Journal of Computer Applications
计算机应用,2020,40(9):2586－2593
ISSN 1001-9081
CODEN JYIIDU
2020-09-10
http://www.joca.cn

文章编号:1001-9081(2020)09-2586-08
DOI:10.11772/j.issn.1001-9081.2020010121

分布式存储系统中的日志分析与负载特征提取

苟子安[1],张　晓[1,2*],吴东南[1],王艳秋[1]
(1.西北工业大学 计算机学院,西安 710129;　2.工信部大数据存储与管理重点实验室(西北工业大学),西安 710129)
(*通信作者电子邮箱zhangxiao@nwpu.edu.cn)

摘　要:对运行在文件系统上的工作负载进行分析有助于优化分布式文件系统的性能,且对构建新型存储系统至关重要。由于工作负载的复杂性和规模多样性的增加,使用基于直觉的分析来显式地捕获工作负载踪迹的特征是不完备的。针对这一问题,提出了一个分布式日志分析与负载特征提取模型。首先,从分布式文件系统日志中根据关键字抽取出与读写相关的信息;其次,从统计与时序两方面对负载特征进行描述;最后,分析基于负载特征进行系统优化的可能。实验结果表明,提出的模型具有一定的可行性与准确性,且可以较为详细地给出负载统计与时序特征,具有低开销、高时效、易于分析等优点,可以用来指导具有相同特征的工作负载的合成、热点数据监测、系统的缓存预取优化。

关键词:分布式文件系统;日志分析;负载特征;时序特征;性能优化
中图分类号:TP311　　**文献标志码:**A

Log analysis and workload characteristic extraction in distributed storage system

GOU Zi'an[1], ZHANG Xiao[1,2*], WU Dongnan[1], WANG Yanqiu[1]
(1. *School of Computer Science, Northwestern Polytechnical University, Xi'an Shaanxi* 710129, *China*;
2. *Key Laboratory of Big Data Storage and Management, Ministry of Industry and Information Technology* (*Northwestern Polytechnical University*), *Xi'an Shaanxi* 710129, *China*)

Abstract: Analysis of the workload running on the file system is helpful to optimize the performance of the distributed file system and is crucial to the construction of new storage system. Due to the complexity of workload and the increase of scale diversity, it is incomplete to explicitly capture the characteristics of workload traces by intuition-based analysis. To solve this problem, a distributed log analysis and workload characteristic extraction model was proposed. First, reading and writing related information was extracted from distributed file system logs according to the keywords. Second, the workload characteristics were described from two aspects: statistics and timing. Finally, the possibility of system optimization based on workload characteristics was analyzed. Experimental results show that the proposed model has certain feasibility and accuracy, and can give workload statistics and timing characteristics in detail. It has the advantages of low overhead, high timeliness and being easy to analyze, and can be used to guide the synthesis of workloads with the same characteristics, hot spot data monitoring, and cache prefetching optimization of the system.

Key words: distributed file system; log analysis; workload characteristic; timing characteristic; performance optimization

0　引言

随着企业数据量的快速增长,大规模的数据处理成为一个具有挑战性的问题,尤其是大数据的高性能存取,直接影响着对上层的服务质量。Hadoop分布式文件系统(Hadoop Distributed File System, HDFS)[1]是Hadoop项目的核心子项目,是分布式计算中数据存储管理的基础。它具有高容错性、高可靠性、高可扩展性、高吞吐率等特点,为超大数据集的应用处理和存储带来了许多优势。分析Hadoop集群上工作负载的特性是提高分布式文件系统性能的关键。针对不同类型应用的访问特征,分布式存储系统可选择不同的缓存策略或数据布局方式来进行性能优化[2]。

然而,Hadoop集群上的工作负载分析,特别是在大型的生产环境中,主要局限于集群的负载特征采集[3]。本文提出了一种离线的负载特征采集和分析方法,通过对HDFS中各个节点日志的关键信息进行提取和处理,来描述运行在其上的多客户端和多个应用的负载特征。由于MapReduce将任务分布在不同节点运行,任务和数据的分布,以及规模和调度的变化都会影响工作负载的特征[4],因此基于静态分析获取工作负载特征的方法是不准确的,并且在混合多种应用的集群

收稿日期:2020-02-13;**修回日期:**2020-05-05;**录用日期:**2020-05-10。
基金项目:国家重点研发计划项目(2018YFB1004400);北京市自然科学基金-海淀原始创新联合基金资助项目(L192027)。
作者简介:苟子安(1996—),男,陕西西安人,硕士研究生,主要研究方向:分布式文件系统性能优化、缓存优化、文件预取、机器学习;　张晓(1978—),男,河南南阳人,副教授,博士,CCF高级会员,主要研究方向:计算机存储系统、云计算、大数据存储与管理;　吴东南(1996—),男,陕西宝鸡人,硕士研究生,主要研究方向:分布式文件系统性能优化;　王艳秋(1995—),女,河南南阳人,硕士研究生,主要研究方向:分布式文件系统性能优化、大数据存储与管理。

中静态分析的方法存在更多的问题[5]。从节点日志收集到的动态数据足以准确描述工作负载,并提供应用程序输入/输出(Input/Output, I/O)工作负载的估计和预测。通过日志分析获取负载特征具有3个优势:1)低开销,采用离线分析方式,不会占用HDFS的I/O带宽等资源,因此不会影响到上层应用的性能,数据采集和分析的成本较低;2)高时效,HDFS的I/O等信息会同步的反映到日志上,因此对日志的实时分析具有一定的时效性;3)易于分析,基于日志的方法对负载分析提供了一定的便捷,例如对于汇聚起来的HDFS集群访问信息,在日志中可以很容易地根据网际互连协议(Internet Protocol,IP)地址来对客户端进行区分等。

本文的主要贡献如下:

