文章编号:1671-4598(2023)04-0252-06 DOI:10.16526/j.cnki.11—4762/tp.2023.04.039 中图分类号:TP301.6 文献标识码:A

基于两阶分区的 MapReduce 实验室系统负载均衡研究

郑文丽[1],熊贝贝[1,2],程立勋[1],蔡伊娜[1,2],包先雨[1]
(1. 深圳市检验检疫科学研究院,广东 深圳 518045;
2. 深圳海关食品检验检疫技术中心,广东 深圳 518000)

摘要:在实验室系统处理海量原始数据时,实际应用场景中存在采样率高、偏度(skewness)高的特殊情况,导致在使用两阶分区算法在平衡同构环境下的 Reducer 节点负载时,无法有效地处理这些问题,为此,引入 MapReduce 的并行化处理,可以提高实验室系统中采样数据利用率;同时,为了解决数据偏度和采样度高的问题,则采用了 ICSC 分区调度的算法;经过实验证明,基于两阶分区的 MapReduce 负载均衡算法能够有效减少 Mapper 和 Reducer 节点空转的时间;随着数据偏度的增加,算法的执行时长基本不产生变化,即数据偏度对该算法执行时间的影响较小,此外,数据采样度的增加,ICSC 分区调度算法也保持着对比模型中最少的时间开销;因此,基于两阶分区的 MapReduce 负载均衡算法弱化了 Reducer 节点间的依赖性,并提升 MapReduce 任务的执行效率和容错率,从而高效地实现 MapReduce 框架下的实验室系统中数据处理的负载均衡。

关键词:两阶分区;MapReduce;分区调度;负载均衡;实验室系统

Research on Load Balancing of MapReduce Laboratory System Based on Two-tier Partition

ZHENG Wenli[1], XIONG Beibei[1,2], CHENG Lixun [1], CAI Yina[1,2], BAO Xianyu[1]
(1. Shenzhen Academy of Inspection and Quarantine, Shenzhen 518045, China;
2. Shenzhen Customs Food Inspection and Quarantine Center, Shenzhen 518000, China)

Abstract: When processing raw data in a laboratory system, there are the special cases of high sampling rate and high skewness in real application scenarios, a two-order partitioning algorithm is used to effectively deal with these problems by balancing the load on the Reducer nodes in a homogeneous environment. Therefore, the parallel processing of MapReduce is introduced to improve the utilization of sampling data in the laboratory system; At the same time, an improved cluster split combination (ICSC) partition scheduling algorithm is adopted to solve the data skewness and high sampling. The experimental results show that the MapReduce load balancing algorithm based on two-tier partition can effectively reduce the idle time of the Mapper and Reducer nodes. With the increase of data skewness, the execution time of the algorithm is basically unchanged, that is, data skewness has little impact on the execution time of the algorithm. In addition, with the increase of data sampling, the ICSC partition scheduling algorithm also maintains the minimum time cost in the comparison model. Therefore, the MapReduce load balancing algorithm based on the two-tier partitions weakens the dependency between the reducer nodes, improves the execution efficiency and fault tolerance of the MapReduce tasks, and effectively realizes the load balancing of data processing in the laboratory system under the MapReduce framework.

Keywords: two-tier partition; MapReduce; partition scheduling; load balancing; laboratory system

0 引 言

在数字化信息时代背景下,通过实验室系统存储和处理海量数据,已成为日常检验检测工作不可或缺的一部分[1]。而随着实验室系统智慧化的要求,改进存储和处理技术在大规模数据高效采集领域中越发重要。其中,MapReduce 是一种处理大规模数据集的分布式框架,拥有拓展性强和容错率高的特点。通过 Mapper 和 Reducer 对海量的检测数据进行拆分和归类,与传统构架相比,能更高效地处理数据。而 Hadoop 框架源于 Apache Nutch,集合了 HDFS(hadoop distributed file system),YARN(yet another resource negotiator)和 MapReduce,是具备高容错性、高拓展性和低成本特点的分布式软件框架,广泛应用于各种领域,包括分布式搜索[2-3]、模式识别[4-5]、计算机

收稿日期:2022-11-11; **修回日期**:2022-12-19。
基金项目:国家重点研发计划课题(2019YFC1605401);海关总署课题(2020HK109)。
作者简介:郑文丽(1975-),女,广东潮阳人,大学本科,高级工程师,主要从事非传统安全与应用信息化方向的研究。
通讯作者:包先雨(1981-),男,安徽桐城人,博士,研究员,主要从事非传统安全与应用信息化方向的研究。
引用格式:郑文丽,熊贝贝,程立勋,等.基于两阶分区的 MapReduce 实验室系统负载均衡研究[J].计算机测量与控制,2023,31(4):252-257.

