计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
49(Suppl.): 252-256, 2012

分布式文件系统的流式数据预读

闫 鹤 李小勇 胡 鹏 刘海涛
(上海交通大学信息安全工程学院 上海 200240)
(yanhe378@sjtu.edu.cn)

Streaming Data Readahead on Distributed File System

Yan He, Li Xiaoyong, Hu Peng, and Liu Haitao
(*School of Information Security Engineering, Shanghai Jiao Tong University, Shanghai* 200240)

Abstract The IO process of Distributed File System includes network transmission and disk IO on server, which tend to be the bottlenecks that affect performance. How to explore the potential of network and disk performance has long been a subject of very active research. We design and implement a distribute file system BlueOcean, but we find that BlueOcean has not paralleled up to network and disk. So we have proposed and implemented a streaming data read-ahead method, which interleaves network transmission and disk IO and achieves a delay hiding. With experimental test, we found this method has actually improved system performance.

Key words distribute file system; readahead; latency hiding; network transmission; disk IO

摘 要 在分布式文件系统中，网络和磁盘往往是影响IO性能的主要因素.如何最大限度地挖掘网络和磁盘的性能潜力,长期以来一直都是非常活跃的研究课题.已有研究工作主要侧重于预取策略和数据的缓存策略,而未能将网络和服务器端磁盘I/O统一进行调度.提出并设计实现了一种流式预读方法,通过在客户端改变发送预读请求和接收预读数据的顺序,交错网络传输和磁盘访问,实现了延迟隐藏.测试表明,这种方法确实显著提高了顺序读性能.

关键词 分布式文件系统;预读;延迟隐藏;网络传输;磁盘IO

中图法分类号 TP316.4

近年来随着计算技术的迅猛发展,分布式环境下的网络瓶颈,I/O瓶颈问题日益严重,用户文件存取能力的需求越来越严格,使得该问题更加突出.因此提高分布式文件系统的IO性能具有重要意义.

常用的文件系统IO性能优化方法包括预读和写合并.其中预读对读性能的提高起到了重要的作用.目前对预读方法的研究注重于cache的有效利用和数据一致性问题,如参考文献[1-2],多数关于预读技术的研究又往往注重于对数据访问模式的识别上,即准确预测应用程序下次将要访问到的数据,并将这些数据预先从服务器端读取到客户端的缓冲区中,如参考文献[3].而对于网络文件系统中如何并行化网络传输和磁盘IO未见考虑,使得系统IO性能未能得到充分挖掘.

本文以我们设计的BlueOcean大规模分布式存储系统为基础,提出了一种适用于分布式文件系统的流式数据预读方法.通过在客户端改变发送预读请求和接收预读数据的顺序,交错网络传输和磁盘

收稿日期:2012-01-04
基金项目:上海市科委基金资助重大项目(10DZ1500200)

访问,实现了延迟隐藏.通过对比实现前后的吞吐量,可以看到该预读方法确实显著提高了系统的读性能.该方法同样适用于其他分布式文件系统.

1　BlueOcean 系统架构

1.1　BlueOcean 系统拓扑结构

BlueOcean 是我们设计的基于对象存储的大规模分布式文件系统,其架构类似于 GFS[4],HDFS[5],KFS[6].BlueOcean 的客户端支持 POSIX 接口,应用程序可以通过它透明地访问文件系统.BlueOcean 支持 NFS 定义的文件属性,支持顺序和随机访问文件数据.

BlueOcean 系统中有一个管理节点来维护名字空间、文件元数据等信息.多个数据节点存储文件数据,提供对数据的访问.多个客户端向应用提供 POSIX 接口用来对存储系统进行访问和操作.他们都运行在 Linux 系统上的用户态空间.文件被分成固定大小的块,以普通 Linux 文件的方式存储在数据节点上.如图 1 所示:

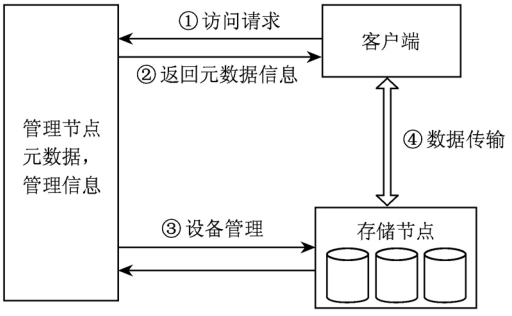


图 1　BlueOcean 系统架构

1.2　BlueOcean 数据节点 IO 处理方式

BlueOcean 数据节点以线程池方式实现高并发 IO 处理.BlueOcean 对每一个磁盘设备都设置一个请求队列和多个工作线程.当有一个 IO 请求到来时,主线程负责把请求放到相应磁盘的请求队列中.同时,有多个工作线程检查这个磁盘的请求队列,一旦发现队列里有请求,就从中取出相应的请求进行 IO 操作.当 IO 操作完成后,工作线程把应答放入应答队列,并且通知主线程 IO 操作完成,由主线程把结果传输给客户端,如图 2 所示.

