

多模态作物表型数据分布式存取方法研究

郝子超^{1,2} 赵向宇² 潘守慧^{2,3} 刘东明³ 王开义²

(1. 沈阳农业大学信息与电气工程学院, 沈阳 110866; 2. 北京市农林科学院信息技术研究中心, 北京 100097;
3. 北京派得伟业科技发展有限公司, 北京 100097)

摘要: 高通量作物表型采集设备的飞速发展育种与栽培研究提供了现代化数据采集手段,同时催生了海量多模态、非结构化表型数据。传统结构化数据存储模式已难以满足此类数据的高效存取需求。为此,提出基于分布式技术的混合存取框架,利用 HBase 和 HDFS 构建结构化和非结构化融合存储引擎,集成客户端缓存和 Redis 缓存设计高效检索机制,并对核心问题进行了优化:针对原生 HDFS 存储表型数据的固有缺陷,设计基于模式聚合的 MCH 存储框架,通过将表型数据按模态分类合并存储,并使用双层哈希技术构建局部索引,有效降低 NameNode 内存压力,同时提高单模态数据的访问效率与存储空间利用率。面向高并发数据读取场景,构建基于数据热度的双层缓存机制,通过元数据分层缓存优化热点数据读取效率,创新提出结合访问频率和时间特性的数据热度评估模型,有效提高缓存命中率。试验结果表明:本文提出的分布式存取方法在数据为 1.0×10^5 份时,NameNode 内存占用量较最佳原生方案 (SequenceFile) 降低 31.2%,检索时间较最佳原生方案 (MapFile) 降低 25.4%,为海量多模态表型数据的存储和检索提供了技术支撑。

关键词: 多模态作物表型数据; 分布式存取; 文件合并; 双层缓存机制

中图分类号: TP392 **文献标识码:** A **文章编号:** 1000-1298(2026)01-0051-11



OSID:

Distributed Access Method for Multimodal Crop Phenotypic Data

HAO Zichao^{1,2} ZHAO Xiangyu² PAN Shouhui^{2,3} LIU Dongming³ WANG Kaiyi²

(1. College of Information and Electrical Engineering, Shenyang Agricultural University, Shenyang 110866, China
2. Information Technology Research Center, Beijing Academy of Agriculture and Forestry Sciences, Beijing 100097, China
3. Beijing PAIDE Science and Technology Development Co., Ltd., Beijing 100097, China)

Abstract: The rapid development of high-throughput crop phenotyping acquisition equipment has provided modern data collection means for breeding and cultivation research, while spawning massive multi-modal and unstructured phenotypic data. Traditional structured data storage models can no longer meet the efficient access requirements of such data. A hybrid access framework was proposed based on distributed technology, which used HBase and HDFS to build a structured and unstructured fusion storage engine, integrated client-side cache and Redis cache to design an efficient retrieval mechanism, and optimized core issues: aiming at the inherent defects of native HDFS in storing phenotypic data, a modal aggregation-based MCH storage framework was designed. By classifying and merging phenotypic data according to modalities and constructing local indexes by using double-layer hashing technology, it effectively reduced NameNode memory pressure while improving access efficiency and storage space utilization of single-modal data. For high-concurrency data reading scenarios, a double-layer cache mechanism based on data popularity was constructed. It optimized hot data reading efficiency through metadata hierarchical caching and innovatively proposed a data popularity evaluation model combining access frequency and time characteristics, which effectively improved cache hit rate. Experimental results showed that when the data scale was 1.0×10^5 , the proposed distributed access method reduced the NameNode memory occupancy rate by 31.2% compared with the optimal native solution (SequenceFile), and the retrieval time by 25.4% compared with the optimal native solution (MapFile),

providing technical support for the storage and retrieval of massive multi-modal phenotypic data.

Key words: multimodal crop phenotypic data; distributed access; file merging; two-level cache mechanism

0 引言

作物表型是基于基因-环境互作关系对遗传信息进行选择性表达的结果,包括在不同尺度和模式下的遗传、变异及环境响应数据,是作物基因图谱在时间维度与三维空间中的动态表达,对其深入分析可以促进农作物的育种和栽培,以提高作物产量和适应更严峻的生存环境^[1-3]。

传统的作物表型数据多为结构化数据,可利用常见的关系数据库来进行管理。但是,近年来,表型组学相关技术和研究手段高速发展,高通量作物表型设备(如流水线式表型平台、轨道式表型平台、机器人表型平台、无人机表型平台等)的应用日益广泛,由此产生了数量巨大、尺度多维、格式多样的非结构化表型数据,包括可见光、多光谱、荧光、热成像等^[4-7]。这些数据为解析基因型-表型-环境互作机制提供了多维度观测窗口,但同时也对数据的管理提出了全新挑战。由于非结构化表型数据在维度、格式、规模和采集频率等方面存在显著差异,因此基于关系型数据库的传统数据存储方式,无法满足大规模非结构化作物表型数据存储和快速数据检索需求^[8-9]。随着大数据的高速发展,以 Hadoop 生态为代表的分布式技术凭借低成本、高可靠、大容量等优点在存储领域得到广泛应用,为海量作物表型数据的存储提供了解决方案^[10-12]。但是高通量表型设备产生的表型数据,单个文件大小一般在 14 MB 以内,远小于 HDFS 默认的 BLOCK 块阈值 128 MB,若直接使用原生 Hadoop 处理这些数据会带来显著的技术瓶颈:大量原始文件直接上传至 HDFS 会产生过多的 BLOCK 块和对应的元数据信息,极大地增加了 NameNode 内存占用量,并造成存储空间的严重浪费;此外,高并发读取场景下,频繁的元数据请求会进一步加重 NameNode 负载,导致系统整体存取效率显著下降^[13-16]。

原生 HDFS 针对显著低于 BLOCK 块阈值文件的存取性能瓶颈问题,提供了 3 种解决方案,分别是 HAR 方案、SequenceFile 方案和 MapFile 方案^[17-18]。但这些解决方案无法满足多模态表型数据分布式存取的需求,尤其是表型数据持续生成并追加带来的存储碎片化和高重建开销。已有学者针对 HDFS 文件存取的性能问题开展了大量的研究工作,其主要围绕文件合并管理和缓存预取机制两方面来展开。

在文件合并方面,研究者通常先将原始文件合并后再存储以降低 NameNode 元数据信息占用压力,但具体实现方式略有不同,部分研究者将原始文件进行随机合并,并使用外部存储系统(如 HBase)来管理原始文件的索引信息^[19],而另一些研究者则先计算文件之间的相关性再进行归类合并^[20-21]。这些方法虽然在一定程度上降低了 NameNode 内存压力,但却存在对数据相关性考虑不足或计算开销过大的局限,降低了文件读写效率。在缓存预取方面,研究者提出了多种策略来应对 NameNode 的通信瓶颈,包括基于 HashMap 的缓存机制^[22]、基于 LRU 算法的缓存策略^[17]以及基于 LRFU 的本地缓存策略^[23]等,以减少与 NameNode 的通信次数,提高文件访问效率。但这些方法缓存命中率低。此外,为进一步提高文件的读取效率,一些学者在索引构建方面也进行了优化,通过使用可扩展哈希函数和完美哈希函数构建单层索引,对 Hadoop 存档文件进行优化^[24],从而提高了文件的读取效率。总体而言,以上方案均在各自针对的场景中取得了一定的进展,但却不具备普适性,无法满足作物表型数据的存取需求。