1) 建立了一个分布式日志数据提取框架,通过在元数据节点和数据节点实时的日志处理,来提取I/O操作相关的关键信息。

2) 建立了一个通用的模型对提取出的I/O信息进行分析,以描述负载统计和时序特征,并通过与现有的benchmark对比验证了模型方法的可行性与准确性。

3) 说明了利用负载统计和时序特征进行负载均衡、智能预取等HDFS优化的可行性。

此外,本文提出的日志分析与负载特征提取方法还可用于任务调度、缓存管理、数据分布的优化,也可以用于HDFS集群性能实时监控、定位热点数据与节点等[6]。

1　背景

1.1　HDFS架构

HDFS是一种主/从架构,如图1所示,一个HDFS集群由一个元数据服务器(NameNode, NN)和多个数据服务器(DataNode, DN)组成。其中,NameNode执行例如打开、关闭、重命名等文件系统命名空间的相关操作、控制客户端对文件的访问和维护数据块到DataNode的映射关系等。此外,集群中的每一个数据服务器通常是一个DataNode,负责为文件系统客户端的读取和写入请求提供服务,同时处理NameNode发来的数据块创建、删除和复制等指令。

HDFS为了保证可靠性,需要对数据进行冗余备份,通常是三副本策略,即存储一个文件块的同时还存储两个冗余块。客户端在读取文件时,首先会联系NameNode获取文件的元数据,包括文件块信息以及每个块的存储信息等,随后客户端会在三副本所在DataNode中选择一个物理拓扑上最近的节点,与其建立传输控制协议(Transmission Control Protocol, TCP)连接并进行读取。而对于文件的写入,客户端首先需要联系NameNode,为待写入的文件块在命名空间中创建元数据信息,随后,NameNode向客户端返回的元数据中包含应持久化存储这个文件块的三个DataNode的相关信息,客户端与之建立相应的pipeline,并将文件块写到管道流中的第一个DataNode上,具体在写入时传输的单位是packet,第一个DataNode收到packet后会传给管道流中后续DataNode,并等待确认字符(ACKnowledge character,ACK),以此类推,当客户端接收到最后一个packet的ACK后,标志着这个块的写入完成,并开始下一个块的传输。

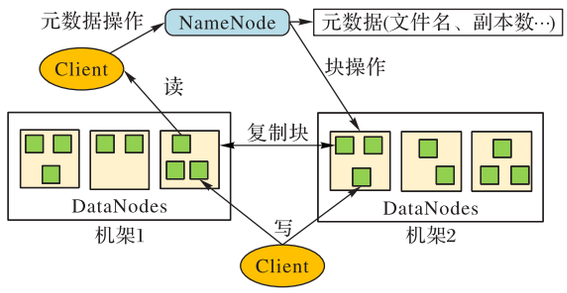

图1　HDFS架构

Fig. 1　HDFS architecture

1.2　Hadoop log

Hadoop的日志有很多种,主要分为Hadoop系统服务输出日志和MapReduce程序输出日志两大类,Hadoop系统服务输出日志是指诸如NameNode、DataNode、ResourceManager等系统自带的服务输出的日志,MapReduce程序输出日志则包括作业运行日志和任务运行日志[7]。

在Hadoop系统服务输出日志中,文件系统元数据日志位于NameNode节点,客户端对分布式文件系统命名空间的所有修改均可以在该日志中查看到,例如,在写入一个文件时需要在命名空间中创建对应的元数据信息,因此日志中可以查看到写入该文件的客户端IP,文件所有块的标识(Identity Document, ID)、大小以及分布等。此外,客户端对文件的读写操作也可以在元数据日志中查看到,例如,客户端读取文件时通过open操作来获取元数据,日志中则可以看到open操作对应的客户端IP。文件系统数据日志位于各个DataNode节点上,该DataNode上所有数据块的I/O信息均可以从日志中查看到,例如数据块的详细读写信息,包括块操作的源和目的IP地址、被操作的块ID和大小以及操作的持续时间等。分布式文件系统客户端将日志输出在其被调度到的Hadoop节点上,日志中主要包含客户端的相关配置等基础运行信息,以及一些读写时的详细信息等,例如读取时联系NameNode获取的文件块分布等元数据信息。由此可见,分布式文件系统客户端对整个Hadoop集群的操作基本可以反映在Hadoop系统服务输出日志中,因此从分布式节点日志中收集的数据足以表征客户端和集群的工作负载特征,并提供应用程序I/O工作负载的估计和预测。

此外,Hadoop日志库将日志分为5个级别,分别为DEBUG、INFO、WARN、ERROR和FATAL。这5个级别对应的日志信息重要程度依次从低到高,Hadoop只输出级别不低于设定级别的日志信息。即级别设定为DEBUG时,这5个级别的日志信息都会被输出。

2　相关工作

已有研究证明,深入了解I/O工作负载特征对于分布式系统的资源调度与性能调优至关重要[8-10],文献[11-13]从后端磁盘阵列(Redundant Arrays of Independent Disks,RAID)控制器或文件系统I/O控制器采集数据,得到应用程序的I/O负载特征。这种采集方式无法很好地应用到Hadoop等分布式环境中,因为实际生产环境的集群规模往往很大,基于硬件存储设备的负载特征采集开销较大、成本较高。基于访问trace的分析方法成本较低,并且证明是有效的,文献[14]分析了清华大学校园云存储系统Corsair的文件系统快照和5个月的访问trace,以研究学生团体对该云存储系统使用的负载特征,包括

文件大小、类型和读写比等。但是目前在诸如Hadoop等分布式环境中，对其底层文件系统的工作负载研究还不够深入，尽管最近的一些研究[15-18]对Hadoop生产环境的trace进行采集，并对I/O工作负载和数据分布进行了表征，但仅仅与现有一些服务的工作负载进行了比较，得出了启发式的规律，或是用于合成基准测试进行性能评估，没有考虑对系统进行性能优化。文献[19-22]使用基于trace的分析方法获取负载特征并指导集群的优化，但是其特征的采集要么通过对Hadoop JobTracker的trace进行处理，以获取job的递交时间、完成时间以及job结构等信息，要么通过对Hadoop YARN日志进行处理，提取有关job和task的统计信息。对特征的采集仅仅专注于应用层上，负载的描述也是关于job的信息，没有关注上层应用对底层分布式文件系统的影响，在负载的描述上可能不十分地准确与全面。此外，这些研究的trace采集时间也长至几周到几个月不等，对系统的优化没有即时的帮助，无法适应工作负载的变化，并且研究也是先有了trace才分析得到负载特征，这样得出的结论和该trace的相关性较大，系统的优化十分局限。

