

蓝鲸文件系统中元数据与数据隔离技术

张敬亮^{1,2}, 张军伟^{1,2}, 张建刚¹, 许 鲁¹

(1. 中国科学院计算技术研究所, 北京 100190; 2. 中国科学院研究生院, 北京 100039)

摘 要: 针对文件系统中元数据访问对应用数据访问的干扰问题, 基于资源分配分离及访问路由分发技术, 在蓝鲸文件系统中实现元数据和数据存储及 I/O 时的隔离。测试结果表明, 将元数据从数据中隔离并将其独立配置到高速存储后, 应用 I/O 带宽提高了 2.6 倍~2.8 倍, 操作吞吐率提高了 2 倍~5 倍。

关键词: 蓝鲸文件系统; 集群文件系统; 元数据隔离; Dbench 测试集

IsolatI/On Technology of Metadata from Data in Blue Whale File System

ZHANG Jing-liang^{1,2}, ZHANG Jun-wei^{1,2}, ZHANG Jian-gang¹, XU Lu¹

(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190;

2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

[Abstract] Aiming at the performance interference between metadata and data access, this paper proposes a novel mechanism in Blue Whale File System(BWFS) to isolate metadata from data both in physical storage and I/O path. The isolatI/On is realized by the resource separatI/On and I/O request dispatching. Test results show that, after metadata is separated and put in the high-performance media, the applicatI/On I/O bandwidth is increased by 2.6 times~2.8 times, while OPS is improved by 2 times~5 times.

[Key words] Blue Whale File System(BWFS); cluster file system; metadata isolatI/On; Dbench test set

1 概述

文件系统中信息分为数据与元数据 2 个部分。其中元数据操作频繁, 占总操作数的 50%以上^[1]; 元数据的特点决定了其粒度较小且分布分散。同时为保证文件系统的状态一致性, 元数据一般采用同步读写的方式^[2]。而文件系统中普通数据的读写多通过 readahead, cache, write through 等技术进行聚合再以大块连续的方式提交到磁盘。文件系统中元数据与数据体现为 2 种差异化的访问模式。在由磁盘这种访问模式敏感的存储介质构成的存储系统中, 虽然元数据访问带宽并不高, 但其访问模式会使磁头移动频繁, 造成磁盘 I/O 带宽急剧下降, 从而对同一存储区域的数据访问产生较大干扰。而同步 I/O 方式又会使元数据操作响应时间变长而成为系统瓶颈。因此, 有效隔离元数据访问对整个文件系统的性能提升有直接意义。

在集群文件系统环境下存储规模更大。随着并发应用负载的增多, 多个数据 I/O 流与元数据 I/O 流同时对底端存储进行访问。尤其在元数据密集型的应用中, 多并发的元数据访问会造成磁盘访问的高度繁忙, 进而严重影响并发数据流的访问性能。通过将集群文件系统中的元数据与数据进行存储分离, 则可在较大程度上提高数据与元数据的访问并行度, 减少元数据操作对应用数据流的干扰; 同时根据元数据访问特点将其分布到快速随机存储介质上可以明显消除元数据的访问瓶颈, 进而有效提升整个集群文件系统的性能。

与数据相比, 文件系统中元数据容量一般仅为前者的 1/1 000~1/100。随着 SSD 等随机存储介质成本的不断降低,

使在大规模集群文件系统中用较小的代价将元数据分布到高速存储上成为可能。

本文提出一种基于蓝鲸集群文件系统(Blue Whale File System, BWFS)^[3]的元数据与数据分离架构, 并通过可配置的卷管理机制将元数据分布到高速存储, 从而有效提高存储系统的性能和扩展性。

2 BWFS 中元数据隔离机制的设计与实现

BWFS 主要包括网络存储设备集群(SN cluster)及文件元数据服务器集群(MS cluster)2 个部分。其中, 网络存储设备集群可以支持多种网络设备如 ISCSI, FC, IB 等, 文件元数据服务器集处理元数据请求, 为应用服务器集群(AS cluster)提供文件系统的元数据服务。

BWFS 中元数据隔离机制的实现如图 1 所示, 具体步骤如下:

- (1)将底层的存储设备虚拟化为可管理的逻辑资源;
- (2)通过资源管理通路的分离完成元数据与数据存储位置的分离;
- (3)通过 I/O 通路的分离实现两者的访问分离。

基金项目: 国家“863”计划基金资助项目(2007AA01Z402); 国家“973”计划基金资助项目(2004CB318205)

作者简介: 张敬亮(1979—), 男, 博士研究生, 主研方向: 网络存储, 集群文件系统; 张军伟, 博士研究生; 张建刚, 副研究员、博士; 许 鲁, 研究员、博士生导师

收稿日期: 2009-06-04 **E-mail:** zhangji@ict.ac.cn

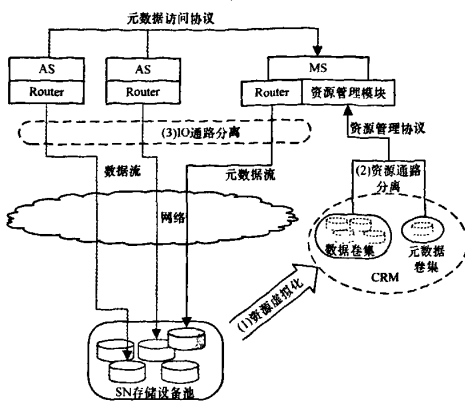


图1 BWFS中元数据隔离机制

2.1 存储资源虚拟化

BWFS中引入了“卷”机制来对存储资源进行管理。BWFS将底层存储设备按照需要虚拟化为多个资源卷。卷按其存储内容不同可分为数据卷、元数据卷、journal卷等不同类型。在BWFS中,由资源管理模块(Central Resource Management, CRM)对系统的卷资源进行统一管理。对不同的卷CRM据其特点赋予不同的数据结构和资源管理算法。比如BWFS中对普通数据、元数据及inode的分配应用不同的分配粒度和资源定位算法。

与ext2/3文件系统不同, BWFS将资源管理与存储设备进行了分离。在ext2/3文件系统中,资源以BG的方式进行组织,每个BG的头部存放管理此段BG资源的元数据,包括inode信息、block及inode管理位图等。而BWFS中资源管理元数据由CRM统一管理并单独存储在元数据卷设备中,仅由CRM进行读写。资源管理元数据与被管理资源在物理上进行了分离,两者只存在逻辑上的对应关系。

2.2 MS资源通路的分离

在BWFS中,superblock, inode, journal、资源管理位图等静态元数据在系统初始化时便可直接分布到元数据卷中;而运行时产生的元数据如目录块,间接块等则可在资源分配环节与数据隔离并分布到元数据卷中。经过上述2步可实现所有元数据在存储位置上与数据相隔离。

BWFS采用2层资源管理结构:CRM资源池及MS端的资源缓存。CRM管理BWFS中全局资源,而MS端则先从CRM“批发”大块资源缓存到本地,再以“零售”的方式提供给资源请求者,以此来提高资源管理的效率。资源申请时元数据与数据的隔离流程如图2所示。具体步骤如下:

(1)MS在上层模块有资源申请时根据资源用途将申请分为数据申请及元数据申请2类。

(2)如果为元数据申请则首先从本地资源缓存的元数据卷组(MV)中选择合适的卷进行分配。否则从数据卷组(DV)中选择(具体卷的选择可通过BWFS的卷管理机制进行配置,缺省情况下则根据预设策略及当前资源使用情况自动选择)。

(3)如果当前MS的资源缓存满足当前资源请求则直接从缓存中返回,否则通过CRM_agent向CRM模块申请。

(4)CRM_agent收到资源请求后将请求通过资源交互协议发送到CRM模块,此协议中包含申请资源卷的id。

(5)CRM端的守护进程CRM_server接收到此请求后,根据指定的卷id,向此卷对象发出具体的资源请求,卷资源管理模块依据本卷对应的资源管理算法寻找合适的资源段并