由于数据节点上的磁盘 IO 和网络传输数据在不同的线程中进行处理,数据节点提供了并行化网络传输和磁盘 IO 的机制.另外,这种线程池的处理模型使得请求被随机分配给不同的线程里进行处

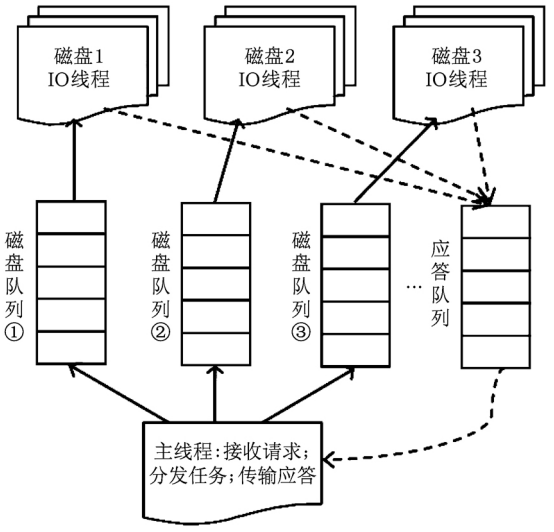


图 2　数据节点架构图

理,其在数据节点上的处理次序是乱序的.因此,客户端能否采取良好的读写策略来利用数据节点提供的并行化磁盘 IO 和网络传输机制或者尽量避免顺序的请求被乱序是提高 IO 性能的关键.

1.3　客户端 IO 请求方式

客户端是 IO 请求的发起方,数据的 IO 操作分为 4 个过程.

1)客户端向管理节点询问元数据信息.

2)管理节点向客户端返回元数据信息,告知它应该和哪个数据节点交互.

3)管理节点向客户端发送设备管理命令,并收取应答.

4)客户端直接和数据节点进行数据传输.

客户端使用预读提高 IO 性能.为了解决应用请求和服务器处理速度不一致的问题,客户端对读请求设置有一个内存缓冲区 Buffer.

2　读性能优化

2.1　顺序读交互过程

BlueOcean 从管理节点获取数据所在的位置信息后,直接同数据节点交互,执行数据的读请求.以预读大小为 256 KB 为例,数据交互过程如图 3 所示.

可以看出,顺序读过程分为以下几个阶段:

1)对于第 1 次读请求,只要请求的数据量小于 1 MB,BlueOcean 就会向数据节点同时发送 4 个 256 KB 的读请求,并且等待,直到收到这 1 MB 的数据并把它放入内存缓冲区中.

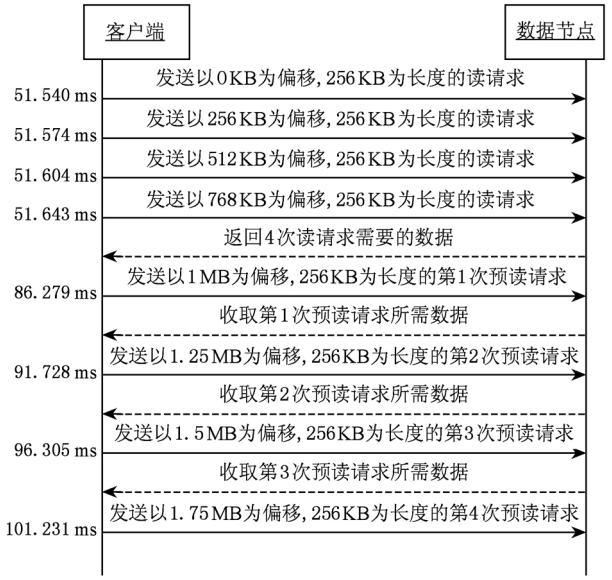


图3 顺序读交互过程

2) 当这1 MB的请求接收完以后,客户端会向数据节点发送一个256 KB的预读请求,这256 KB的预读请求是以异步的方式发送出去的,即客户端不必等待收到这256 KB的数据,就可以获得控制权.