本文旨在以低开销的trace分析方法，先建立负载分析模型，再对分布式系统日志进行处理。首先，系统的调优不局限于某一具体的trace，也无需长时间的采集；其次，模型可以根据集群的负载以固定的时间间隔运行，也可以按需运行，其输出结果可以对HDFS做到及时的反馈，以适应工作负载的变化；最后，负载特征的描述采用文件系统层级的信息，与应用层相比在负载的描述上更加全面，对分布式存储系统的优化更具有意义。这些在先前的研究工作中是没有的。

3 设计与实现

3.1 模型架构简述

模型架构如图2所示，共分为两个模块：数据收集器和数据分析器。数据收集器从NameNode与DataNode的日志中根据关键字抽取出与读写相关的日志记录，并根据这些日志记录的数据结构提取读写原始数据。数据分析器对这些读写原始数据进行分析与计算得到负载特征，并据此进行负载描述与HDFS优化。具体的，数据分析器的主要功能又分为两部分：用于指导负载均衡的负载统计分析与用于指导智能预取的负载时序分析。本文提出的模型利用数据库进行中间数据交互，对HDFS日志提供的信息进行分析，将分析结果再反馈给HDFS进行优化，形成了一个闭环正反馈系统。

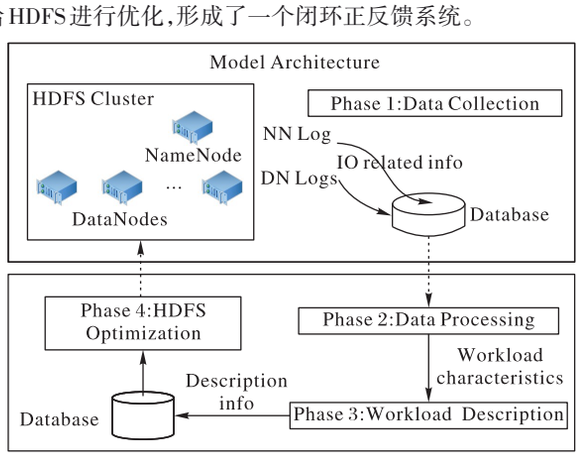

图2 模型架构

Fig. 2 Model architecture

3.2 日志数据结构

本文用于描述负载特征的原始数据均采集自Hadoop分布式节点日志，通过设置关键字来对日志文本进行处理，找出包含有关读写信息的日志记录，并对这些结构化日志记录中的数据进行提取以作后续分析，下面说明数据收集器选取的每一个关键字并简单描述其对应的内容。

cmd=open NameNode日志关键字、DEBUG级别、文件打开操作时生成，记录了有关客户端读取的详细信息，包括时间戳、执行读操作的客户端IP、打开文件的绝对路径等。

NameNode. create NameNode日志关键字、DEBUG级别、文件写入前在命名空间创建元数据时生成，记录了有关客户端写入的详细信息，包括时间戳、写入文件的绝对路径、执行写操作的客户端IP等。

HDFS_READ DataNode日志关键字、DEBUG级别、块读取时生成，记录客户端读取该块的详细信息，包括时间戳、读取块的客户端IP、读取字节数(通常略大于块的数据大小)、操作持续时间等。

HDFS_WRITE DataNode日志关键字、INFO级别、块写入时生成，记录pipeline中该块写入的详细信息，包括时间戳、pipeline上游节点IP、写入字节数(通常是块的数据大小)、操作持续时间等。

此外，还有一些关键字，例如Block* allocate日志中包含文件名与块ID列表，op：AddBlockOp日志中包含客户端IP与其写入块的ID列表，这些对负载特征的提取也有一定的用处，可以帮助从日志数据中构建一些必要的映射关系，因此本文在数据收集器中也加入了对它们的采集。

本研究基于DEBUG日志级别，而Hadoop默认的日志级别为INFO，因此数据收集器的工作需要将Hadoop日志级别修改为DEBUG，这可能会造成集群性能的下降，本文将在4.2节详细讨论其影响。需要注意的是，可以通过将DEBUG级别的日志在源码上改为INFO级，来避免对性能可能造成的影响，但是本文没有采用这种设计，原因是这样的设计在将模型移植到其他的集群上工作时，需要对集群源码修改并进行编译，操作较为复杂，使模型的可移植性变差，而修改集群日志级别仅仅需要更改配置文件中的参数即可，操作方便。此外，对源码中日志级别的修改，一定程度上会影响调试过程，违背代码编写的初衷，因此本文采用了修改集群日志级别这种源码无关的设计。

3.3 原始数据采集

模型设计为离线的工作方式，可以在一天内以固定的时间间隔运行，其间隔可以根据系统负载情况进行适当的调整，也可以按需运行，分析特定应用执行时间范围内的日志信息。利用3.2节描述的NameNode和DataNode日志关键字，并行地对集群各个节点日志目录下的系统服务输出日志进行文本分析，可以跟踪并获取到具体的I/O相关信息，将这些原始数据保存于数据库中以做后续使用，至此便完成了模型的数据收集阶段。此外，Hadoop系统服务日志会不断生成，并且旧的历史日志会以特定格式重命名并备份，因此日志量可能会较大，本文模型会记录各个节点上次的日志处理位置，以实现数据的增量收集，提高处理效率。模型运行时有两个输入参数，分别是负载分析的开始和结束时间戳，后续的数据处理阶段