返回。

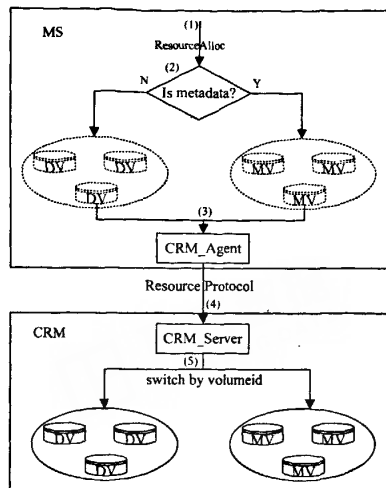


图2 资源申请隔离流程

2.3 I/O通路的分离

传统文件系统如ext2/3只有一个主设备,因此其元数据及数据的读写操作直接发送到此设备即可,而在BWFS运行时可访问多个数据卷及元数据卷,因此,需要提供一种路由机制将元数据及数据请求转发到不同设备。BWFS中由router模块完成I/O请求的转发。其路由信息的建立通过“卷标扫描”机制实现。

(1)在BWFS文件系统初始化时,系统会给每个卷分配唯一的卷标(vid),同时将此vid通过“打标”过程写入此卷的特定位置;在块资源申请时也将其在所在卷的vid作为高位部分写入块号中(BWFS采用64位块号,其中高8位分配给vid)。

(2)在MS及AS端文件系统运行之前均需执行卷标扫描过程,通过对所有网络设备的卷标域进行扫描,建立每个网络设备与vid的路由映射表并导入router模块。

(3)当MS或AS的需要访问某一块的数据时,router模块从其块号中提取vid,然后通过查找路由表将读写请求发送到对应的网络设备队列中,最后以块号中低位部分作为卷内偏移在某一具体卷内完成实际读写操作。

因为元数据卷与数据卷在初始化时就分配了不同的vid,所以基于上述路由机制便可实现元数据与数据在I/O通路上的隔离。此外在BWFS中仅有block读写时需要路由。而inode资源存放在元数据卷上,其读写通过专门协议完成,并不经过I/O通路,因此,不需要路由分发。

3 测试及分析

本文测试基于BWFS环境,其中存储设备为ISCSI。为减少不同元数据操作之间的干扰,笔者又将元数据细分为3类:文件系统元数据,journal及资源管理元数据。元数据与数据按如下4种分布方式进行了测试:

(1)I-I:将元数据、journal、资源元数据及数据均分布在一块ISCSI磁盘中。

(2)I3-I:将元数据、journal、资源元数据及数据各自分布在一块ISCSI磁盘中。

(3)S-I:将元数据、journal、资源元数据分布在MS的一块本地SATA磁盘中,数据分布在一块ISCSI磁盘中。

(4)R-I:将元数据、journal、资源元数据分布在MS的RAM DISK上,数据分布在一块ISCSI磁盘中(因条件所限,

本测试中用 RAM 来模拟 DRAM-SSD 设备的效果)。

上述 4 种方式中“1”为元数据/数据混杂方式,后三者则为元数据隔离到元数据卷的方式。在本文测试环境下元数据卷的性能按 1→S→R 顺序递增。

本文利用 Dbench^[4]测试集进行应用模拟测试验证。Dbench 是对 NetBench^[5]的改进,可在一台 Linux 节点上通过多进程模拟多个应用节点对 samba 服务器的 I/O 负载。负载通过对实际应用中抓取 trace 进行回放产生。为提高测试时元数据操作部分的压力,将 Dbench 中数据操作部分过滤掉后生成了一个新的测试工具 DbenchNOI/O,其仅产生元数据负载。通过 Dbench 与 DbenchNOI/O 的并发组合,生成了 3 个测试用例(如表 1 所示)。3 个用例分别模拟了不同元数据/数据负载比例下的应用情况。

表 1 用例进程组合的并发进程数

case	Dbench	DbenchNOI/O
case1	128	128
case2	256	384
case3	0	128

测试对 Dbench 中如下指标进行了比较:

(1)OPS(OperatI/Os Per Second): 1 s 内 dbench 的一个客户端执行完成的 trace 中的操作条数,即客户端的操作吞吐率。

(2)数据卷设备 busy 度:在设备驱动层抓取的表征磁盘繁忙程度的参数,范围为 0%~100%。磁盘 busy 度与访问模式相关,因此,不与输出带宽呈简单线性关系。