本文结合作物表型数据的特点和实际存取需求,在以上研究的基础上进行改进与优化,设计一种满足多模态表型数据整体存取需求的方案。为降低单模态表型数据检索时跨合并文件块读取概率,本方案将表型数据按模态分类合并存储且不需要提前计算文件之间的关联性,解决 NameNode 元数据内存占用大的问题并提升单模态表型数据的检索速度;同时,在文件合并过程中引入降序最佳适应算法(Best fit decreasing, BFD),以提高数据块的空间利用率;其次,利用双层哈希编码技术构建索引文件,并使用 HBase 来存储索引信息,以实现 O(1)时间复杂度内的索引信息读取;进一步,引入 ElasticSearch 支持多样化复杂查询;最后,设计一种基于数据热度的双层缓存机制,以提升表型热点数据的访问效率和缓存命中率,避免缓存污染。

1 材料与方法

本文从作物表型数据的特征出发,结合 Hadoop 平台,设计了一种基于分布式技术的海量表型数据存取优化方案,整体框架如图 1 所示。核心技术主要包括 4 部分,文件合并存储、双层哈希索引构建、

多样化检索以及缓存机制。

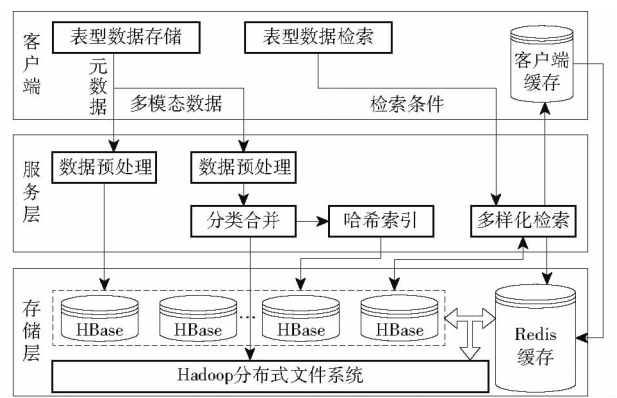


图 1 整体架构图
Fig.1 Overall architecture diagram

存储层是整体架构的最底层,其主要作用是持久化存储数据。包括分布式文件系统 HDFS、非关系型数据库 HBase 以及内存数据库 Redis。HDFS 用来存储非结构化多模态表型数据;HBase 用来存储表型数据所配套的元数据信息以及原始文件的局部索引信息等结构化数据;Redis 用来存储热点数据的全局索引信息。

服务层位于中间层,其主要是为上层的业务管理和下层的数据存取提供服务。服务层包括数据预处理模块、分类合并和索引构建模块、多样化检索模块。数据预处理模块主要是对采集的原始表型数据进行清洗、去噪以及标准化,为后面存储奠定基础。分类合并和索引构建模块用来解决 HDFS 处理原始文件时不足的问题,将表型数据按模态进行分类合并存储,同时采用可扩展哈希和 MWHC 哈希构建原始文件的索引信息,并将索引信息存入 HBase 中。针对表型数据的多样化检索,设计了基于 ElasticSearch 的多样化检索模块,通过将表型数据所配套的元数据信息从 HBase 中同步到 ElasticSearch,实现了基于特定检索条件的精确查询。

客户端是用户与系统的交互入口,负责接收原始的表型数据并执行业务逻辑与服务层紧密协作,共同完成数据从输入到存储、检索的全生命周期管理,最终向用户提供高效、可靠的表型数据存取服务。

1.1 基于 HDFS 的海量作物表型数据存储架构设计

针对 HDFS 处理表型数据时存在的 NameNode 元数据膨胀与存取效率慢等性能瓶颈问题,本文提出了一种基于模态分类和双层哈希的优化存储方案 (Modal Classification merging file with Hash index, MCH)。高通量表型设备采集的多模态数据实时上传至平台中,平台根据时间间隔和存储阈值,将原始

数据分批处理。同批数据按模态分类合并存储,并采用双层哈希建立数据索引存入 HBase 中,存储架构如图 2 所示。

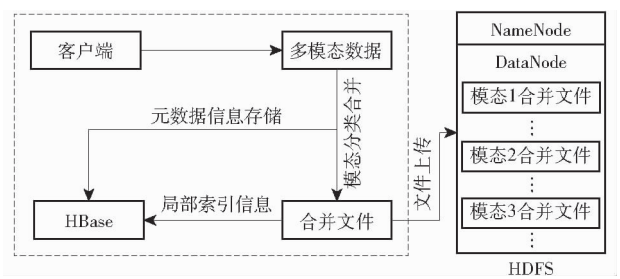


图 2 存储架构图
Fig.2 Storage architecture diagram

1.1.1 基于模态分类的降序最佳适应合并算法

高通量表型设备产生的作物表型数据主要包括可见光、多光谱、荧光、热成像、点云等多模态数据。如果直接将此类原始文件存储至 HDFS,会瞬间导致 NameNode 内存中元数据量急剧膨胀,造成严重的 NameNode 内存压力和响应延迟,因此需要先对其进行合并处理。

常用的文件合并策略是基于顺序合并的,然而这种合并策略所生成的合并文件块大小仍与数据块阈值有较大差距,且不可控,造成了 HDFS 数据块资源的显著浪费^[25]。另一方面,将不同模态的数据混合存储在同一数据块中,当研究人员查看某种作物在一段时间内的特定模态表型数据进行表型分析时,系统需要从多个不同的数据块中读取数据,这种跨合并文件块的读取方式,I/O 效率极其低下,严重降低了表型数据的分析效率。

为解决上述问题,本文提出一种基于模态分类的 BFD 文件合并策略。首先集成表型设备采集的数据分批按模态分类,然后采用 BFD 算法对分类后的各模态数据进行合并处理,具体流程如图 3 所示。

BFD 算法首先会判断上一批次数据的数据存储块是否存在剩余。如果存在,则对该数据块重建文件合并线程并加载到内存中,此时,会与剩余空间最大的合并文件块进行比较,当文件大小超过剩余空间最大的合并文件块时,则创建空的合并文件块,并将其放入其中,否则将其放入剩余空间与其最接近的合并文件块中。如果不存在,则会直接创建空的合并文件块,并依次选取 n 个总大小不超过数据块阈值的文件放入该合并文件块中,剩余数据执行上述存在时的合并步骤,直至所有文件均放入合并文件块后,策略执行结束,并将合并文件上传至 HDFS,同时建立该批次数据的索引。