会根据这个时间戳从数据库提取原始数据,这样可以根据需要分析某个特定时段的负载特征。

数据收集器根据3.2节描述的日志记录结构,从NameNode日志中提取出客户端的读写信息并存储到数据库中。这些读写日志记录是客户端混合的,后续需要按IP来进行区分,但考虑客户端数目较多可能需要大量的表去存储,因此本文将不同客户端的读写原始数据按时序存入数据库的同一张表中。该表包含五个字段:操作时间戳、客户端IP、操作名、操作文件名称和该文件的大小。需要注意的是,NameNode日志中无法提取出有关文件大小的信息,但是为了便于后续的负载特征计算,这里设计了这个保留字段。

在DataNode端,按时序提取出该节点的块读写信息并存入数据库中,每个DataNode维护一张表来保存该信息。该表包含四个字段:操作时间戳、操作名、操作块ID和操作数据大小。对于HDFS写操作,操作数据大小即为该数据块的大小,利用这一点,通过日志数据建立块ID到块大小的映射以及文件名到块ID的映射来获取文件大小,以填充上述保留字段。当然,使用HDFS的du子命令也可以获取文件大小,但是本研究的目的是设计一个离线分析工具,在线对HDFS进行du操作,当文件数目较多时会对集群的性能造成影响。对于HDFS读操作,操作数据大小偏大于该数据块的实际大小,因为在块读取时,HDFS会为数据块生成校验和信息,因此这个操作数据大小仅可用来计算和读取量有关的负载特征。最后,整个系统模型统一维护一张表,该表仅有两个字段:Hadoop设备IP和该节点提取的最后一条日志记录的时间戳,下次该节点直接提取该时间戳之后的日志记录中的数据,以加速数据收集过程。

3.4　负载统计分析

负载统计分析的目的是对HDFS集群进行负载均衡,因此,需要了解客户端与集群的负载统计情况,利用这些统计数据去充分描述负载。在负载统计特征选取时,对于客户端来说,选取的特征应尽可能反映客户端的基本读写类型及其对集群读写负载的影响等。而对于集群,应尽可能反映整个集群的读写承载情况以及各个节点的负载变化等。在客户端与集群两方面进行负载统计特征选取,有助于较为准确、全面地了解集群的承载情况及其可能发生的变化。此外,在进行特征选取时,要尽可能从不同的角度去进行描述,以免冗余。具体负载统计特征的选取及其描述见表1。

表1　负载统计特征

Tab. 1　Workload statistical characteristics

对象	负载统计特征	描述
客户端	读/写文件大小	读/写的平均文件大小,反映客户端倾向于读/写大文件还是小文件
	读/写量	读/写的数据总量,反映客户端对集群的读/写影响
	读写比	读写操作数的比值,反映客户端操作类型是读密集还是写密集
	IOPS	每秒的平均读写次数,反映客户端的I/O操作密度
	读/写分布	客户端的读/写分布,反映该客户端对各个DN的读/写影响
集群	读/写文件大小	集群读/写的平均文件大小,反映集群倾向于读/写大文件还是小文件
	吞吐率	每秒的平均读写量,反映整个集群的读写承载情况
	DN读/写量	每个DN的读/写数据总量,反映各个DN的读/写承载情况

为了获取上述负载统计特征,数据分析器首先需要处理NameNode日志中提取出的客户端文件读写信息,区分客户端并获取不同客户端的读写时序流。其次,处理从DataNode日志中提取出的块读写信息,获取文件块在集群中的读写分布。最后也是最重要的,需要根据日志中提取出的原始数据建立一些必要的映射关系,例如文件名到块ID的映射、文件到文件大小的映射等,这些映射关系就像桥梁,连接NameNode与DataNode日志中分析出的信息,从而得到负载统计特征。通过建立映射来进行分析的优势是可以准确得出负载统计特征的数值,例如HDFS的写入中pipeline上游节点接收数据后会写往下游节点,因此下游节点的DataNode日志中该数据块写入的源IP即为上游节点的IP地址,但是这次写入操作并不能计算在上游节点客户端的写入中,而通过NameNode日志可以分析出具体某一个客户端写入的所有文件名称,通过文件名到块ID的映射以及每个DataNode上的块写入信息,可以准确得到这个客户端在集群上的写入分布以及写入总量。

数据分析器可以通过分析日志直接得到不同客户端的文件读写流、文件读写分布、文件读写量以及不同DataNode上的读写承载,以这些负载特征数据为基础,数据分析器可以计算输出表1所描述的所有负载统计特征,例如通过客户端的文件读写流可以计算出该客户端的每秒读写次数(Input/Output Operations Per Second, IOPS)等。此外,数据分析器可以通过分析集群的读写承载来进行负载均衡,例如某个节点读取承载过多时,说明该节点存储的文件块较多,可以通过HDFS的balancer子命令使集群数据自动迁移达到均衡,一定程度上降低该节点的读取负担[23],另一方面,可以结合客户端的负载统计特征,当某个客户端在该节点读取较多时,上层MapReduce应用程序可以将该读取任务适当调度到文件块所在的其他节点上,以此来达到负载均衡。

3.5　负载时序分析

负载时序分析的目的是进行智能预取,鉴于HDFS的读取等操作大多是在文件层级上的,因此本文选择客户端的文件读取流作为负载时序特征的一部分,同时数据分析器也会更加细粒度地给出客户端按时序、跨节点的块读取流,以适应更多的智能预取算法。为了得到客户端的负载时序特征,数据分析器同样需要用到NameNode日志中提取出的客户端文件读写信息,以从不同客户端的读写时序流中获取其文件读取流。而按时序、跨节点的块读取流则可以从DataNode的块读取日志中分析得到,如3.2节中所述,“HDFS_READ”日志中详细包含了某个客户端从某个节点上读取的块ID及时间戳。