机器学习[6-7]和文献聚类[8-9]等。

MapReduce 框架主要运用于大型云平台的海量数据库存储的应用场景中，可解决数据采集高频、数据类型多样、数据来源复杂以及用户需求响应及时等难点痛点。同时，根据不同场景的业务应用流程，云平台的设计与开发通常采用分层式管理并按需优化，这为大数据的分流工作提出了新的挑战。王立俊等人[10]在设计气象大数据云平台时，其总体架构共分为基础设施层、数据管理与处理层和应用层，实现了更快的数据采集以及更优的数据处理功能。鉴于气象数据属于典型的时序性数据，其数据流量高达 6 万次/秒，利用 MapReduce 这种高性能的分布式计算框架则可以大幅简化数据处理流程。在该云平台中，其 MapReduce 任务包括了 3 个任务阶段，分别为 Map 阶段、Shuffle 阶段以及 Reduce 阶段，这 3 个阶段可连接为链式工作流。数据流在 Map 阶段，依据数据的不同属性和特点进行处理之后，形成不同长度的数据块，以相应的规则映射成键值对，这些键值对就是实例化的 Map 任务。经过 Map 阶段处理后的分片数据，生成＜key，value＞键值对，输送至 Shuffle 阶段。在 Shuffle 阶段，主要是对 Map 阶段处理后的分片数据进行规则化采集，即采用键值对列表的序号与键值对形成映射，该阶段最终产生分片数据排序表。在 Reduce 阶段接收该排序表后，便按该表的序号以及 Reduce 算法对分片数据进行分类处理。通过各阶段数据处理方式，不仅对批量数据实行了有针对性的调度管理，同时也对用户隐藏了具体的管理流程，提升了处理效率，也降低了用户的使用成本。

MapReduce 除了在云平台系统运用方面有很多理论研究基础，在其框架的改良优化方面也有着广泛的研究成果，胡东明等人[11]在 MapReduce 框架下提出了一种负载均衡的 Top-k 连接查询算法。该算法不仅在 MapReduce 框架下实现了 Top-k 连接查询算法，还通过提前终止机制和负载均衡机制来增强其数据连接处理性能。由于 Map 任务和 Reduce 任务可以并行处理的特点，能尽量避免了链接 MapReduce 作业的初始化开销，降低了数据处理成本。该算法流程包括直方图构建、提前终止机制、数据过滤、负载均衡机制 4 个步骤组成，其中提前终止、数据过滤都在 Map 阶段实现，Reduce 阶段则通过 Top-k 连接查询算法完成数据清洗。在数据过滤阶段，Map 任务会处理每个记录，因为针对每个作业会形成不同的过滤机制。通过了数据过滤机制的记录则进入到 Reduce 阶段，并使用启发式任务调度算法对每个记录的 Reduce 任务进行数据调度，将这些任务依次分配给连接总数较低的 Reducer 进行处理。该算法利用了 Map 阶段输出的键值对以及连接值进行分组，将其按照自定义分区程序的结果分配给 Reduce 任务。MapReduce 的并行化处理大大降低了算法的总执行时间，但采取不同 Top-k 算法则产生不同的任务执行时间，经比较，最终选用了 P-TKJ 算法，相较于 RSJ 算法获得了更快的执行速度，并随着数据集的增大，其效率的优势越明显，使得 MapReduce 框架下不仅实现了海量数据的 Top-k 连接查询算法，还提高了 CPU 的利用效率。