1.1.2 双层哈希索引

上文提到的合并策略有效提升了数据块的空间

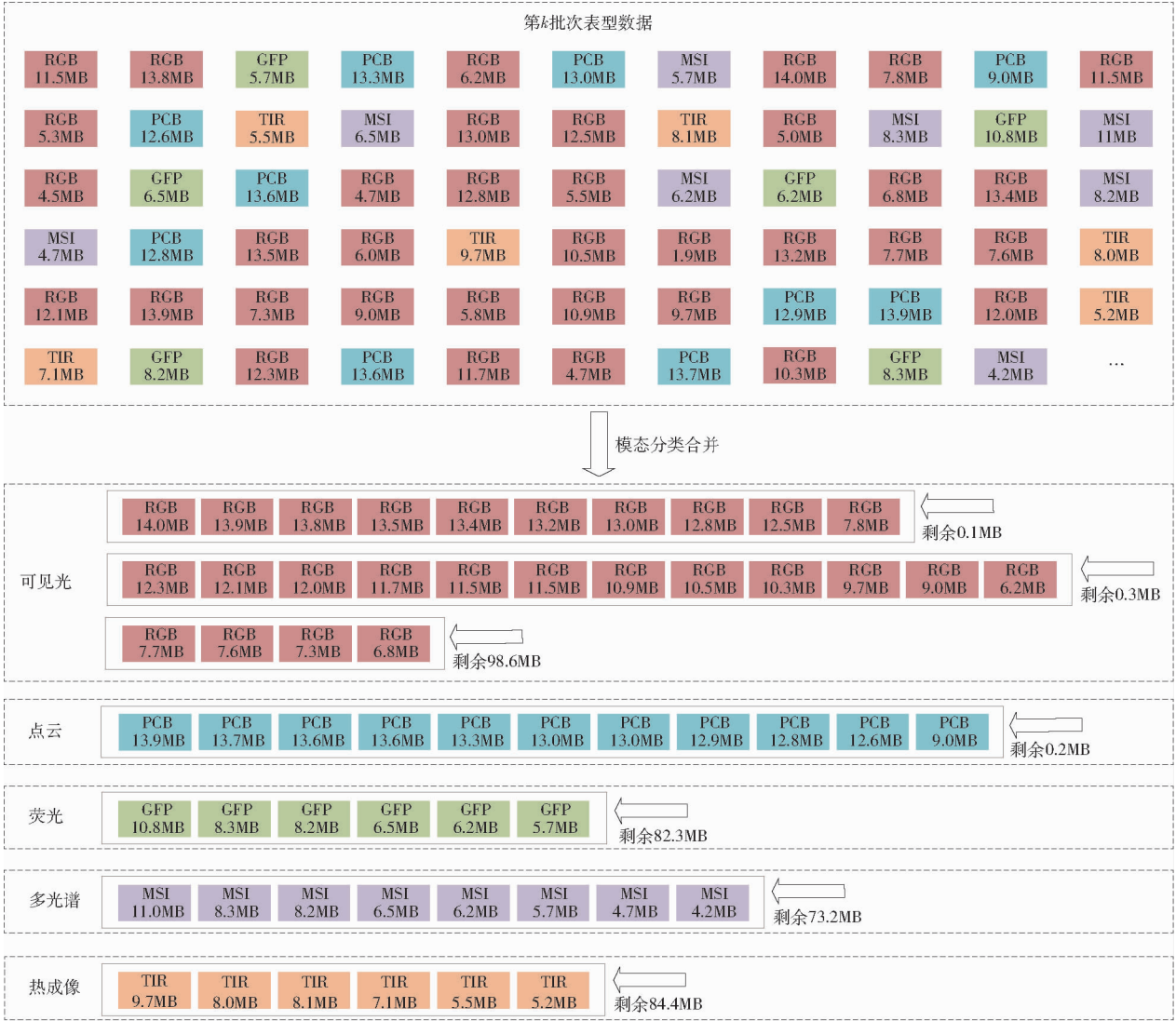


图 3 文件合并
Fig. 3 File merging

利用率,缓解了 NameNode 元数据存储压力,并降低了单模态数据的跨合并文件块读取几率。但合并后的文件缺乏原始文件到合并文件的映射关系,将迫使每次查询都进行代价高昂的全文件扫描,严重拖慢表型数据读取效率。为此,本文采用双层哈希技术为原始文件构建索引,可以直接定位目标原始文件在合并文件中的具体位置,从而避免了全文件扫描,有效提升数据读取效率。

将原始文件按模态进行分类合并存储,为了能够高效地从合并文件中获取目标原始文件,在每次合并的时候都需要保存原始文件的元数据信息(包括文件名哈希、合并文件 ID、偏移量、文件大小等),通过这些元数据信息即可快速地从合并文件中读取到目标原始文件。而这些元数据信息被保存在某一个 Index 文件中,为能够快速地获取目标原始文件的元数据信息,这里采用双层哈希函数来构建索引文件。

如图 4 所示,展示了双层哈希函数构建索引文件的具体流程:使用可扩展哈希表构建 n 个存储桶,每个桶中存储 k 个原始文件的元数据信息。同时,将每个桶中的文件元数据信息按照文件名的哈希字典序进行排序,再使用 MWHC 哈希函数存储每个文件元数据信息的位置信息,MWHC 哈希函数写在每个索引文件的开头,最后将索引信息存入 HBase 中。

本文使用 HBase 存储表型平台原始文件的元数据信息。HBase 由两个列族组成,第 1 个列族存储表型数据所配套的元数据信息(包括作物名字、品种、植株 ID、模态标识等)用于实现对特定数据的精确检索,第 2 个列族存储原始文件的局部索引信息(即原始文件到合并文件的映射关系)用于快速定位原始文件在合并文件中的具体位置。同时为保证局部索引信息的每部分都具有固定大小,将文件名用哈希代替,每个文件的索引信息共 24 bytes。

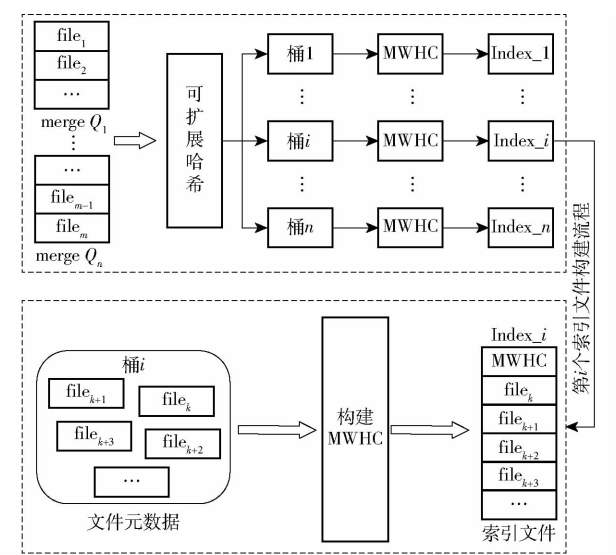


图 4 索引构建结构图

Fig. 4 Index construction architecture diagram

由于 HBase 可以实现高性能并发读写操作,因此能够进一步加快查询所需读取文件的索引信息,进而提高文件读取效率。在进行文件读取时,首先由列族 1 确定所要读取的特定表型数据,然后获取所对应列族 2 中的局部索引信息,从而直接定位原始文件在合并文件中的具体位置。

当文件进行追加时,首先会重新创建文件合并线程,并追加新增文件至对应模态类型的合并文件块中,然后使用可扩展哈希函数将文件索引放入对应的桶中;然后,在其对应的存储桶中,重新对索引信息按照文件名哈希字典序进行排序,并使用 MWHC 哈希函数存储索引位置信息;最后,将文件索引信息追加至 HBase 中。