数据分析器可以通过对客户端负载时序特征进行文件访问模式分析得出文件预取信息,并显式调用HDFS集中式缓存

管理指令对文件块进行缓存[24]，这是HDFS自身提供的一种显式缓存机制，允许用户根据存储路径指定HDFS中的文件或目录并将其缓存在内存中，NameNode通知存储该文件数据块的DataNode将数据块缓存在进程Java虚拟机(Java Virtual Machine，JVM)的堆外内存中，MapReduce应用程序可以根据从NameNode查询到的元数据获取缓存块的分布，随后再次发生缓存文件的读取时，可以将客户端调度到缓存块所在的节点上，通过调用HDFS源码中提供的零拷贝函数，直接从内存中获取所需数据块，一定程度上提升客户端的读取效率。

此外，数据分析器也可以结合所获取的负载统计与时序特征来进行智能预取，例如可以通过负载统计特征来对客户端进行分类，为不同访问模式的客户端选取不同的缓存策略，对HDFS分布式日志的后续跟踪分析中，如果发现了客户端访问模式的匹配，则可以用相应的预取模型对其负载时序特征中的文件访问流进行分析，从而得出需要主动缓存的预取文件。在本文提出的模型中，数据分析器除了文件级别的负载时序特征，还给出了更加细粒度的按时序、跨节点的块级别负载时序特征，因此使得智能预取的策略选择更加灵活和普适，并且缓存预取不会仅仅局限于文件层级。

4 实验评估

4.1 实验基础环境

实验搭建的Hadoop集群共由6台服务器组成，包括1个NameNode、4个DataNode和1个没有存储数据的DFSClient节点，Hadoop版本为2.9.0。NameNode所在的服务器的型号为Sugon S650，CPU为AMD Opteron 6212，CPU频率为2.6 GHz，核心数为16核，内存大小为64 GB，硬盘大小为1.7 TB。DataNode与DFSClient的节点配置与NameNode相比，除了内存大小不同以外其他均相同，每个DataNode的内存为40 GB，DFSClient则为48 GB。除此之外，机器的操作系统为Ubuntu16.04，linux内核版本为4.4.0-124-generic。

实验使用了两个benchmark，分别为TestDFSIO和HiBench。其中：TestDFSIO是Hadoop自带的基准测试组件，其jar包版本同Hadoop，也为2.9.0；HiBench为Intel开发的大数据基准测试套件，实验使用的HiBench版本为7.0。此外，本章所有的实验，benchmark均在DFSClient节点上执行，以模拟更加真实的场景。

4.2 模型可行性分析

本文模型要求Hadoop集群日志设置为DEBUG级别，而集群在不同日志级别下性能可能会有所不同，因为日志级别设置得越低，集群在进行读写时就会输出越多的日志，因此需要在两种不同的日志级别下对集群进行性能分析。使用TestDFSIO进行文件的读写测试，因为多个小文件的读写会进行多次的元数据查询、命名空间创建等，会输出更多的日志，因此本实验主要进行多个小文件的读写性能测试。控制文件大小为128 MB，读写文件数从4增大到64，测试结果见图3，从中可以看出，更改Hadoop集群的日志级别，对MapReduce多个小文件的读写任务执行时间影响不大，因此提出的模型具有一定的可行性。

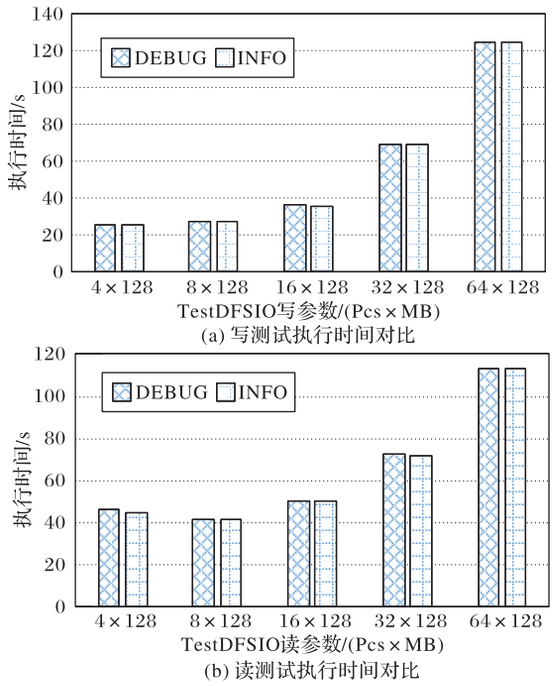

图3 不同日志级别读写性能对比

Fig. 3 Comparison of read and write performance at different log levels

4.3 模型准确性验证

现使用TestDFSIO进行准确性验证，为了充分验证本文模型的准确性，分别从以下两种情况来进行文件的读写测试：1)控制文件大小不变，改变读写文件数；2)控制读写文件数不变，改变文件大小。先执行写入任务再执行读取任务，将这视为一组测试，每组测试结束后运行模型进行日志分析，模型分析读写量与benchmark理论读写量的对比结果如图4所示，其中：图4(a)控制文件大小为128 MB，分别改变读写文件数为8、16、32和64；图4(b)控制读写文件数为10，分别改变文件大小为128 MB、256 MB、512 MB和1 024 MB。

从结果可以看出，模型准确地分析出了执行MapReduce任务后的集群读写量，均略大于理论值是因为TestDFSIO在进行实际数据的写入前，会由一个客户端向集群io_control目录下写入控制文件，在这之后其他客户端会先从这个目录下读取控制文件，然后再向io_data目录下写入实际的数据。对于读取操作，也是先需要读取控制文件，然后再进行数据文件的读取。此外，HDFS在读取文件块时，每512 B会生成一个4 B的校验和，这意味着每读一个128 MB的块会额外读取1 MB的校验和，模型可以分析出这些额外的读写，充分说明了利用日志分析来进行负载特征提取的准确性。模型的执行过程几乎没有时延，可以很好地适应大数据分析的实时性。因此，从实验结果来看，具有一定的准确性与时效性。