(3)数据卷设备 I/O 带宽:数据卷设备的读写带宽总和,即应用的 I/O 带宽。

指标 2 和指标 3 为对数据卷设备负载情况的描述。因为 case3 中不含数据操作,所以这 2 个指标中不包括 case3 的情况。测试结果如图 3、图 4 所示。

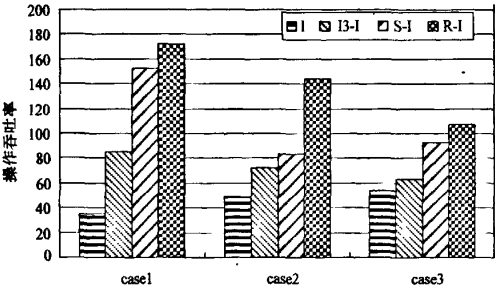


图 3 操作吞吐率

图 3 中所有用例的吞吐率随分布方式变化有明显增长。3 个用例在 R-I 方式下比 1 方式分别有近 5 倍、3 倍、2 倍的提升。因为 OPS 为模拟实际 Windows 应用中的操作吞吐率,由测试结果可以推断元数据隔离技术对实际应用的性能提升会有显著效果。

case3 中仅包含元数据 I/O,但 I3-I 方式比 1 方式 OPS 有提高。这是因为 BWFS 中元数据更新通过 journal 机制实现,元数据更新时要首先写入 journal 区域。而 I3-I 方式将 journal 和元数据负载隔离到单独磁盘,从而比 1 方式有更好的性能。

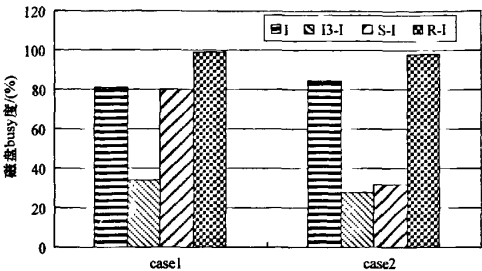
2 个用例的 busy 度随分布方式变化均呈现先降后升的“U”型分布,而输出带宽却一直增长,其原因如下:

(1)在 1 方式下数据与元数据混杂,访问时 2 种模式的负

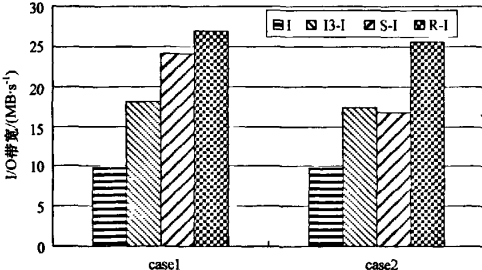
载流相互干扰,造成磁盘的高度 busy;I3-I 方式中在将元数据从数据卷设备隔离后,消除了易造成磁头抖动的元数据负载流的干扰,数据卷设备 busy 度降低,同时由于数据卷中只剩下访问连续性较好的数据负载,因此,其带宽反而有较大增加。

(2)在元数据隔离后,随着元数据卷设备性能的提高,处于文件系统操作关键路径上的元数据操作变快,因而使数据操作请求频度增高,对数据卷的 I/O 负载变大,因而造成数据卷设备的 busy 度及带宽一同增加。即随元数据卷设备性能的提高,元数据操作路径变短,系统瓶颈逐渐转移到数据卷设备上。R-I 方式下 2 个用例的 busy 度均已接近 100%上限,说明此时数据卷设备利用率已近饱和,数据卷设备性能成为系统瓶颈。

元数据隔离前后数据卷设备的负载变化如图 4 所示。



(a)数据卷设备 busy 度



(b)数据卷设备 I/O 带宽

图 4 元数据隔离前后数据卷设备的负载变化

由图 4 可以看出,1 方式与 R-I 方式下设备的 busy 度相差不多,而 2 个用例的 I/O 带宽却分别提高了 2.8 倍和 2.6 倍(由于 1 方式下数据与元数据混杂,本文抓取的数据卷设备带宽为元数据与数据带宽之和,因此实际带宽提高倍数较图 4(b)更高)。本结果进一步验证了磁盘介质的非线性访问特性:即多模式流混杂访问情况下磁盘的最大带宽远低于单模式流顺序访问的最大带宽。而这也是本研究中将元数据流与数据流进行隔离的依据所在。