1.2 基于 ElasticSearch 与双层缓存机制的表型数据检索与预取方案设计

本文使用 HBase 存储表型数据的元数据信息,

支持通过行键 (RowKey) 实现高效单点查询,但在多条件表型数据复合检索场景中,无明确行键信息,依赖全表扫描,查询效率明显下降。基于此,本文引入了 ElasticSearch 组件,为 HBase 设计了二级索引,实现了多条件复合查询,避免了全表扫描。同时,为实现高并发读取场景下,表型热点数据的低延迟访问需求,设计了一种基于数据热度的双层缓存机制,对关键索引信息进行分层缓存,有效提高了热点数据的访问效率。

1.2.1 基于 ElasticSearch 的多样化检索架构

在农作物表型组学研究中,通常需要通过作物名称、品种类型、植株 ID 等多条件复合检索表型数据,以满足多维度、多类型的作物表型数据对比分析的需求。为了高效地实现这种多维度、多类型的检索需求,本文提出基于 ElasticSearch 的多样化检索方案:将 HBase 中作物名称、品种类型、植株 ID 等核心检索字段同步至 ElasticSearch 建立索引,并将 HBase 的行键作为 ElasticSearch 文档的唯一标识符,通过 ElasticSearch 的多条件检索快速过滤到符合条件的 RowKey,进而在 HBase 中查询符合条件的结果,多样化检索架构如图 5 所示。

在检索时,客户端首先会向 ElasticSearch 发送多样化检索请求,ElasticSearch 接受请求并完成多样化检索后会返回给客户端对应的行键集合,此时客户端会根据行键列表到 HBase 中查询对应目标文件的局部索引信息,然后与 NameNode 交互获取所属合并文件索引,最后计算出目标文件的全局索引信息(包括原始文件文件名、偏移量、文件大小、合并文件 ID、DataNode 信息)并与 DataNode 交互获取数据。

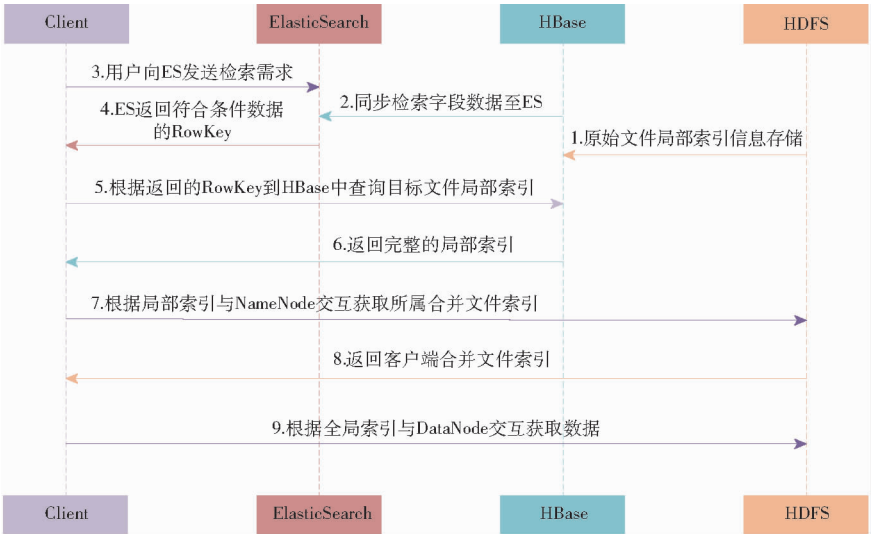


图 5 多样化检索架构图

Fig. 5 Diverse retrieval architecture diagram

1.2.2 基于数据热度的双层缓存机制

通过对上述检索流程分析可知,当批量读取表型数据文件时,会涉及多次与 NameNode 的通信,产生显著的延迟,不能满足实际检索中表型热点数据的低延迟访问需求。基于此问题,本文提出了一种基于数据热度的双层缓存机制。该机制的核心思想是主动且分层缓存关键索引信息,Redis 中持久化存储热点数据的全局索引信息,客户端缓存预取目标数据所属合并文件的索引信息,核心处理如下:

当检索命中 Redis 缓存时,直接根据目标数据文件的全局索引信息与 DataNode 通信获取数据,此时无需更新客户端缓存;当检索没有命中 Redis,但命中了客户端缓存时,客户端会根据得到的合并文

件索引信息与目标文件局部索引信息综合计算得到全局索引信息,然后与 DataNode 通信获取数据,并将所得目标文件的全局索引信息加载至 Redis 中;当上述两种缓存均未命中时,则通过传统方式读取数据,并将所得目标文件的全局索引信息加载至 Redis 中,目标文件所属合并文件的索引信息预取至客户端缓存中。

通过上述双重缓存机制,后续再次访问相同文件或同合并文件内的其他文件时,客户端无需频繁再与 NameNode 交互获取索引信息,可以直接从缓存中读取,显著降低了访问延迟,从而有效提升了文件读取效率,满足表型热点数据实际检索低延迟访问需求。图 6 所示为缓存机制处理流程图。

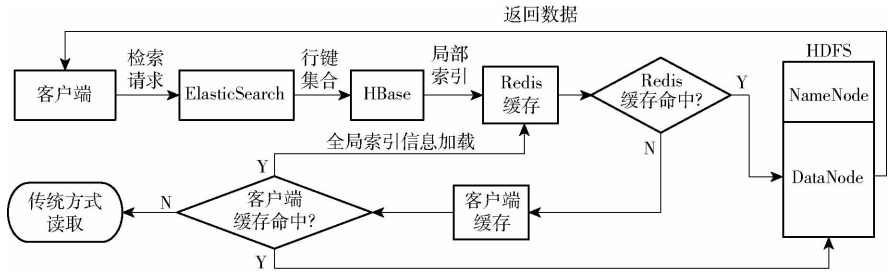


图 6 缓存机制处理流程图

Fig. 6 Caching mechanism architecture diagram

当 Redis 缓存或客户端缓存达预设的阈值后,均需要进行数据淘汰以维护缓存的有效性。客户端缓存中的索引数据在淘汰时更加注重访问频率的影响,因此直接采用了标准的 LFU 算法进行数据淘汰。对于 Redis 缓存中的索引数据,其有明显的热点动态变化特性,而传统 Redis 缓存策略往往采用简单的缓存数据淘汰算法,无法充分适应这种动态变化的特点。为解决这些问题,本文提出一种数据热度评估模型(Data heat evaluation model,DHE),该模型综合考虑了数据的访问频率与时间特性,实现了对数据热度的实时更新,当 Redis 缓存达预设阈值后,按照热度从低到高的顺序淘汰数据,从而有效提高了缓存的有效性和系统性能。

数据热度评估模型可表示为

$$H+=\frac{F}{\ln(C-L+1)+1}$$

式中 H ——数据热度

F ——访问频次

C ——系统当前时间,ms

L ——上次最近访问时间,ms

根据上述定义对模型进行初始化,将 F 初始值设为 1, C 和 L 的初始值设为系统当前时间。并不断更新每次检索后的 F 、 C 、 L 以及 H 值。当数据第

1 次放入 Redis 中时, H 值设为 0,此时无需根据热度评估模型进行计算,当后续再次访问相同数据时, F 进行加 1 操作, C 更新为当前系统时间,数据热度 H 根据热度评估模型进行累加计算更新,最后将 L 更新为当次访问的系统时间。同时,数据热度会随着访问频次的降低而衰减,当数据超过一定时间没有被访问时,其热度会大幅衰减,实现热点数据的动态更新。