4.4 负载特征的获取

下面使用模型来分析HiBench微基准Sort的执行过程，说明模型在具体应用下获取到的负载统计与时序特征。微基准数据量参数设置为“huge”，实际进行排序的数据量大小约为3 132.73 MB。需要注意的是，benchmark虽然是在DFSClient节点执行的，但是TestDFSIO和HiBench均为MapReduce应用，也就是说，集群存在多个客户端进行并发读写，这些客户端分布在集群的不同DataNode上。

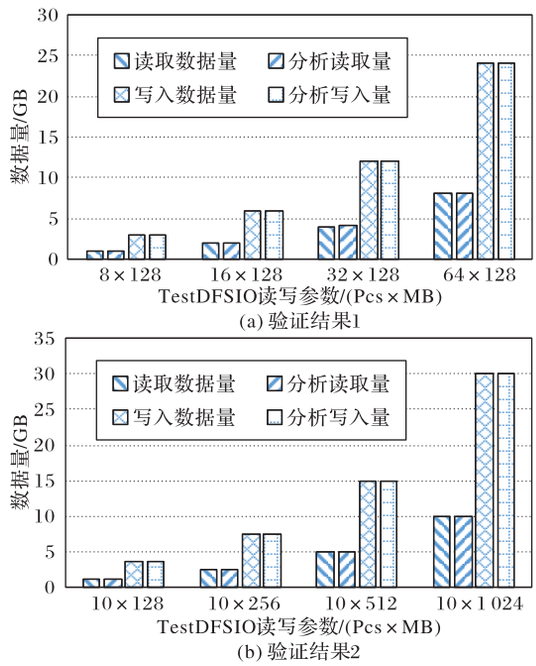

图4　不同读写参数下的准确性验证结果

Fig. 4　Accuracy verification results at different read and write parameters

首先,对于各个客户端的负载统计特征,模型分析结果如表2、3所示,其中日志跟踪时间为93. 39 s。其次,对于HDFS集群的负载统计特征,跟踪时间内模型的分析结果如下:平均读文件大小136. 65 MB;平均写文件大小1 174. 78 MB;集群吞吐率236. 39 MB/s。此外,各个DataNode节点的读写承载见图5。

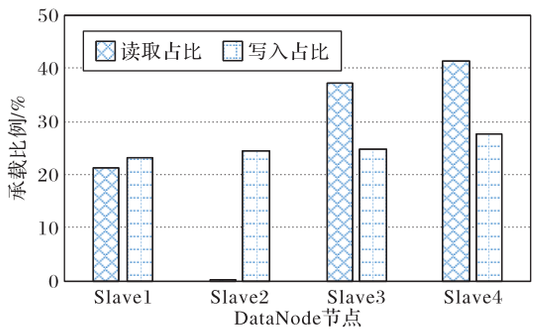

图5　集群各节点读写承载情况

Fig. 5　Read and write load status of different nodes in the cluster

表2　客户端负载统计特征分析结果

Tab. 2　Client workload statistical characteristics analysis results

客户端IP	读/写文件大小/MB	读/写量/MB	读写比	IOPS
172. 19. 2. 172	137. 81/1 175. 28	689. 04/4 701. 12	1. 25∶1	0. 10
172. 19. 2. 173	1. 34/1 174. 77	1. 34/3 524. 32	1∶3	0. 05
172. 19. 2. 174	135. 65/1 174. 45	1 492. 13/4 697. 79	2. 75∶1	0. 16
172. 19. 2. 185	137. 32/1 174. 64	1 098. 54/5 873. 19	1. 6∶1	0. 14

表3　客户端读写分布　　单位:MB

Tab. 3　Client read and write distribution　　unit:MB

客户端IP	读取量				写入量			
	Slave1	Slave2	Slave3	Slave4	Slave1	Slave2	Slave3	Slave4
172. 19. 2. 172	680. 24	0. 38	0. 51	7. 91	1 567. 04	927. 04	1 039. 58	1 167. 46
172. 19. 2. 173	0. 17	1. 01	0	0. 16	783. 17	1 174. 77	655. 19	911. 18
172. 19. 2. 174	9. 55	0	1 224. 2	258. 38	918. 34	1 046. 77	1 565. 93	1 166. 74
172. 19. 2. 185	7. 65	0	0	1 090. 88	1 061. 73	1 430. 54	1 423. 19	1 957. 73

至于负载时序特征,分为文件层级的客户端文件读取流和块层级的按时序、跨节点的客户端块读取流两部分,模型的部分分析结果如表4、5所示。

表4　客户端(172. 19. 2. 172)文件读取流

Tab. 4　Client (172. 19. 2. 172) file reading stream

时间戳	文件HDFS路径
20200109170049	/HiBenchData/HiBench/Sort/Input/part-m-00000
20200109170049	/HiBenchData/HiBench/Sort/Input/part-m-00001
20200109170059	/HiBenchData/HiBench/Sort/Input/part-m-00002
20200109170059	/HiBenchData/HiBench/Sort/Input/part-m-00001
20200109170108	/HiBenchData/HiBench/Sort/Input/part-m-00002

4. 5　模型的应用

在缓存算法、调度策略的研究中,或是机器学习、深度学习等模型的训练中,往往需要trace集来辅助研究,以作为实验的一个手段。在缺乏理想、可用的开源trace的情况下,需要使用严谨的实验方法来制作trace集。本节描述在HDFS缓存预取研究中模型在trace集制作上发挥的作用,以说明模型在实际工作中的应用价值,及其能为后续分析提供什么样的数据。

表5　客户端(172. 19. 2. 172)块读取流

Tab. 5　Client (172. 19. 2. 172) block reading stream

时间戳	块ID	DataNode
20200109164526	blk_1073747220_6397	Slave1
20200109164527	blk_1073747221_6398	Slave1
20200109164527	blk_1073747223_6400	Slave1
20200109164605	blk_1073747247_6424	Slave1
…	…	…
20200109164737	blk_1073747241_6418	Slave1
20200109165614	blk_1073747226_6403	Slave2
20200109165457	blk_1073747222_6399	Slave3
20200109165655	blk_1073747226_6403	Slave3
20200109165904	blk_1073747222_6399	Slave4
20200109170049	blk_1073747227_6404	Slave4
20200109170052	blk_1073747246_6423	Slave4