3) 当应用再有读请求的时候,它首先会检查当前的内存缓冲区中是不是有所需要的数据,如果有,则直接从内存缓冲区中把数据返回给用户;如果没有则检查之前发送的预读请求是否已经包含了当前所需读的数据,如果包含了,那么客户端就直接把上一次预读的数据收上来,并返回给用户;否则说明该次读请求不是一个顺序的读请求.

4) 只要收上来上一次预读的数据,就说明读请求是顺序的,接着发送下一次的256 KB的预读请求.

5) 重复步骤3).

总体来说,BlueOcean的读分为前1 MB的PIPEREAD和之后256 KB的异步预读.

2.2 预读过程分析

从顺序读交互过程来看,当有较大数据量的顺序读请求时候,只有前1 MB请求是做的PIPEREAD,后面的所有读请求都是以256 KB为单位的预读,因此1 MB后面的读才是影响性能的关键因素.

BlueOcean前1 MB读请求之后的预读吞吐量不高的原因主要在于预读策略的设计.这些预读请求相对于客户端来说是异步的,但是每2次的预读之间却是串行的过程.在客户端收数据的这段时间内,数据节点并不做任何的IO操作.这样磁盘IO和数据传输串行的进行,并没有利用好数据节点并行化磁盘IO和网络传输的机制,极大地影响了系统的读吞吐量.

因此读性能的优化主要从1 MB之后的预读请求方面着手.对于前1 MB的请求不采用PIPEREAD方式即可,而对于较大数据量的顺序读请求,1 MB之后的持续的预读是性能优化的关键.

2.3 预读的优化方法

2.3.1 流式数据预读方法

由第2.2节的分析可以看出,BlueOcean客户端的预读最明显的缺点是没有利用好数据节点磁盘IO和传输并发的特性,读数据和传输数据是串行进行的,因此调整预读的顺序使得IO和传输并行起来是性能优化的关键.

为了把IO和传输并行起来,本文提出了一种流式数据预读方法.和原有的预读策略相比最主要的区别在于适当调整了客户端部分发送预读请求和收取预读数据的顺序,当需要接收当前预读数据的时候,不必等收完数据再发下一次预读请求,而是提前先发送下一次预读请求再去接收当前预读的数据,这样就可以让数据节点在向客户端传输数据的时候能够进行下一次的磁盘IO,这样,数据节点的IO和传输就完全并行起来了,实现了延迟隐藏.具体过程如下:

1) 应用通过客户端读取第1次读请求的数据后发送第1次预读请求.

2) 当有应用再对当前的块有读请求的时候,客户端首先会检查当前的内存缓冲区中是否包含所需要的数据,如果有,则直接从内存缓冲区中把数据返回给用户;如果没有则检查之前发送的预读请求是否已经包含了当前所需读的数据,如果是,则首先向数据节点发送下一次预读请求,然后再把上一次预读请求的数据收上来,并返回给应用;否则说明该次读请求不是一个顺序的读请求.

3) 重复第2)步.

图4是对比BlueOcean的预读方法和流式数据预读方法.

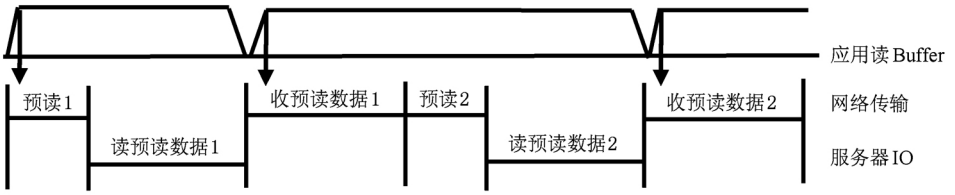
从上面的对比图可以明显地看到,客户端收上一次预读数据的时候,下一次的预读请求已经发出,数据节点可以一边做IO,一边把数据传给客户端,这样IO和网络传输就完全并行起来了.

