例与研究现状.软件学报,2009,20(5):1337−1348. http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[42] 肖侬,舒继武,刘芳,李明强.存储技术的研究发展现状与趋势.科技报告,2009.

王意洁(1971－),女,湖南长沙人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,海量数据处理和并行与分布处理.

孙伟东(1982－),男,博士生,主要研究领域为分布式存储,纠删码.

周松(1985－),男,硕士,主要研究领域为容错存储,纠删码.

裴晓强(1986－),男,博士生,主要研究领域为云计算,容错存储.

李小勇(1982－),男,博士生,CCF 学生会员,主要研究领域为网络计算,数据库.

© 中国科学院软件研究所 http://www.jos.org.cn