大数据环境下 Spark 性能优化分析研究与应用

黄志　苏传程　苏晓红

（广西壮族自治区气象信息中心，南宁 530022）

摘要　针对长时间序列、多站点和多气象要素的大数据量查询需求，现有的 CIMISS（China Integrated Meteorological Information Sharing System）存在支撑能力严重不足的问题。本研究使用广西气象站点建站至今的历史地面气象记录月报表数据资料和现有 Hadoop 集群物理资源，重新设计数据 ETL 流程，构建 Parquet 格式数据集并完成 HDFS 转换存储；嵌入 Spark 的 Broadcast 广播变量，优化 Spark 集群执行参数，提高了集群的处理并行度和 SparkSql 的关联查询效率。结果表明，Parquet 格式数据集的最高压缩比超过 95%，一次性大数据量的查询效率比原来提升了 1～5 倍，并支持高并发访问，为各类相关预报预测业务的开展提供了有效的技术支撑。

关键词　Hadoop；Spark；ETL；Parquet；列式存储；Broadcast

中图分类号：P409　**DOI**：10.19517/j.1671-6345.20210251　**文献标识码**：A

引言

CIMISS（China Integrated Meteorological Information Sharing System）作为国省统一数据环境，是目前开展气象数据服务的核心基础数据支撑平台。CIMISS 的存储架构采用 ORACLE 关系型数据库，目前数据量已达到亿数量级，由于关系型数据库存储压缩比不高，使得存储空间日趋紧张，数据库管理维护任务繁琐而艰巨[1-3]。MUSIC（Meteorological Unified Service Interface Community：气象数据统一服务接口）接口对于短时间序列的数据查询与统计时效较快，随着用户对地面气象观测数据的小时、日值需求不断增加，对于超过 10 年以上的长时间序列、多站点和多气象要素的查询处理响应支撑能力明显不足，严重影响了此类数据服务业务的开展。

中国气象局和北京市气象信息中心使用 HBase 数据存储架构和相关大数据技术，将实时计算框架和分布式数据库系统相结合，实现了海量自动站分钟数据快速入库处理，各类要素查询和统计处理均能达到毫秒级响应，满足了大规模自动站分钟数据在业务应用中对存储和查询的性能需求。HBase 是一个高可靠性、高性能、面向列的分布式数据库，适合处理实时随机读写超大规模的数据，但是 HBase 表的设计模式和复杂的散列化使用方法，对于使用频率集中在少数列、数据很少更新、格式固定的海量历史数据，其存储和管理的模式过于复杂，增加了设计和维护的难度，具有很大的改进空间[4-7]。研究表明 Hadoop 大数据相关技术通过自定义格式构建 TXT 大文本数据集，完成 HDFS 转换存储，在大数据环境下对于海量数据的查询处理，Spark 并行处理 TXT 大文本的方式明显优于 CIMISS 关系型数据库处理的方式[8-9]。

本文在前期 Hadoop 相关大数据研究应用的基础上[8]，以历史地面气象记录月报表为数据源，重新设计数据 ETL 流程，重构基于 Parquet 列式的存储特性的 HDFS 数据集，嵌入 Spark 的 Broadcast 广播变量，优化 Spark 并行度执行参数，实现 Hadoop 集群各工作节点的物理资源利用率最大化，并提升 SparkSql 的关联查询效率。

1　业务系统现状分析

目前 CIMISS 系统采用行式存储的关系型数据库架构，以使用频率最高的地面观测小时值、日值数据库表为例，其表字段有 150～300 个左右，存储的

2021 年广西气象科研计划指令性项目（桂气科 2021ZL02）资助

作者简介：黄志，男，1981 年生，硕士，高级工程师，主要从事气象信息化、大数据、云计算研究与应用，Email：616646373@qq.com

收稿日期：2021 年 5 月 24 日；定稿日期：2021 年 9 月 6 日

站点数 2900 多个，累计数据量已达到亿数量级。用户在实际使用中查询最多的是降水、温度、风、相对湿度等几个常用要素列，数据库每次运行需要把一整行中所有列读取后，再从中提取出目标列，这种处理模式直接导致大数据量查询效率低下，很多数据产品的统计处理只能通过二次处理去获取。此外，数据缺失、数据异常的问题较严重，直接影响了月、年值的数据统计，由于关系型数据库对于海量数据修改需要耗费大量的人力和时间，导致此类数据的完整性、准确性问题一直得不到很好的解决[10-11]。