在实验室系统中引入 MapReduce 框架[12]，可以极大简化分布式程序，集中精力于数据处理的任务本身，提高实验室系统数据处理的效率[13]。实验室系统在执行 MapReduce 任务时，数据是以键值对输入，并在 Mapper 节点时进行聚合处理，最终根据哈希值函数 Shuffle 至各 Reducer 节点进行 Sort 和 Reduce。在风力发电行业中，其实验室监测数据包含设备的状态参数、气象环境、地理信息等，数据体量较大，类型多。虽然 Hadoop 平台能够满足大数据的基本需求，但在运营效率方面仍有待提高。王林童等人[14]提出了基于 MapReduce 的多源数据并行关联查询的优化方法，主要是针对风电大数据进行存储预处理，尽量使具有关联关系的数据存储在同一存储节点中，之后采用哈希分桶算法对数据存储进行优化，并在查询时利用 MapReduce 框架的并行性特点，最终对数据可以采用并行优化的查询与计算。具体而言，该系统首先对具有相同属性（时间、地点、设备号等）的风电数据进行归并的预处理，在关联查询的基础上数据清洗之后得到风电时序数据合并表。合并表按照关联字段的哈希值分配到不同的“桶”中，同一个“桶”中的数据即存储到同一个数据节点中，实现本地化的存储优化。当用户使用该系统进行数据查询时，设计在 Map 阶段完成风电数据的查询、过滤、筛选等操作，尽量减少 Reduce 阶段的操作，此时就大大降低了传输时延，提升了数据查询效率。文献［15］则利用 MapReduce 框架，构建了一个基于物联网的细胞生物学智慧云实验室系统，该模型共包括四层：实验设备层、IT 基础设施层、控制层以及数据分析层。实验设备层主要工作为自动采集不同实验仪器设备上产生的数据，采集后的实验数据交由 IT 基础设施层进行存储，在控制层可对这些存储的数据进行查询，而在数据分析层，则针对数据种类应用不同的分析方式进行筛选；通过 4 个层级的相互作用，为实时根据用户的需求进行针对性调整和优化提供了极大的便利。

在这个过程中，如果原始数据分布不均，易出现数据倾斜的问题，引发 Reducer 节点负载不均衡，导致整个 MapReduce 任务的执行时间过长。因此，基于 MapReduce 的实验室系统负载均衡成为近期国内外学者们的研究热点。

1　基于两阶分区的 MapReduce 负载均衡算法

针对原始数据分布不均导致的数据倾斜问题，国内外学者从不同角度提出了解决方案。杜鹃等人提出了一种利用快速无偏分层图抽样算法的负载均衡算法，在小规模数据上运行良好，但未在大型真实的复杂数据集中运用、验证[16]。陶永才等人提出了 MR-LSP（MapReduceon-line load balancing mechanismbasedon sample partition）算法，对原始数据进行采样分析，通过分析结果进行负载均衡的分区分配 Reducer 节点的策略，但该算法忽略了数据采样率不高时的情况[17]。在马青山等人[18]提出的 DSJA（data skew join algorithm）算法从数据关系表中的连接键出现频率的

角度出发，区分数据是否产生连接倾斜的情况，再分配到相应的 Reducer 节点中进行处理，但该算法仅考虑了 Reduce 端输出的负载均衡，未考虑 Map 端到 Reduce 端的输入阶段的负载均衡处理。M. A. Irandoost 等人提出的 LAHP（learning automata hash partitioner）算法[19]是根据学习自动机策略，在作业执行阶段，对数据键值进行调配各个 Reducer 节点的数据量；但该算法只考虑了数据偏度高，而未考虑数据采样率高的情形，且只优化了执行阶段，忽略了计算时间对流程的影响。Elaheh Gavagsaz 等[20]提出的可拓展的高偏度数据的随机采样模型 SBaSC（sorted-balance algorithm using scalable simple random sampling）对高偏度数据进行了随机采样，通过采样中键值近似分布推算数据的分布，从而进行负载平衡的分配；该算法在基于 Spark 的网络数据中得到了较好的效果，但未能泛化至不同的数据集，存在一定的局限性。黄伟建[21]等使用并行随机抽样贪心算法，缩短了采样阶段的执行时间，但当 MapReduce 输入数据量较大时，准确性不够且负载效果较差。