2 试验结果与分析

2.1 试验环境搭建和数据准备

本试验使用 VMware 软件虚拟了 4 台 Ubuntu 服务器,并在上面配置了相关环境,搭建了 Hadoop 集群、HBase 集群、Zookeeper 集群、Redis 集群、ElasticSearch 集群,客户端使用 Java 语言,具体硬件软件信息见表 1。

本试验以 1.0×10^5 份作物表型数据作为基准数据集,涵盖可见光成像、荧光成像、热红外成像等非结构化数据文件。将全部文件进行随机混合后分为 6 组测试数据,文件数量依次为 1.0×10^4 、 2.0×10^4 、 4.0×10^4 、 6.0×10^4 、 8.0×10^4 及 1.0×10^5 。试验将从文件存取性能、数据块空间利用率、NameNode 内存空间占用以及双层缓存性能这 4 方面来评估本文所提出方案的性能。

表 1 试验环境配置

参数	数值/型号
CPU	Intel(R) Core(TM) Ultra7 155H 3.80 GHz
内存容量/GB	32
硬盘容量/TB	1
操作系统	Ubuntu - 18.04.6 - desktop
JDK	8u331 - Linux - x64
Hadoop	3.2.3
HBase	2.1.0
Zookeeper	3.7.2
Redis	5.0.4
ElasticSearch	7.10.1

2.2 文件存取性能对比

为验证 MCH 方案的有效性,使用原生 HDFS 方法、MapFile 方法、HAR 方法以及 SequenceFile 方法作为对比方案,评估了上述第 2 ~ 6 组文件上传至 HDFS 的写入性能。同时,MCH 为了提高表型数据的检索效率增加了 HBase 的写入,会降低部分写入性能,为定量评估该环节对存取性能的影响,增加不使用 HBase 存储索引的简化 MCH 方案(HN)作为对比方法。试验分别记录每个方案的上传时间,每个方案试验进行 3 次,取其平均值,试验结果如图 7a 所示。为了对比读取性能,特别测试了处理第 6 组文件时,上述 6 种方案的文件随机检索时间,试验结果如图 7b 所示。

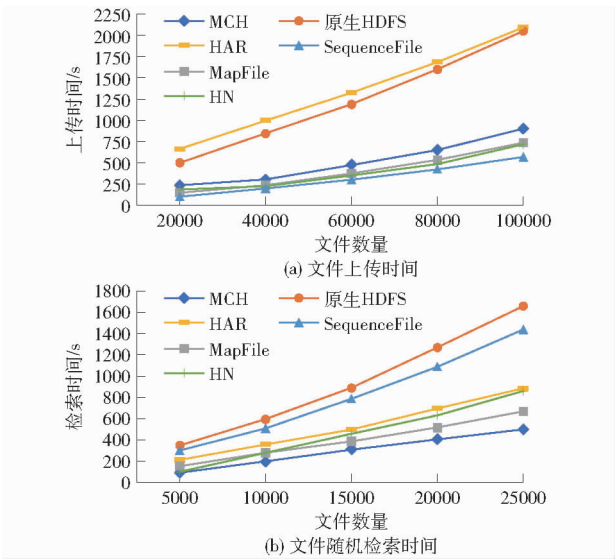


图 7 文件存取性能对比

Fig. 7 File access performance comparison

由图 7a 分析可知原生 HDFS 和 HAR 方法文件上传所用时间最多,本文方案略高于 SequenceFile 方法和 MapFile 方法,当文件上传数量达到 1.0×10^5 时,写入时间相比于原生 HDFS 降低 56%,原因主要在于:原生 HDFS 由于没有经过文件合并,直接

将大量原始文件上传至 HDFS 时,会产生高频次的 RPC 通信,显著增加 NameNode 的负载压力,进而增加了原始文件的上传时间,且随着文件数量的增加,劣势越来越明显。HAR 虽然经过了文件合并,但其将所有原始文件先上传至 HDFS 后再进行合并,且 HAR 文件使用 MapReduce 进行打包,需要消耗很多时间从 NameNode 中获取文件的元数据信息,然后再执行 map 任务,因此其写入速度甚至超过原生 HDFS。本文方案在写入文件时,需要将原始文件元数据信息额外写入到 HBase 中,因此增加了文件的写入时间,略高于 MapFile 方法和 SequenceFile 方法。虽然本文方案的写入效率不是最佳,但是本文方案直接从 HBase 中获取原始文件的索引信息,读取效率会大幅度提升。

由图 7b 分析可知原生 HDFS 和 SequenceFile 方案读取时间最慢,本文方案的读取时间最快,当随机读取 2.5×10^5 份文件时,其检索时间相比于 MapFile 方案降低 25.4%,原因在于:①本文方案和 HAR、MapFile、SequenceFile 方案均使用了文件合并策略,相比于原生 HDFS,在每次读取文件时减少了与 NameNode 的交互时间。②SequenceFile 方案没有建立索引,在每次读取原始文件时需要扫描全部的 SequenceFile 合并文件,因此其读取时间仅次于原生 HDFS 方案。③本文的文件合并方案在合并时考虑到了文件之间的相关性,将相同模态类型的原始文件合并到同模态类型的合并文件中,降低了跨合并文件块读取几率。④本文的文件合并方案使用单层索引创建索引文件,并将索引信息存储到了 HBase 中,可以直接定位原始文件。HAR 和 MapFile 方案虽然也创建了索引文件,但使用的是双层索引,增加了读取时间。综上所述,本文的文件合并方案读取性能最佳。

由图 7 整体分析可知,本文方案在文件写入时需要将索引信息存储至 HBase 中,额外的 HBase 存储操作增加了文件的写入时间。当文件数量为 2×10^4 份时,写入时间相比于不使用 HBase 时增加 54 s,当文件数量达到了 8×10^4 份时,额外写入时间趋于稳定,不再随着文件数量的增大而增大,约占整体写入时间的 26.2%。由于本文方案主要针对高通量表型设备集成应用时一次写入频繁读取的场景,因此数据的检索效率是需要关注的重点,通过 24.4% 平均写入效率的降低,有效提升了 54.5% 的平均检索效率。

2.3 空间占用对比

为综合评估本文所提出 MCH 方案在数据块空间利用率和 NameNode 内存空间占用方面的整体优

化效果,选取了上述第 2~6 组文件进行了两项关键指标的对比试验。

针对数据块空间利用率,将所述 5 组文件分别采用 MCH 方案、顺序合并策略、MapFile 合并策略、HAR 合并策略以及 SequenceFile 合并策略进行存储,分别对比在不同文件规模下不同策略的整体数据块空间利用率,试验结果如图 8a 所示。另一方面,为验证本文的 MCH 方案在降低 NameNode 内存占用方面的效果,选取了原生 HDFS、MapFile、HAR、SequenceFile 传统方案作为对照组,同样将所述 5 组文件分别采用这 5 种方案上传至 HDFS 集群。试验过程中,重点监测并记录各方案对应的 NameNode 编辑日志(edit_inprogress)所占内存空间增量变化。试验结果如图 8b 所示。