2 Spark 集群执行参数优化分析与设计

如何提高 Spark 的并行处理效率是本文首要研究的内容。以本文的 Hadoop 集群为例，因为本集群需要配置主服务器接收 Http 请求，需要固定的 IP 地址在应用服务过程进行交互、调试以及 app 的相关的输出信息，所以 Spark 集群采用 yarn-client 运行模式；如果采用 Yarn-Cluster 模式，Yarn 会随机选择一个 NodeManager 启动 Driver 端程序，服务器的 IP 地址将是随机的，此外集群设计 3 个 driver 作为 driver 端的负载均衡。图 1 为 yarn-client 模式下 Spark 集群的执行流程[12]。

Spark 作业中各个 stage 的 task 数量代表了 Spark 作业在各个阶段 stage 的并行度，合理设置并行度可以充分利用集群物理资源，使得每个 task 处理的数据量处于最优值，进而提升运行处理速度。

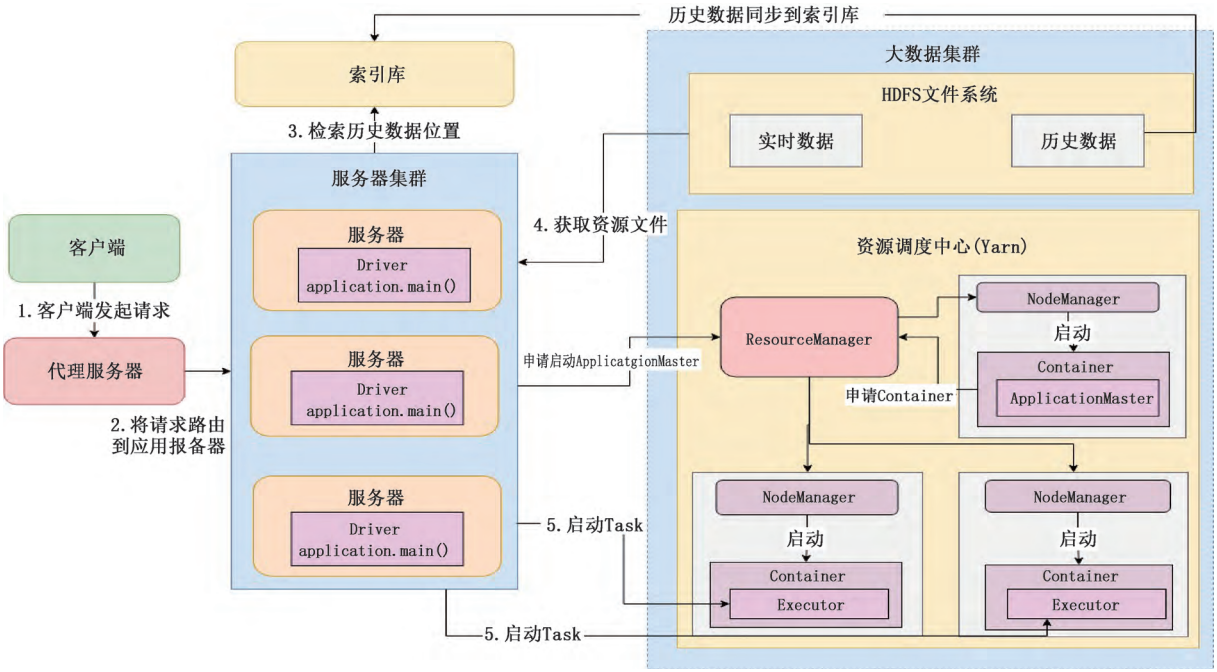

图 1 yarn-client 模式下 Spark 集群的执行流程

基于现有 Hadoop 集群的物理资源，如果 Spark 集群的处理并行度未能与之相匹配，不仅会造成资源的浪费，也使得 Spark 集群整体处理效率未能达到最优。

本文所描述的 Hadoop 的大数据集群包含两种不同型号的 8 台虚拟服务器，包括了 1 台 NameNode 和 7 台 DataNode，表 1 已列出 7 台 DataNode 的 CPU 的 core 总数和内存总量。实际部署中每个节点需要预留资源给非 Yarn 部分的管理服务使用，比如 NameNode 在 Spark 应用中需要做数据归约操作，需要把各个节点上的数据聚合到客户端，需要的 JVM 应用的内存较大，而 Driver 的内存大小与 jvm 的内存大小相关，在 Yarn-client 模式下 Driver 的内存就是非 Yarn 所管理支配的内存。在不影响系统以及其他应用程序的正常运行情况下，系统需要预留物理资源给系统以及其他应用程序，所以本文约定给每个节点预留约 20%～25%的内存、2 个 CPU 给非 yarn 部分的管理服务使用，例如对于 16 核 CPU 和 32 GB 内存的服务器，集群只分配 14 个 CPU 和 24 GB 内存给该节点供 yarn 管理调度使用。