为解决上述问题，Xu 等[22]提出的两阶分区算法在数据偏度较低时通过 CC（cluster combination）调度方案实现负载平衡，在数据偏度较高时则通过 CSC（cluster split combination）调度方案，可比传统的 MapReduce 减少 60％的运算时长。在该算法中，一个数据处理作业被分为了两个阶段，一是制定分区方案，二是执行 MapReduce 作业。在制定分区方案中，CC 调度方案主要思路是选择具有最大数量的键值对的数据块调度给当前最小工作量的键值对的 Reducer，这种启发式算法使用了标准差这一评价指标来自适应地衡量所有 Reducer 的负载量。当标准差高于设定的阈值后，将对数据块进行再分配。其分配的方式基于对数据块所含键值对数量的降序排序，并对 Reducer 也进行工作量进行降序排列，接下来则将两个表中的首位进行匹配，以确保整个 MapReduce 系统的负载均衡。CC 调度方案可以解决数据量轻度倾斜的情况，但面对数据量极度倾斜的情况，就无法实现较好的效果。针对这样的情况，该算法又提出了第二种情况的方法，即 CSC 调度方案。面对包含着较多数据记录的数据块，CSC 采用的是划分的方式，将较大的数据块划分成较小的数据块。由于在 MapReduce 框架下，每个数据块都对应着一个 Reducer，因此在对较大数据进行划分的时候，需要额外分配一个 Reducer。与 CC 类似，CSC 在进行数据块分割的时候也是采用启发式算法解决这个 NP-Hard（non-deterministic polynomial hard）问题，偏度较大的数据块将被分成 n 块（n 为 Reducer 的总数），以确保整个 MapReduce 系统保持负载平衡。在分区任务完成之后，该算法便执行 MapReduce 任务。在 MapReduce 任务阶段主要是对数据集进行采样，不同的数据则有不同的特点，因此，不同的数据集则采用不同的采样方法，如对预训练过的数据集，则采用区间分布；对于全新的数据集，则采用随机抽样的方式。该算法为了具备更强的泛化性能，采取了随机抽样的方式，并利用 Map 任务和 Reduce 任务的并行处理特点，先对数据集进行预处理，利用键值中位数估值的概率进行分布。首先对数据集进行均匀的数据采样，再针对采样中出现键值的概率分布进行统计，基于统计结果，拟合不同的概率分布，并根据相应概率分布对数据块进行抽样，以解决数据偏度和采样率都较高的问题。虽然该两阶分区算法在解决数据偏度和采样率都较高的问题中都有相当好的性能表现，但在执行 MapReduce 作业，该算法也存在着以下的缺陷：

1）由于两次 MapReduce 任务采用了串行的执行方式，导致整体任务的执行时间长。

2）在采样率较高时，由于抽样数据会在执行阶段进行重复的 MapReduce 作业，处理时间过长。

3）执行 CSC 调度方案时，资源利用率较低，导致出现了部分 Reducer 节点出现空转的情况。

为了解决上述算法执行时间较长及负载不均衡的问题，本文提出了一种基于 MapReduce 框架下两阶分区的改进算法。该算法主要工作为：

1）对 MapReduce 任务的流程进行并行化处理。

2）在采样率高的情况下，将采样阶段输出数据进行回收。

3）在原始数据偏度较高的情况下，提出 ICSC（improved cluster split combination）算法。本文提出的算法可以在算法速率提高且执行时间缩短的同时，有针对性地解决高偏度，高采样率情况下数据倾斜的问题。改进后两阶段的流程如图 1 和图 2 所示。

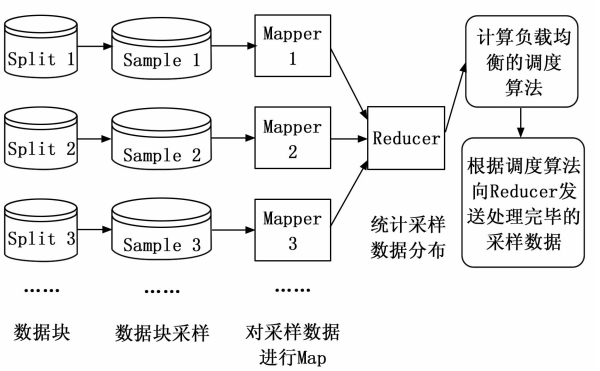

图 1 采样阶段流程

在采样阶段，与优化前的两阶分区算法不同的是，之前是先进行采样数据进行数据统计再划分数据块，优化后则是先对数据块进行划分，并对划分后的数据块进行采样后与数据集的概率分布进行比对，当某一个数据块的采样率很高（超过设定的阈值）时，就不需要再重新对数据块进行采样而直接使用采样出的数据。低于采样率阈值的数据块则需要再进一步采样，但也是增加而非重新分配。这样的做法不仅能够降低时间开销，同时也降低了系统对于数据取用的 I/O 成本。