2.3.2 提前发送预读请求

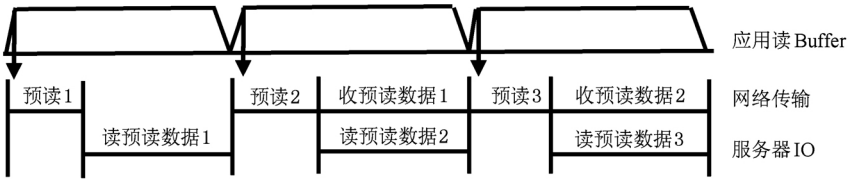
从图4可以看到,当应用读完整个内存缓冲区之后,发现其中没有所需的数据了,才去收当前预读数据.这样当从内存缓冲区里读不到数据的时候,应用就不得不阻塞来等待收取上一次预读的数据.这

个问题在流式数据预读方法中依然存在.

本文通过提前发送预读请求的方法解决了这个问题.对内存读缓冲区 Buffer 设置一个阈值[7],阈值指示了当应用读 Buffer 读到某个百分比时,就开始允许执行下一次的预读.比如设置阈值为 50%,当应用发送读请求的范围在当前的 Buffer 内并且超过了 Buffer 的 50%的时候,则触发下一轮的预读.先发送下一次预读请求,再收取上一次的预读数据.这样,由于把预读请求的发送提前了,应用总是可以从 Buffer 中读到所需要的数据.如图 5 所示.



(a) 优化前的BlueOcean的预读策略



(b) 采用流式数据预读方法优化后的BlueOcean预读策略

图 4　BlueOcean 优化前后对比图

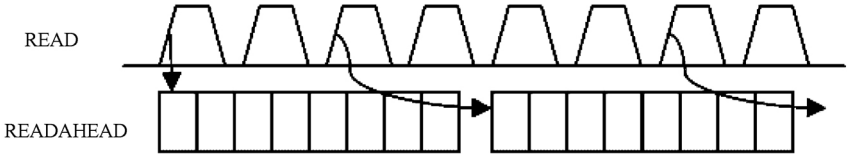


图 5　设置 Buffer 阈值 50%时的预读过程

3　性能测试

3.1　测试平台环境

测试系统由 1 台高性能管理节点,3 台高性能存储节点以及 10 台普通 PC 客户端组成.

每个数据节点挂载 4 块 4TB 硬盘.所有节点全部通过千兆交换机互联.使用 iozone 通过调整不同参数进行测试.BlueOcean 系统的部署如图 6 所示:

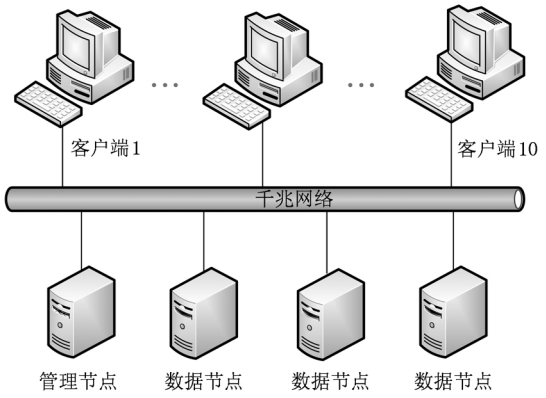


图 6　系统部署图

3.2　测试内容

例 1.比较采用流式预读方法前后系统吞吐量.

测试集:文件大小 8 GB,客户端数量 1,BlueOcean 副本数量 3,iozone 块大小 4 KB,16 KB,64 KB,256 KB,1 024 KB,设置 Buffer 阈值为 1/2.测试结果:

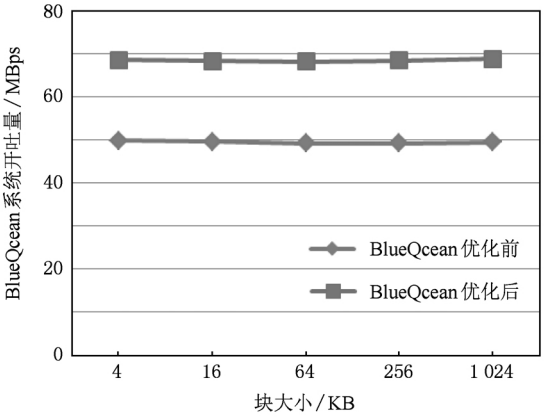


图 7　单台客户端下 BlueOcean 优化前后系统性能对比

例 2.比较设置不同 Buffer 阈值时的系统吞吐量.