本文使用SWIM (Statistical Workload Injector for MapReduce)[25]在实验集群上复现来自Facebook生产系统的真实工作负载。该工具包含5个trace集,本文使用的trace集为Facebook上包括600个节点的Hadoop跟踪历史记录的公开部分,完整trace从2009年5月到2009年10月共6个月,大约包含100万个job。该trace集主要包含job的递交时间、map

和 reduce 阶段的数据量等，而 HDFS 缓存预取研究需要不同节点的文件读取时序流，因此需要将该 trace 从应用层转化为文件系统层，以提供研究所需要的数据。

本文首先使用 SWIM 工具以较低的开销在实验 Hadoop 环境上重现上述 Facebook trace，该工具依照原 trace 中的时间和数据大小不断向 Hadoop 递交 job。根据需要，本实验将 job 执行所需的数据平均文件大小设置为 32 MB，总文件数为 1 501。在脚本执行完毕后，即获取到了 job 执行完毕后 GB 级别的 Hadoop 日志，接着使用模型对这些日志进行负载时序分析，以获取不同节点的读取时序流，部分分析结果如表 6 所示。根据统计结果显示，仅前 1 000 个 job 运行时间约为 9 h 38 min，访问文件次数约为 21 000 个，足以证明文本提出的模型在复杂应用中的工作价值。

表 6 执行 Facebook trace 时 172. 19. 2. 172 部分文件读取流

Tab. 6 Some 172. 19. 2. 172 file reading stream when executing Facebook trace

时间戳	文件 HDFS 路径
2020-04-02 19:27:14,779	/Input/part-00139
2020-04-02 19:27:14,807	/Input/part-00140
2020-04-02 19:27:20,431	/tmp/hadoop-yarn/staging/root/. staging/job_1585826523351_0004/job_1585826523351_0004_1. jhist
2020-04-02 19:27:20,525	/tmp/hadoop-yarn/staging/root/. staging/job_1585826523351_0004/job_1585826523351_0004_1_conf. xml
2020-04-02 19:29:43,075	/tmp/hadoop-yarn/staging/root/. staging/job_1585826523351_0007/job. jar
2020-04-02 19:29:43,162	/tmp/hadoop-yarn/staging/root/. staging/job_1585826523351_0007/job. xml
2020-04-02 19:29:46,381	/tmp/hadoop-yarn/staging/root/. staging/job_1585826523351_0007/job. split

5 结语

本文面向 HDFS，提出了一个基于分布式文件系统日志的负载特征提取模型，模型分为两个模块：数据收集器和数据分析器。数据收集器从 NameNode 与 DataNode 的日志中根据关键字抽取出与读写相关的日志记录，并根据这些日志记录的数据结构提取读写原始数据。数据分析器对这些读写原始数据进行分析与计算得到负载特征，并据此进行负载描述与 HDFS 优化。实验结果表明，本文提出的模型具有一定的可行性与准确性，且可以较为详细地给出负载统计与时序特征，具有低开销、高时效、易于分析等优点，可以用来指导合成具有相同特征的工作负载、热点数据监测、系统的缓存预取优化等。

本文的研究重点在于探究以什么样的关键字可以从分布式存储系统日志中准确地跟踪到 I/O 信息，并且提出一个通用的模型来对集群在不同工作负载下的特征进行描述，最后开发出实际的系统原型来为下一步的工作打下基础。因此本文在 3. 4 与 3. 5 节提出描述负载特征的模型后只是简单给出了利用分析结果进行 HDFS 性能优化的思路，具体优化策略的研究将作为未来的工作。此外，在后续的研究中发现有利用 Web 日志构建预测模型进行 Web 文档预取的应用[26]。利用日志的更新进行模型自适应调参，也显现出了在不断变化的工作负载下保持较好的预测准确率的优势[27]。但是，缺乏基于分布式文件系统日志进行预取模型构建的研究。本文的工作是后续拓展工作的基础，基于本文提出的日志分析与负载特征提取方法，下一步研究将利用分布式文件系统的历史日志来构建基于机器学习的预取模型，并且使用后续日志的增量更新来自适应地对模型进行调参，达到对 HDFS 热点文件的预取，提升 Hadoop 上层应用性能的目的。

最后关于模型分析结果的可视化展示，目前原型的设计是在获取负载特征后，对于负载统计特征使用 Python 的可视化库将其展示出来，包括客户端的统计特征，集群的读写文件大小、吞吐率以及各个 DataNode 的读写承载分布等。而关于负载时序特征，鉴于客户端文件和块的读取流内容较多，并且这部分结果将作为智能预取模型的输入，因此本文目前将其以文本的形式进行保存。此外，现有的可视化原型系统还可通过 AJAX（Asynchronous Javascript And XML）等技术实现基于 Web 的监控系统，在此不再赘述。

参考文献（References）

[1] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop distributed file system [C]// Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies. Piscataway: IEEE, 2010: 1-10.

[2] LIAO J, TRAHAY F, XIAO G, et al. Performing initiative data prefetching in distributed file systems for cloud computing [J]. IEEE Transactions on Cloud Computing, 2017, 5(3): 550-562.

[3] REN Z, XU X, WAN J, et al. Workload characterization on a production Hadoop cluster: a case study on Taobao [C]// Proceedings of the 2012 IEEE International Symposium on Workload Characterization. Piscataway: IEEE, 2012: 3-13.

[4] ABAD C L. Big data storage workload characterization, modeling and synthetic generation [D]. Champaign-Urbana, IL: University of Illinois at Urbana-Champaign, 2014: 35-50.

[5] DITTRICH J, QUIANÉ-RUIZ J A. Efficient big data processing in Hadoop MapReduce [J]. Proceedings of the VLDB Endowment, 2012, 5(12): 2014-2015.