而在 MapReduce 执行阶段，本文算法除了使用其并行处理的特点，在数据被划分为不同数据块后被分配给不同

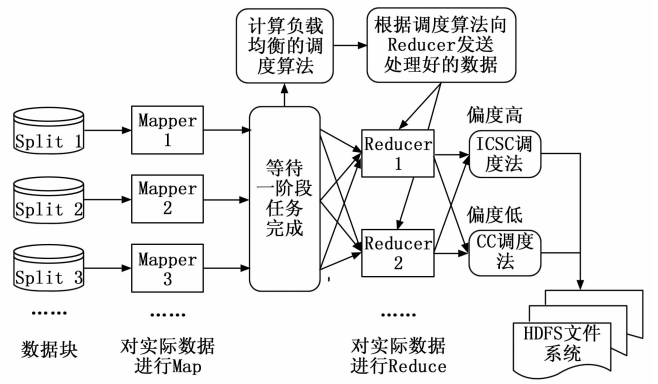

图 2　执行阶段流程

的 Mapper，由于数据块在采样阶段就出现了处理速度不一致的情况，因此如果采用串行方式则会大大拖累不需要重采样的数据块，增加时间开销。本文算法通过直接处理和计算系统负载均衡的方式并行处理数据块，并在分配好数据块的 Reducer 之后再进行不同数据偏度方式进行调度。对于偏度较低的数据块，本文依然沿用了两阶分区算法中的 CC 调度方式，而针对偏度较高的数据块则采用了优化了 CSC 调度方式的 ICSC 方法，具体的过程在第 2 节中进行详细地阐述。

2　算法设计

2.1　两阶并行的设计

根据 Apache 官方文件所述，当系统批处理数据且执行 MapReduce 任务时，Mapper 和 Reducer 的任务可以进行一定程度的并行处理。其中，Mapper 需完成 Map 的子任务，并将输入的键值对进行归类。而 Reducer 需要完成的任务主要分为 3 个子阶段：Shuffle、Sort 和 Reduce。Shuffle 阶段的职责为根据提供的哈希函数将 Map 任务的结果分别传输到各个 Reducer 节点上，同时也是唯一一个可以和 Map 任务同时进行的子任务。因此，Shuffle 子任务的结束时间至少晚于 Map 任务的结束时间。Sort 阶段是各个 Reducer 将收到的数据进行排序、统合的过程。最后的子阶段 Reduce 则是对数据进行计算，并将结果输出到 HDFS 的过程。在 Hadoop 的 MapReduce 框架下，可根据 Map 的任务进度，通过调整 Slowstart 参数，对 Shuffle 的开始时间进行自定义。这种若参数设置为 1.00，则为串联运行，Shuffle 子任务将在 Map 任务全部完成后再执行。

在两阶分区的算法中，由于执行阶段 Shuffle 子任务并不依赖于 Map 任务的完成，而是依赖于上一个阶段——采样阶段的 Reducer 结果所产生的分区方案，因此采样阶段和执行阶段并非两个相互独立的任务，所以没有利用调整 Slowstart 参数来进行并行化。但在 MapReduce 任务中，Map 任务的执行时间往往远长于 Shuffle 任务的时间，所以若执行阶段的 Map 和 Shuffle 并行处理，会导致 Shuffle 任务的执行时间被拉长。基于此，本文提出在第一阶段得到采样结果后便开始第二阶段的 Shuffle 任务，通过并行化流程来减少 Mapper 节点的空转导致的资源浪费。具体流程如图 3 所示，实线箭头标识了串行工作时段，虚线箭头标示了并行工作时段。本文算法将原有的流程做两方面更改：首先，让执行阶段 MapReduce 的 Map 任务随着采样阶段 Map 任务的结束立刻开始；其次，让执行阶段 MapReduce 的 Shuffle 任务随着采样阶段的分区方案的计算完成而立即启动，通过并行化的方式缩短整体任务的执行时间。