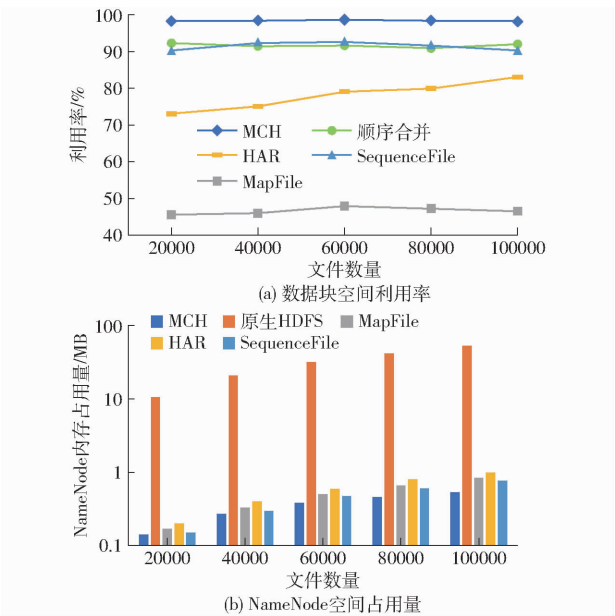


图 8 空间占用对比
Fig. 8 Space occupancy comparison

从图 8a 中可以看出,不同合并策略在数据块利用率方面存在显著差异。其中,MapFile 策略数据块空间利用率最低,HAR 策略处于中间水平,顺序合并策略和 SequenceFile 策略基本接近,而本文提出的合并策略效果最佳,整体数据块空间利用率维持在 98% 左右。原因在于顺序合并策略在存储时没有考虑到原始文件大小的因素,导致合并后的文件块内部产生了较多碎片,进而影响了整体的空间利用率。与之不同的是,本文合并策略基于 BFD 算法实现,有效减少了合并文件块内部碎片,使得整体表现更接近数据块阈值。另外,HAR 策略和 MapFile 策略由于其内部索引需要单独占用数据块空间,造成了数据块空间的极大浪费,因此整体空间利用率最低。

从图 8b 中可以看出,随着原始文件数量增加,

原生 HDFS 的 NameNode 内存占用量大幅增加,当达到 1.0×10^5 份原始文件时,内存占用量接近 70 MB。而其他方案由于采用了文件合并策略,均能有效控制内存消耗。其中,本文提出的文件合并方案表现最优,在所有测试场景下内存占用均处于最低水平,且随文件数量的增加优势更加明显,当文件规模达到 1.0×10^5 份时,相比于 SequenceFile 方案,内存占用量降低 31.2%。原因在于本文所提出的文件合并方案在合并过程中采用了 BFD 算法,使得每个合并文件尽可能合并更多的原始文件,相比于其他几种传统的合并方法一定程度上减少了合并文件的数量,进而 NameNode 管理的合并文件块元数据数量减少。这一结果表明了本文方法可以降低 NameNode 元数据存储压力。

2.4 双层缓存机制性能对比

为测试双层缓存机制的检索效率,选取了第 6 组文件,并采用 MCH 方案上传至 HDFS。在此基础上,分别模拟了双层缓存机制、仅有 Redis 缓存机制以及无缓存机制 3 种场景下的随机读取操作,并对比了这 3 种缓存方案的随机读取时间,试验结果如图 9a 所示。由图 9a 分析可得,Redis 缓存机制读取效率位于中间,本文所提出的双层缓存机制读取效果最佳,当文件读取个数达到 2.5×10^4 份时,相比于仅有 Redis 缓存机制,读取时间降低 36.2%。原因在于本文所提出的双层缓存机制不仅可以减少重复读取数据的检索时间,而且可以减少该数据所属合并文件块里其数据的检索时间,因此整体读取效率更高。

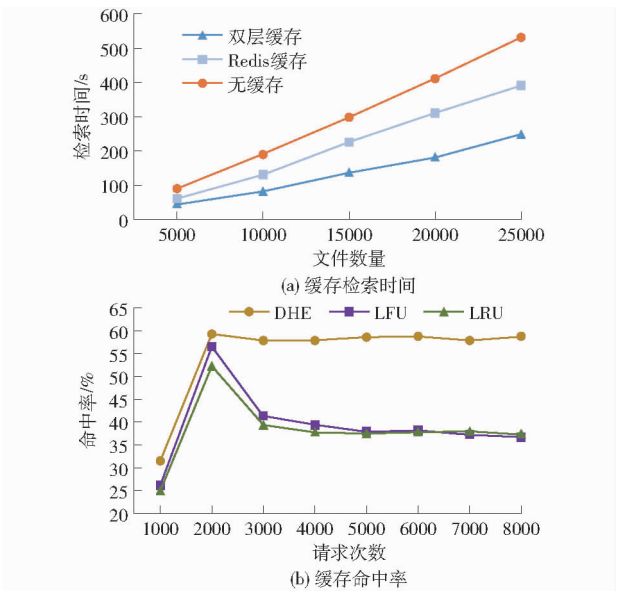


图 9 缓存机制性能对比

Fig. 9 Caching mechanism performance comparison

为了验证 DHE 策略的有效性,将其与传统的

LFU、LRU 算法进行了命中率对比。将第 1 组文件中 10% 的数据标志为热数据并上传至 HDFS 中。进行 8 000 次随机检索请求,在这 8 000 次检索请求中,有 20% 的概率访问冷数据,80% 的概率访问热数据。最后,统计每种数据淘汰策略前 1 000 次到 8 000 次中高频数据命中率,试验结果如图 9b 所示。由图 9b 分析可得,在随机检索请求次数为 2 000 时,LFU 算法和 LRU 算法的命中率均达到 55% 左右,后面开始下降。本文提出的 DHE 方案在命中率达极限值之后逐渐趋于平稳,整体命中率高于 LFU 算法和 LRU 算法。其原因在于相比于 Redis 所采用的单 LFU/LRU 算法,本文方案综合考虑了数据的访问频率与时间特性,能对新热点数据及时响应,快速提高其热度,同时又对高热度低频访问数据的热度周期性进行了衰减,进而更好地保证了热点数据的 Redis 留存率。

3 讨论

高通量作物表型平台是深入解析作物复杂性状与环境互作关系、加速作物遗传改良和智慧农业发展的重要工具^[26]。近年来,随着平台表型采集设备的快速迭代与规模化应用,产生了海量多模态非结构化作物表型数据。由于这些非结构化表型数据在维度、格式、规模和采集频率等方面存在显著差异,传统关系型数据的存储方式已无法满足此类数据的高效存取需求。因此,本文基于 Hadoop 生态设计了一种作物表型大数据存取框架,并对其中的核心问题进行了优化。