[6] 金国栋，卞昊穹，陈跃国，等. HDFS 存储和优化技术研究综述 [J]. 软件学报, 2020, 31(1): 137-161. (JIN G D, BIAN H Q, CHEN Y G, et al, Survey on storage and optimization techniques of HDFS[J]. Journal of Software, 2020, 31(1): 137-161.)

[7] LIN X, WANG P, WU B. Log analysis in cloud computing environment with Hadoop and Spark [C]// Proceedings of the 5th IEEE International Conference on Broadband Network and Multimedia Technology. Piscataway: IEEE, 2013: 273-276.

[8] ZHANG B, KŘIKAVA F, ROUVOY R, et al. Self-balancing job parallelism and throughput in Hadoop [C]// Proceedings of the 16th IFIP WG 6. 1 International Conference on Distributed Applications and Interoperable Systems, LNCS 9687. Cham: Springer, 2016: 129-143.

[9] 于晓龙. MapReduce 模型在 Hadoop 实现中计算资源利用率分析和多作业批调度优化 [D]. 济南：山东大学, 2016: 26-38. (YU X L. MapReduce model for computing resource utilization analysis and multi-job batch scheduling optimization in Hadoop implementation [D]. Jinan: Shandong University, 2016: 26-38.)

[10] ZACHEILAS N, KALOGERAKI V. Pareto-based scheduling of

MapReduce workloads [C]// Proceedings of the 19th IEEE International Symposium on Real-Time Distributed Computing. Piscataway: IEEE, 2016: 174-181.

[11] KIM Y, GUNASEKARAN R. Understanding I/O workload characteristics of a Peta-scale storage system [J]. Journal of Supercomputing, 2015, 71(3): 761-780.

[12] LIU Y, GUNASEKARAN R, MA X, et al. Automatic identification of application I/O signatures from noisy server-side traces [C]// Proceedings of the 12th USENIX Conference on File and Storage Technologies. Berkeley: USENIX Association, 2014: 213-228.

[13] GUNASEKARAN R, ORAL S, HILL J, et al. Comparative I/O workload characterization of two leadership class storage clusters [C]// Proceedings of the 10th Parallel Data Storage Workshop. New York: ACM, 2015: 31-36.

[14] LIU S, HUANG X, FU H, et al. Understanding data characteristics and access patterns in a cloud storage system [C]// Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. Piscataway: IEEE, 2013: 327-334.

[15] REN Z, SHI W, WAN J, et al. Realistic and scalable benchmarking cloud file systems: practices and lessons from AliCloud [J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(11): 3272-3285.

[16] REN K, GIBSON G, KWON Y, et al. Hadoop's adolescence; a comparative workloads analysis from three research clusters [C]// Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis. Piscataway: IEEE, 2012: 1452-1453.

[17] REN Z, XU B, SHI W, et al. iGen: a realistic request generator for cloud file systems benchmarking [C]// Proceedings of the 2016 IEEE 9th International Conference on Cloud Computing. Piscataway: IEEE, 2016:343-350.

[18] BOCCHI E, DRAGO I, MELLIA M. Personal cloud storage: usage, performance and impact of terminals [C]// Proceedings of the 2015 IEEE 4th International Conference on Cloud Networking. Piscataway: IEEE, 2015:106-111.

[19] ABAD C L, ROBERTS N, LU Y, et al. A storage-centric analysis of MapReduce workloads: file popularity, temporal locality and arrival patterns [C]// Proceedings of the 2012 IEEE International Symposium on Workload Characterization. Piscataway: IEEE, 2012: 100-109.

[20] KAVULYA S, TAN J, GANDHI R, et al. An analysis of traces from a production MapReduce cluster [C]// Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. Piscataway: IEEE, 2010: 94-103.

[21] REN Z, WAN J, SHI W, et al. Workload analysis, implications, and optimization on a production Hadoop cluster: a case study on Taobao [J]. IEEE Transactions on Services Computing, 2014, 7(2): 307-321.

[22] DIMOPOULOS S, KRINTZ C, WOLSKI R. PYTHIA: admission control for multi-framework, deadline-driven, big data workloads [C]// Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing. Piscataway: IEEE, 2017:488-495.

[23] Apache Hadoop Software Library. HDFS users guide [EB/OL]. [2020-01-05]. http://hadoop. apache. org/docs/r2. 9. 0/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide. html#Balancer.

[24] Apache Hadoop Software Library. Centralized cache management in HDFS [EB/OL]. [2020-01-05]. http://hadoop. apache. org/docs/r2. 9. 0/hadoop-project-dist/hadoop-hdfs/CentralizedCacheManagement. html.

[25] CHEN Y, ALSPAUGH S, GANAPATHI A, et al. SWIM: Statistical Workload Injector for MapReduce [EB/OL]. [2020-03-23]. https://github. com/SWIMProjectUCB/SWIM/wiki.

[26] HUANG Y F, HSU J M. Mining Web logs to improve hit ratios of prefetching and caching [J]. Knowledge-Based Systems, 2008, 21(1): 62-69.

[27] WANG H, YI X, HUANG P, et al. Efficient SSD caching by avoiding unnecessary writes using machine learning [C]// Proceedings of the 47th International Conference on Parallel Processing. New York: ACM, 2018: No. 82.

This work is partially supported by the National Key Research and Development Program of China (2018YFB1004400), the Beijing Natural Science Foundation-Haidian Original Innovation Union Foundation (L192027).

GOU Zi'an, born in 1996, M. S. candidate. His research interests include distributed file system performance optimization, cache optimization, file prefetching, machine learning.

ZHANG Xiao, born in 1978, Ph. D., associate professor. His research interests include computer storage system, cloud computing, big data storage and management.

WU Dongnan, born in 1996, M. S. candidate. His research interests include distributed file system performance optimization.

WANG Yanqiu, born in 1995, M. S. candidate. Her research interests include distributed file system performance optimization, big data storage and management.