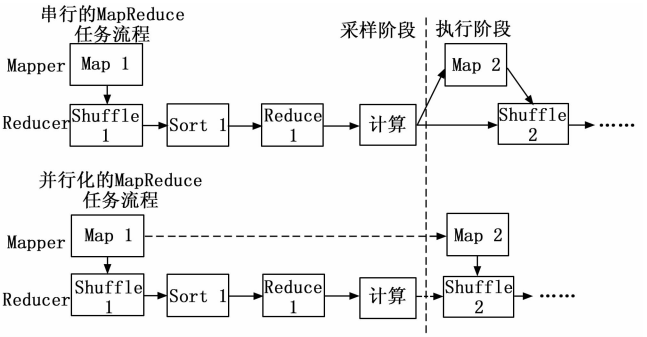

图 3　MapReduce 流程优化前后对比

2.2　采样阶段的设计

在两阶分区算法中，第一次 MapReduce 任务的结果只用来规划执行阶段各 Reducer 节点的负载分配，在数据计算完毕后分区处理后则被舍弃。该算法的设计在数据采样率较高的时候，由于对同样的数据进行相同的多次操作，就会造成计算资源严重浪费，增加数据使用开销。针对这样的问题，本文对上述算法进行了以下改进：将第一阶段数据块采样输出的结果重新 Shuffle 至对应的 Reducer 节点上。因此，到了执行阶段的 MapReduce，则可直接使用采样数据块，而不必对同一数据块进行重复作业，使采样数据得到充分利用，且节省了存储资源与计算资源。

2.3　分区调度阶段的设计

根据数据偏度情况，本文采用了两种不同的分区调度算法。在偏度较低时，则继续沿用 Xu 等人[22]提出的 CC（cluster combination）调度算法，对采样结果进行分区。在偏度较高时，在执行 CC（cluster combination）调度算法，大数据块会导致负载不均衡。因此，上述文献中提出一种 CSC（cluster split combination）调度算法来执行分区。该调度算法先对大数据块进行拆分，由多个 Reducer 节点分别处理，当所有 Reducer 完成各自的任务后，还需要一个 Reducer 进行额外的合并任务。此时，其它 Reducer 会陷入空转的状态；同时，如果所有的 Reducer 都参与到最后一次 Reduce 任务中，任何一个节点的故障都会导致最后一次 Reduce 任务无法进行，致使 MapReduce 任务执行失败，如图 4 所示。

为此，本文提出的 ICSC 算法，如图 5 所示。其主要思想是将 Reducer 节点的工作量进行一定程度的压缩，使得这些 Reducer 完成工作的总时间与其他不参与处理大数据分片的 Reducer 节点保持一致，减少节点的空转时间，更好地利用计算资源，也达到了数据并行处理使得系统优化的

效果。同时，由于执行时间对齐，该算法产生的结果并不会因为任何一个节点的故障而执行失败，意外产生的情况能够独立执行作业进行处理，提升了系统的稳定性。为实现以上功能，ICSC 算法的伪代码如下：

```
ICSC(Clusters C, Reducers R){
average = C.size/R.count;
C.sort();
largeC = new Array();
for (i = 1; i <= C.size; i++){
if (C[i - 1].size >= average){
largeC.add(C[i - 1]);
C.remove(i - 1);
}
}
// n 为参与到包含 split 的 reduce 任务的节点数量。
n = ceiling(largeC.size/average)
for (i = 1; i <= n; i++){
R[i - 1].assign(largeC.split(n));
}
R.sort();
while(! C.empty){
C.assign(R);
R.sort();
}
}
```

算法 1 ICSC 算法伪代码

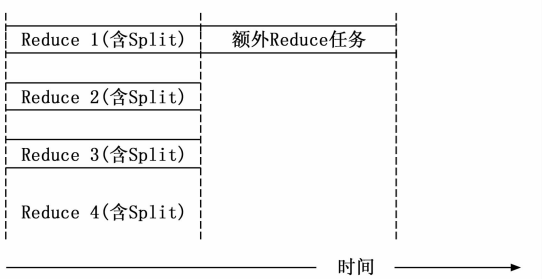

图 4 CSC 调度

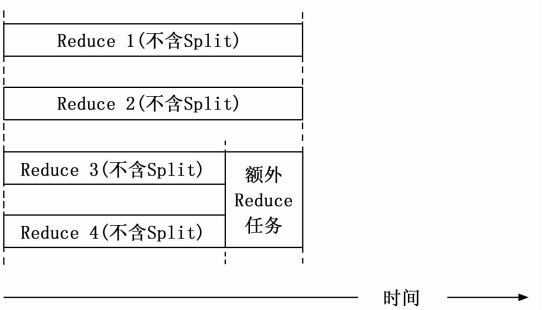

图 5 ICSC 调度

3 试验与分析

3.1 试验验证

为模拟实验室系统的数据收集和处理过程，本试验采用 WordCount 算法和人工生成的数据，在同构环境下以固定大小的输入数据进行实验验证本文提出的算法。由于实验室系统的实际应用环境下，数据属性中的偏度和采样度是影响数据处理阶段的主要因素，因此本文通过控制变量法，验证在不同数据偏度和采样度的情况下，ICSC 算法可在实验室系统的数据处理阶段有效减少耗时，达到流程优化的效果。