作物表型数据通常呈现海量、小体积文件居多的特点,且在育种分析中需要针对单一模态的表型数据进行高频次访问。原生 HDFS 在处理此类数据时面临读写效率低、NameNode 元数据信息存储内存消耗大等性能瓶颈问题。针对此问题本文提出了一种基于 MCH 的大数据存储框架,将表型数据按模态分类合并存储,并使用双层哈希技术构建原始文件的局部索引,相比于传统 HAR 方案、SequenceFile 方案和 MapFile 方案^[17-18],有效提高了数据块整体空间利用率并降低了 NameNode 内存占用压力。

在高并发读取场景下,频繁的元数据请求会显著增加 NameNode 通信压力,极大地降低了数据访问效率。针对此问题,现有研究大多聚焦于引入单层缓存机制,通过对部分数据的元数据信息进行缓存预取,以减少 NameNode 通信次数。例如秦加伟

等^[22]构建了基于 HashMap 的缓存机制,曾梦等^[17]构建了基于 LRU 算法的缓存策略,这些方法均在一定程度上缓解了 NameNode 通信压力,提高了数据访问效率,但是其仅针对重复文件读取时的效率高,针对表型数据的遍历式读取,缓存命中率极低,并且对于一些偶发性和周期性的批量操作,还会出现缓存污染的问题。基于这些问题,本文设计了一种基于数据热度的双层缓存机制,通过将目标数据的索引信息分层缓存,不仅可以提高重复读取数据的访问效率,还可以提高目标数据所属合并文件里其他数据的访问效率,并在一定程度上降低了单缓存的访问压力,避免了缓存污染。

本文所提出的多模态表型数据存取方案不仅提高了文件的存取效率,而且还减少了存储空间的费用,满足作物表型数据存取需求,但是也存在一定的局限性。在上传时,本文方案需将元数据信息同步写入 HBase,这会增加写入耗时,在写入多检索少的场景下,难以满足数据实时写入的性能要求,未来仍需进一步优化,以满足更多场景下的存取需求。

4 结论

(1) 面对海量作物表型数据的存储与检索需求,基于关系型数据库的传统处理方式已不能保证作物表型数据的存取性能。为此,本文采用大数据分布式技术,利用 Hadoop 和 HBase 平台作为数据的存储底层,优化了海量表型数据的存取性能。并利用 ElasticSearch 设计了基于 HBase 的二级索引,实现了表型数据的多样化检索需求。

(2) 针对原生 HDFS 处理海量表型数据的缺陷问题,提出了一种基于模态分类的降序最佳适应文件合并策略,并使用双层哈希创建索引文件。该策略先将原始文件按模态分类合并后再上传至 HDFS,相比于传统方法,有效降低了 NameNode 元数据信息存储压力,并提高了单模态表型数据的检索效率。

(3) 针对高通量作物表型平台中,热点数据的低延迟访问需求,提出了一种基于数据热度的双层缓存机制,通过将热点数据的元数据信息分层缓存,有效提高了热点数据的读取效率。同时为提高热点数据的 Redis 缓存命中率,提出了一种数据热度评估模型,该模型综合考虑了数据的访问频率和时间特性,一定程度上提高了热点数据的 Redis 缓存留存率。

参 考 文 献

[1] 赵春江. 植物表型组学大数据及其研究进展[J]. 农业大数据学报,2019,1(2):5-14.

- ZHAO Chunjiang. Big data of plant phenomics and its research progress[J]. Journal of Agricultural Big Data, 2019, 1(2): 5 – 14. (in Chinese)
- [2] KIM S L, SOLEHATI N, CHOI I C, et al. Data management for plant phenomics[J]. Journal of Plant Biology, 2017, 60(4): 285 – 297.
- [3] WEN W L, GU S H, ZHANG Y, et al. Standard framework construction of technology and equipment for big data in crop phenomics[J]. Engineering, 2024, 42(11): 175 – 184.
- [4] 张慧春, 周宏平, 郑加强, 等. 植物表型平台与图像分析技术研究进展与展望[J]. 农业机械学报, 2020, 51(3): 1 – 17.
ZHANG Huichun, ZHOU Hongping, ZHENG Jiaqiang, et al. Research progress and prospect in plant phenotyping platform and image analysis technology[J]. Transactions of the Chinese Society for Agricultural Machinery, 2020, 51(3): 1 – 17. (in Chinese)
- [5] 宋鹏, 李正达, 杨蒙, 等. 作物表型机器人研究现状与展望[J]. 农业机械学报, 2025, 56(3): 1 – 17.
SONG Peng, LI Zhengda, YANG Meng, et al. Current status and prospects of crop phenotyping robots[J]. Transactions of the Chinese Society for Agricultural Machinery, 2025, 56(3): 1 – 17. (in Chinese)
- [6] 赵燕东, 刘浩宇, 宋润泽, 等. 多传感器作物表型时间同步采集方法[J]. 农业机械学报, 2025, 56(3): 58 – 66.
ZHAO Yandong, LIU Haoyu, SONG Runze, et al. Multi-sensor crop phenotype time synchronous acquisition method[J]. Transactions of the Chinese Society for Agricultural Machinery, 2025, 56(3): 58 – 66. (in Chinese)
- [7] 刘艳萍, 杜雅丽, 聂铭君, 等. 基于称重式蒸渗仪及多种传感器的作物表型及蒸散监测系统研制[J]. 农业工程学报, 2019, 35(1): 114 – 122.
LIU Yanping, DU Yali, NIE Mingjun, et al. Design of crop phenotype and evapotranspiration monitoring system based on weighing lysimeter and multi-sensors[J]. Transactions of the CSAE, 2019, 35(1): 114 – 122. (in Chinese)
- [8] 许鑫, 时雷, 何龙, 等. 基于 NoSQL 数据库的农田物联网云存储系统设计与实现[J]. 农业工程学报, 2019, 35(1): 172 – 179.
XU Xin, SHI Lei, HE Long, et al. Design and implementation of cloud storage system for farmland internet of things based on NoSQL database[J]. Transactions of the CSAE, 2019, 35(1): 172 – 179. (in Chinese)
- [9] 孟超英, 张雪彬, 陈红茜, 等. 基于 Hadoop 的蛋鸡设施养殖智能监测管理系统研究[J]. 农业机械学报, 2018, 49(9): 166 – 175.
MENG Chaoying, ZHANG Xuebin, CHEN Hongqian, et al. Design of intelligent monitoring and management system based on Hadoop for large-scale layer house[J]. Transactions of the Chinese Society for Agricultural Machinery, 2018, 49(9): 166 – 175. (in Chinese)
- [10] LU Y, HUANG Z X, YU S W, et al. Research status and prospect of clinical medical big data[J]. Chinese Medical Equipment Journal, 2017, 16(3): 112 – 115.
- [11] WANG Y G, ZHAN G Y, SHAO Z G. Research on the integrated application of hospital LIS and HIS[J]. Software Guide, 2016, 15(6): 157 – 159.
- [12] 陈文杰. 基于 Hadoop 的医疗影像文件的存储与检索[D]. 北京: 北京工业大学, 2019.
CHEN Wenjie. Storage and retrieval of medical image files based on Hadoop[D]. Beijing: Beijing University of Technology, 2019. (in Chinese)
- [13] 段涛. 基于近感图像的作物表型高通量测量方法研究[D]. 北京: 中国农业大学, 2017.
DUAN Tao. The research of high-throughput methodology for crop phenotyping using proximal sensing images[D]. Beijing: China Agricultural University, 2017. (in Chinese)
- [14] 王新阳, 贾相宇, 陈志泊, 等. 森林生态站大数据快速存储与索引方法[J]. 农业机械学报, 2021, 52(8): 195 – 204, 212.
WANG Xinyang, JIA Xiangyu, CHEN Zhibo, et al. Fast storage and indexing method of big data in forest ecological station[J]. Transactions of the Chinese Society for Agricultural Machinery, 2021, 52(8): 195 – 204, 212. (in Chinese)
- [15] 倪锦涛. 基于 HDFS 的小文件优化存储管理系统的研究与实现[D]. 北京: 北京邮电大学, 2020.
NI Jintao. Research and implementation of small file optimization storage management system based on HDFS[D]. Beijing: Beijing University of Posts and Telecommunications, 2020. (in Chinese)
- [16] 李国, 李汶晓, 徐俊洁. 航空货运中海量小文件的存储优化[J]. 计算机工程与设计, 2018, 39(5): 1484 – 1489.
LI Guo, LI Wenxiao, XU Junjie. Storage optimization of massive small file in air cargo[J]. Computer Engineering and Design, 2018, 39(5): 1484 – 1489. (in Chinese)
- [17] 曾梦, 邹北骥, 张文生, 等. 多模态医疗数据中海量小文件存储优化方法[J]. 软件学报, 2023, 34(3): 1451 – 1469.
ZENG Meng, ZOU Beiji, ZHANG Wensheng, et al. Optimization method for storing massive small files in multi-modal medical data[J]. Journal of Software, 2023, 34(3): 1451 – 1469. (in Chinese)
- [18] 程晗, 汪学明. 基于 Redis 的海量智慧医疗小文件存储架构设计[J]. 计算机应用与软件, 2018, 35(4): 86 – 90, 128.
CHENG Han, WANG Xueming. Massive wisdom medical small files storage architecture design based on Redis[J]. Computer Applications and Software, 2018, 35(4): 86 – 90, 128. (in Chinese)
- [19] 郑通, 郭卫斌, 范贵生. HDFS 中海量小文件合并与预取优化方法的研究[J]. 计算机科学, 2017, 44(增刊 2): 516 – 519, 541.