测试集:文件大小 8 GB,客户端数量 1,BlueOcean

副本数量 3,iozone 块大小 64 KB,调整 Buffer 阈值为 0,1/8,2/8,3/8,4/8,5/8,6/8,7/8,1 测试结果:

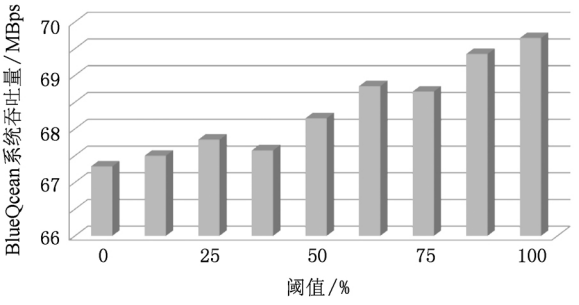


图 8 不同 Buffer 阈值时的系统吞吐量

例 3. 比较多台客户端优化前后的系统吞吐量.

测试集:文件大小 8 GB,副本数量为 3,iozone 块大小 64 KB,Buffer 阈值 1/2,调整客户端数量 1～10.

测试结果:

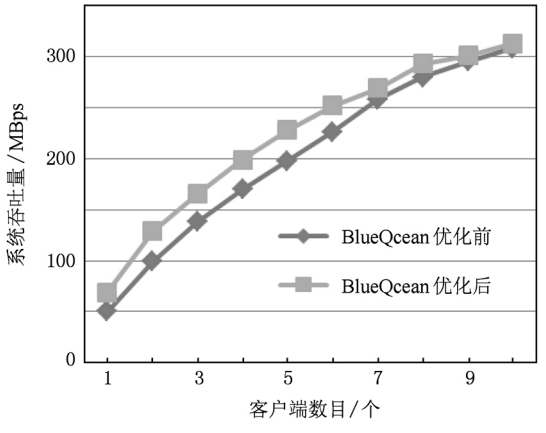


图 9 多客户端读不同文件优化前后性能对比

3.3 结果分析

1) 采用流式数据预读方法优化后系统吞吐量有明显提升,块大小对性能影响不大.

2) 性能随 Buffer 阈值的增加总体呈上升趋势,这表明了较晚的发送下一次预读请求确实有助于服务器端消除多个请求导致的磁头干扰现象.

3) 随着客户端数目增加,系统总体吞吐量大体呈线性增长,流式数据预读方法的作用依然明显.

4 小 结

本文介绍了 BlueOcean 分布式文件系统,并对其客户端和服务器端的读交互过程进行了详细的分析,指出了影响 BlueOcean 系统顺序读性能的关键因素,提出并实现了流式预读策略并进行了对比测试. 测试表明流式预读确实提高了顺序读性能.

本文提出的流式预读方法完全可以应用在其他分布式文件系统或网络文件系统.

参 考 文 献

[1] Chen Yong, Roth P C. Collective Prefetching for Parallel I/O Systems // Proc of IEEE PDSW'10. Piscataway, NJ: IEEE, 2010

[2] Ellard D, Seltzer M. NFS tricks and benchmarking traps // Proc of the Annual Conf on USENIX Anuual Technical Conf. Berkeley, CA: USENIX Association, 2003: 101-114

[3] Wu Fengguang. Sequential file prefetching in Linux // Advanced Operating Systems and Kernel Applications: Techniques and Technologies. Hershey, PA: IGI Global, 2010: 218-261

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

[5] Shvachko K, Kuang H, Radia S. The Hadoop Distributed File System //Proc of IEEE MSST'10. Piscataway, NJ: IEEE, 2010

[6] Srirams T, Blake H. Kosmos Distributed Filesystem. [2011-05-19]. http://kosmosfs.sourceforge.net/

[7] 吴峰光, 奚宏生, 徐陈锋. 一种支持并发访问流的文件预取算法. 软件学报, 2010, 21(8): 1820-1833

闫 鹤 男,1986 年生,硕士研究生,主要研究方向为海量存储、数据保护和备份.

李小勇 男,1972 年生,博士,副教授,硕士生导师,ACM、中国计算机学会会员,主要研究方向为海量存储、高性能网络、嵌入式系统(xiaoyongli@sjtu.edu.cn).

胡 鹏 男,1987 年生,硕士研究生,主要研究方向为分布式系统(hupeng@sjtu.edu.cn).

刘海涛 男,1976 年生,博士,讲师,中国计算机学会会员,主要研究方向为存储系统(htl@sjtu.edu.cn).