具体而言，试验以 JAVA JDK 11.0.13 编制程序模拟 12 节点的集群，在同构环境下使用 WordCount 算法处理 128 MB 人工数据。在实验室系统的环境中，节点间的传输采用 100 Mbps 带宽。测量时间使用 currentTimeMillis 函数来测量自采样阶段 Mapper 到 Reducer 完成最后一个任务为止的时间差值。通过试验得出结果如表 1 和表 2 所示。

表 1 数据偏度在不同方式下对耗时的影响

数据偏度	0.1	0.5	1.0	1.5
CC 所耗时间/s	65	163	210	258
CSC 所耗时间/s	68	79	86	88
ICSC 所耗时间/s	68	72	75	78

表 2 采样度在不同方式下对耗时的影响

数据采样度	0.025	0.05	0.1	0.15	0.2
CC 所耗时间/s	215	209	199	209	217
CSC 所耗时间/s	196	189	185	203	219
ICSC 所耗时间/s	209	188	180	195	210

3.2 比较分析

为了更好地比较 ICSC 算法与 CC 和 CSC 调度法的耗时效果。根据在上述结果（表 1 和表 2）的基础上分别以输入数据偏度和采样率为自变量，以各自的执行时间为因变量，得出不同算法条件下，各自的偏度和采样度的效率，如图 6 和图 7 所示。

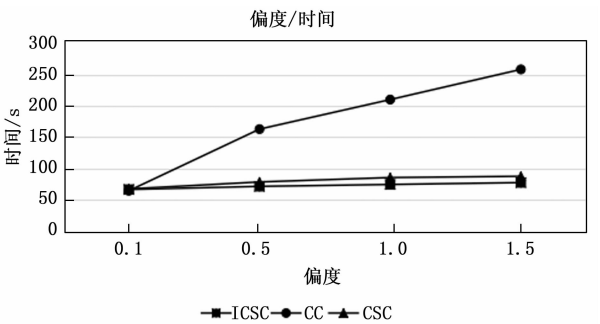

图 6 偏度与时间对比

如图 6 所见，当数据偏度大于 0.1 时，ICSC 算法采样时间更少，且偏度愈大优势更明显，通过测算，在处理偏度为 1.5 的数据时，对比 CSC 算法，ICSC 算法可节约 10% 的执行时间；但当数据偏度小于 0.1 时，由于 ICSC 在最后需要进行一次额外的 Reduce 以合并被拆分的大数据块，因此执行时间是会稍大于 CC 和 CSC 调度法所耗时间。

同样在图 7 中，可以看到在不同采样度的环境下，IC-

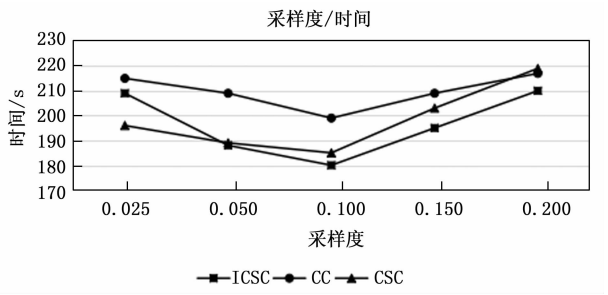

图 7　采样度与时间对比

SC 与两阶分区中算法的执行时间对比。3 种算法均在采样度为 0.1 左右时执行效率最高；而在 0.1 及以上时，ICSC 算法比其它两种算法执行时间更短。这是因为在本地性较高的环境下，ICSC 通过节省第二轮 MapReduce 的数据处理量以达到更优的执行时间。

4　结束语

本文针对两阶分区算法中在实验室系统的实际数据处理场景中存在的不足，对其调度算法和执行流程进行改进。实验结果证明，改进后的算法在数据存在高偏度和高采样度的情况下，均可有效地减少了 Mapper 和 Reducer 节点空转的时间，弱化了 Reducer 节点间的依赖性，缩短 MapReduce 的执行时间，优化了实验室系统数据处理阶段的流程，从而高效地实现 MapReduce 框架下的实验室系统中数据处理的负载均衡。但值得注意的是该算法在本地性较差的环境下仍有待改进，即数据采集系统与数据处理系统之间需要一定时间开销的情况，因为在这样条件的环境下，数据处理节点间的传输时间远大于 MapReduce 作业的计算时间，此时节点间的传输时间成为 ICSC 算法执行时间的主要影响因素。而 ICSC 算法主要针对计算时间进行优化，在执行时间的优化效果较为逊色，这也是本研究下一步工作努力的目标和方向。

参考文献：

[1] 黄宗兰，黄　婷，张文中，等. 实验室信息管理系统在食品检验检测机构中的应用分析 [J]. 食品安全质量检测学报，2020，11 (19)：7130－7134.