图 4 中 case2 的 busy 度及带宽在 I3-I 方式与 S-I 方式下差别不大,这是因为 case2 中元数据操作比例较高,所以其对应 journal 负载也较高。虽然 I3-I 元数据卷设备性能较差,但是通过将元数据负载在多元数据卷上的隔离并发,I3-I 方式可取得与 S-I 方式大致相当的元数据操作性能,从而对数据卷产生大致相同的负载。

基于 Dbench 的测试结果表明在将元数据隔离后,通过对数据卷设备的卸载及元数据访问瓶颈的消除,应用 I/O 性能得到有效提高。

(下转第 38 页)

性。本文选取最具代表性的 BM25 模型和临近信息模型(记做 PROX)作为基于“词频”的传统相关性排序模型的代表,与 MED 进行比较实验。

3.2 实验结果

受篇幅限制仅列出在锚文本和 URL 2 个域上的结果,实验中所用到的参数都是在训练数据集合上得到的最佳参数,然后在测试集合上直接使用。为了克服实验的偶然性,采用 5-fold 交叉验证的方法,所列结果是 5 个测试集合上得到的平均评测结果,如表 3 所示,其中, $I1, I2, D1, D2$ 分别代表插入查询词、插入非查询词、临时删除和彻底删除。

表 3 实验参数

方法	网页域	参数
BM25	锚文本	$k_1=0.2, b=0.1$
	URL	$k_1=1.1, b=0.99$
PROX	锚文本	$k_1=0.4, b=0.1$
	URL	$k_1=1.3, b=0.99$
MED	锚文本	$I1=1, I2=3, D1=7, D2=40$
	URL	$I1=1, I2=1, D1=4, D2=33$

对传统的排序模型和 MED 在简短网页域上的排序结果进行比较,如图 1、图 2 所示。

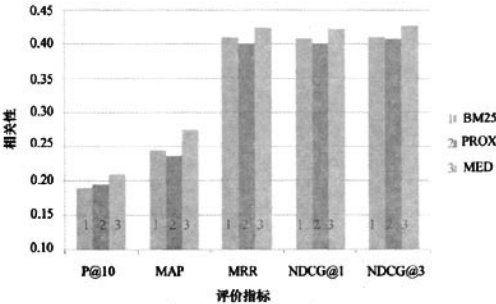


图 1 锚文本域的排序结果

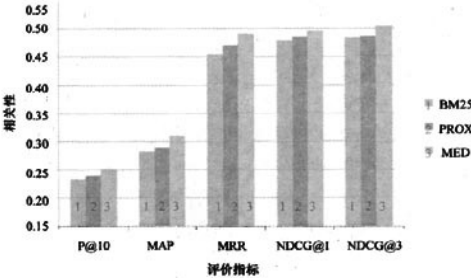


图 2 URL 域的排序结果

从图 1、图 2 可以看出, MED 在 2 个域的所有评价指标上都远超过传统方法,且通过显著性测试表明,这些超越都具有统计意义上的显著性($p\text{-value}<0.05$)。这说明相比于传统的基于“词频”的相关性排序算法, MED 可以更好地表达用户查询和简短的网页域之间的相关性。另外,从实验结果中可以看到一个比较有趣的现象:在 URL 域中,引入距离信息的 PROX 的结果反而低于 BM25 的结果。这个结果其实并不奇怪,因为 PROX 的基本假设就是“查询词在域中分布的越近,则查询和域之间越相关”,但是 URL 中重要的信息分布于 URL 字符串的两头,所以查询词在 URL 中并非“越近越好”。从锚文本的结果中也可以看出,在锚文本中 PROX 的上述假设有一定的合理性,所以在各种指标下都较 BM25 好。

总的来说, MED 比传统方法更适衡量简短网页域和用户查询之间的相关性,这对于改善整个网页的排序结果有着重要意义。

4 结束语

本文提出的改进的编辑距离算法通过引入代表相关性的顺序、距离和位置等信息以及精心设计的编码过程,将查询和简短域之间的相关性,转化为编码后的字符串的相似性问题,并通过动态规划降低了整个算法的时间复杂度,使其具有实用能力。实验结果表明,相比于传统相关性排序算法,该算法在所有的评价指标上都提高了排序的性能。这个结论对于搜索引擎的算法选择具有很好的启示作用。今后的工作将研究如何通过机器学习的方法自动地从标注好的数据集上学习算法参数。