ZHENG Tong, GUO Weibin, FAN Guisheng. Research on optimization method of merging and prefetching for massive small files in HDFS[J]. Computer Science, 2017, 44(Supp.2): 516–519, 541. (in Chinese)

[20] 陈法河, 柴小丽. 基于 Ceph 存储系统的小文件存储优化方案[J]. 计算机系统应用, 2022, 31(2): 108–113.
CHEN Fahe, CHAI Xiaoli. Optimization of small file storage algorithm based on Ceph storage system[J]. Computer Systems Applications, 2022, 31(2): 108–113. (in Chinese)

[21] 梁懿, 刘迪, 陈又咏, 等. 国产化环境下的海量小文件数据分布式存储技术[J]. 计算技术与自动化, 2023, 42(3): 141–146.
LINAG Yi, LIU Di, CHEN Youyong, et al. Distributed storage technology of massive small file data in localization environment[J]. Computing Technology and Automation, 2023, 42(3): 141–146. (in Chinese)

[22] 秦加伟, 刘辉, 方木云. 大数据平台下基于类型的小文件合并方法[J]. 软件工程, 2020, 23(10): 12–14, 11.
QIN Jiawei, LIU Hui, FANG Muyun. Type-based small file merging method on big data platform[J]. Software Engineering, 2020, 23(10): 12–14, 11. (in Chinese)

[23] 谢磊, 吴建华, 廖卓凡, 等. 一种基于 LRFU 缓存替换策略的 HDFS 客户端本地缓存设计与实现[J]. 计算机应用与软件, 2018, 35(5): 15–20, 94.
XIE Lei, WU Jianhua, LIAO Zhuofan, et al. Design and implementation of an HDFS client local cache based on LRFU cache replacement policy[J]. Computer Applications and Software, 2018, 35(5): 15–20, 94. (in Chinese)

[24] ZHAI Y L, KONDI J T, LIN K J, et al. Hadoop perfect file: a fast and memory-efficient metadata access archive file to face small files problem in HDFS[J]. Journal of Parallel and Distributed Computing, 2021, 156: 119–130.

[25] 许昌涛, 黄连生. 基于 HBase 的极向场电源系统异构数据存储设计[J]. 仪表技术, 2024(2): 1–5, 38.
XU Changtao, HUANG Liansheng. Design of heterogeneous data storage for poloidal field power supply system based on HBase[J]. Instrumentation Technology, 2024(2): 1–5, 38. (in Chinese)

[26] 陈龙跃, 段丹丹, 徐卫兵, 等. 地、空尺度的农作物高通量表型分析的应用与挑战[J]. 广东农业科学, 2024, 51(11): 149–162.
CHEN Longyue, DUAN Dandan, XU Weibing, et al. Applications and challenges of high-throughput phenotyping of crops at ground and aviation scales[J]. Guangdong Agricultural Sciences, 2024, 51(11): 149–162. (in Chinese)

(上接第 18 页)

[46] WU Sheng, ZHANG Ying, ZHAO Yanxin, et al. Using high-throughput phenotyping platform mvs-pheno to decipher the genetic architecture of plant spatial geometric 3D phenotypes for maize[J]. Computers and Electronics in Agriculture, 2024, 225: 109259.

[47] WANG Jinglu, WANG Chuanyu, LU Xianju, et al. Dissecting the genetic structure of maize leaf sheaths at seedling stage by image-based high-throughput phenotypic acquisition and characterization[J]. Frontiers in Plant Science, 2022, 13: 826875.

[48] WU Sheng, WANG Jinglu, ZHAO Yanxin, et al. Characterization and genetic dissection of maize ear leaf midrib acquired by 3D digital technology[J]. Frontiers in Plant Science, 2022, 13: 1063056.

[49] LIAO Shengjin, ZHANG Ying, WANG Jinglu, et al. Exploring the developmental progression of endosperm cavity formation in maize grain and the underlying molecular basis using X-ray tomography and genome wide association study[J]. Frontiers in Plant Science, 2022, 13: 847884.

[50] GUO Shangjing, ZHOU Guoliang, WANG Jinglu, et al. High-throughput phenotyping accelerates the dissection of the phenotypic variation and genetic architecture of shank vascular bundles in maize (*Zea mays* L.) [J]. Plants, 2022, 11(10): 1339.

[51] WANG Xiaqing, SHI Zi, ZHANG Ruyang, et al. Stalk architecture, cell wall composition, and qtl underlying high stalk flexibility for improved lodging resistance in maize[J]. BMC Plant Biology, 2020, 20(1): 515.

[52] GUO Zhonglong, LI Bo, DU Jianjun, et al. LettuceGDB: the community database for lettuce genetics and omics[J]. Plant Communications, 2023, 4(1): 100425.

[53] WANG Jinglu, ZHAO Shuaihao, ZHANG Ying, et al. Investigating the genetic basis of maize ear characteristics: a comprehensive genome-wide study utilizing high-throughput phenotypic measurement method and system[J]. Frontiers in Plant Science, 2023, 14: 1248446.