[2] POTLUR I，AVINAS H. Enhanced-sweep：communication cost efficient top-k best region search [J]. Arabian Journal for Science and Engineering，2022：1－12.

[3] 杨国华，冯　骥，柳　萱，等. 基于改进秃鹰搜索算法的含分布式电源配电网分区故障定位 [J]. 电力系统保护与控制，2022，50 (18)：1－9.

[4] JIMMY M T. Fuzzy high-utility pattern mining in parallel and distributed Hadoop framework [J]. Information Sciences，2021，553：31－48.

[5] 李　华，刘占伟，郭育艳. 并行 PSO 结合粗糙集的大数据属性约简算法 [J]. 计算机工程与设计，2020，41 (8)：2238－2244.

[6] MOSTAFAEIPOUR A. Investigating the performance of Hadoop and Spark platforms on machine learning algorithms [J]. The Journal of Supercomputing，2020，77 (2)：1－28.

[7] 陈文青. 新型基于大数据分析与挖掘的战略决策框架 [J]. 无线电工程，2022，52 (5)：824－832.

[8] SARDAR T，ANSARI Z. An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm [J]. Future Computing and Informatics Journal，2018，3 (2)：200－209.

[9] 刘卫明，崔　瑜，毛伊敏，等. 基于 MapReduce 和 MSSA 的并行 K-means 算法 [J]. 计算机应用研究，2022，39 (11)：3244－3251，3257.

[10] 王立俊，杜建华，刘骥超，等. 基于决策树挖掘算法的气象大数据云平台设计 [J]. 计算机测量与控制，2022，30 (11)：140－146.

[11] 胡东明，刘旭敏，徐维祥. MapReduce 框架下一种负载均衡的 Top-k 连接查询算法 [J]. 计算机测量与控制，2018，26 (8)：238－242.

[12] 李馥娟. 大数据实验室建设与应用研究 [J]. 实验技术与管理，2018，35 (5)：243－246.

[13] 栾亚建，黄翀民，龚高晟，等. Hadoop 平台的性能优化研究 [J]. 计算机工程，2010，36 (14)：262－263，266.

[14] 王林童，赵　腾，张　焰，等. 基于 Hadoop 的风力发电监测大数据存储优化及并行查询方法 [J]. 电测与仪表，2018，55 (11)：1－6.

[15] PARK S，DAVID F. IoT cloud laboratory：Internet of Things architecture for cellular biology [J]. Internet of Things，2022，20：100618.

[16] 杜　鹃，张　卓，曹建春. 利用快速无偏分层图抽样算法的 MapReduce 负载平衡算法 [J]. 计算机应用与软件，2021，38 (11)：288－294，313.

[17] 陶永才，丁雷道，石　磊，等. MapReduce 在线抽样分区负载均衡研究 [J]. 小型微型计算机系统，2017，38 (2)：238－242.

[18] 马清山，钟　勇，王　阳. 数据倾斜情况下基于 MapReduce 的连接算法 [J]. 计算机应用，2018，38 (S2)：192－195.

[19] IRANDOOST M A，RAHMANI A M，SETAYESHI S. A novel algorithm for handling reducer side data skew in MapReduce based on a learning automata game [J]. Information Sciences，2019，501 (C)：662－679.

[20] GAVAGSAZ E，REZAEE A，JAVADI H H S. Load balancing in reducers for skewed data in MapReduce systems by using scalable simple random sampling [J]. The Journal of Supercomputing，2018，74 (7)：3415－3440.

[21] 黄伟建，贾孟玉，黄　亮. 并行随机抽样贪心算法分区的 MapReduce 负载均衡研究 [J]. 现代电子技术，2020，43 (16)：170－173.

[22] YU J. Balancing reducer workload for skewed data using sampling-based partitioning [J]. Computers & Electrical Engineering，2014，40 (2)：675－687.