参考文献

[1] Robertson S E. Experimentation as a Way of Life: Okapi at Trec[J]. Information Processing & Management, 2000, 36(1): 95-108.
[2] Buttcher S. Term Proximity Scoring for Ad-hoc Retrieval on Very Large Text Collections[C]//Proceedings of ACM SIGIR'06. [S. l.]: ACM Press, 2006.
[3] Levenshtein V I. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals[J]. Soviet Physics Doklady, 1966, 10(8): 707-710.
[4] 车万翔, 刘挺, 秦兵, 等. 基于改进编辑距离的中文相似句子检索[J]. 高技术通讯, 2004, 14(7): 15-20.
[5] Cormen T H. 算法导论[M]. 北京: 机械工业出版社, 2006.

编辑 陈文

(上接第 30 页)

4 结束语

本文介绍了 BWFS 中元数据与数据隔离机制的设计与实现,应用模拟结果表明其可有效消除元数据访问瓶颈、大幅提高数据设备利用率,从而有效提升应用系统的 I/O 性能。本文研究的前提是磁盘的非线性访问特性,其本质是将不同访问模式的 I/O 流在磁盘介质上进行隔离。结果表明,对磁盘存储系统采用基于模式流而治之的优化方法会有较好效果。对不同 I/O 模式的应用数据流进行有效隔离是进一步要研究的内容。


参考文献

[1] Roselli D, Lorch J, Anderson T. A Comparison of File System Workloads[C]//Proc. of USENIX'00. San Diego, CA, USA: [s. n.],

2000: 41-54.

[2] Ruemmler C, Wilkes J. UNIX disk access patterns[C]//Proc. of the USENIX'93. San Diego, CA, USA: [s. n.], 1993: 405-420.
[3] 黄华, 张建刚, 许鲁. 蓝鲸分布式文件系统的分布式分层资源管理模型[J]. 计算机研究与发展, 2005, 42(6): 1034-1038.
[4] Tracer A, Zadok E, Joukov N, et al. A Nine Year Study of File System and Storage Benchmarking[R]. New York, USA: Computer Science Department, Stony Brook University, Tech. Rep.: FSL-07-01, 2008.
[5] NetBench[Z]. (2002-04-06). <http://www.veritest.com/benchmarks/netbench/>.

编辑 金胡考

作者：[张敬亮](#)，[张军伟](#)，[张建刚](#)，[许鲁](#)，[ZHANG Jing-liang](#)，[ZHANG Jun-wei](#)，[ZHANG Jian-gang](#)，[XU Lu](#)
作者单位：[张敬亮, 张军伟, ZHANG Jing-liang, ZHANG Jun-wei \(中国科学院计算技术研究所, 北京, 100190; 中国科学院研究生院, 北京, 100039\)](#)，[张建刚, 许鲁, ZHANG Jian-gang, XU Lu \(中国科学院计算技术研究所, 北京, 100190\)](#)
刊名：[计算机工程](#) 
英文刊名：[COMPUTER ENGINEERING](#)
年，卷(期)：2010，36(2)
被引用次数：0次

参考文献(5条)

1. [Roselli D, Lorch J, Anderson T](#) [A Comparison of File System Workloads](#) 2000
2. [Ruemmler C, Wilkes J](#) [UNIX disk access patterns](#) 1993
3. [黄华, 张建刚, 许鲁](#) [蓝鲸分布式文件系统的分布式分层资源管理模型](#) [期刊论文] - [计算机研究与发展](#) 2005 (6)
4. [Traeger A, Zadok E, Joukov N](#) [A Nine Year Study of File System and Storage Benchmarking](#)
[Tech. Rep. :FSL-07-01](#) 2008
5. [NetBench](#) 2002

本文链接：http://d.g.wanfangdata.com.cn/Periodical_jsjgc201002010.aspx

授权使用：中科院计算所(zkyjsc)，授权号：6bc0344e-a3a0-4d8d-b09f-9e4001012ed3

下载时间：2010年12月2日