表 1　大数据集群资源配置清单

服务器	CPU/个	内存/G	CPU 总核数	总内存/G	Yarn 的 CPU	Yarn 的内存/G	非 Yarn 的内存/G
(Datanote 1～4)	16	32	64	128	56	96	32
(Datanote 5～7)	16	16	48	48	42	39	9

从表 1 可知,整个集群可利用的 CPU 资源为 56＋42＝98 个,内存为 96 GB＋39 GB＝135 GB,其中 CPU 总个数影响 Spark 的 Job 中的 Task 的并行度,CPU 越多可同时执行的 Task 就越多,计算处理就越快;在内存分配方面,内存分配过小,会引起 Task 计算任务频繁 GC 从而影响执行效率甚至导致宕机,内存分配过大,则会形成冗余浪费。

整个 Spark 集群中设定的 Executor 进程个数是与 CPU 和内存的组合分配方案匹配的。因为 DataNote 1～4 的内存和 CPU 比例为 2∶1 而 DataNote 5～7 的内存和 CPU 比例是 1∶1,整个集群物理机上的资源比例不一致,假设 1 个 Executor 进程以 1 个 CPU 为基准配置,如果 Task 作业需要的内存≤1 GB,那么每台机器上的 CPU 可以完全利用,但会造成内存浪费,DataNode 1～4 尤为严重;如果 Task 作业需要的内存在 1～2 GB 之间,DataNode 1～4 的 CPU 基本完全利用,DataNode 5～7 的 CPU 和内存会出现冗余;如果 Task 需要的内存＞2 GB,那么全部节点的 CPU 和内存都存在冗余,资源浪费情况最严重。所以本集群从 CPU 或内存资源利用率最大化的角度出发,得出的资源最优配置方案如表 2 所示,此方案的 Spark 集群的 executor 配置为:2 核心数＋2900 M＋384 M(堆外内存),方案可以将两种型号服务器即 DataNode 1～4 的 CPU、DataNode 5～7 的内存都分别利用完,使得 executor 个数与整体资源利用率达到最优匹配。

表 2　Spark 计算资源分配方案

服务器	executor 数	CPU/个		内存/G	
		使用	剩余	使用	剩余
Datanote 1	7	14	0	22	2
Datanote 2	7	14	0	22	2
Datanote 3	7	14	0	22	2
Datanote 4	7	14	0	22	2
Datanote 5	4	8	6	13	0
Datanote 6	4	8	6	13	0
Datanote 7	4	8	6	13	0
汇总	40	80	18	127	8

注:每个 executor 的配置:2 核心数＋2900M＋384MB(堆外内存)。

对于 Spark 处理过程中需要采集的文件分区数(partition),本文分为加载数据和处理数据的两个分区进行描述,加载数据由 HDFS 进行分区,每个 block 对应一个分区,不同文件之间不能合并,例如对于日值数据而言,每次全站统计需要加载 2700 多个站的 parquert 文件,分区数为 2700＋,每个文件都是独立加载提取数据并分别获取结果;本文涉及的处理任务为数据基础查询,处理过程没有涉及数据聚合操作,所以不会产生数据倾斜。本文设置 Spark.default.Parallelism 的 task 的参数值为 300(之前使用 Spark 默认设置的参数一般为几十个 task,task 过少会导致资源的浪费,不能最大限度发挥 Spark 集群的处理效率),是 CPU 核心总数的2～3 倍左右,一个 job 的 task 数等于分区数,所以本文的 Spark 的 task 总任务数最大为 2700＋,实际任务数据会根据每个节点实际的 Executor 资源情况进行重新分配,所以总任务数会远小于 2700。根据当前 Spark 集群的配置方案,提交集群初始化命令行如下[12]:

```
MYMSpark_HOME/bin/Spark-submit
－－master yarn
－－executor-memory 2900M
－－num-executors 40
－－executor-cores 2
－－driver-memory 4G
－－conf Spark.default.parallelism＝300　\
－－conf Spark.storage.memoryFraction＝0.6 \
－－conf Spark.shuffle.memoryFraction＝0.2 \
/home/applications/MeteoDataArchives/MeteoDataArchives.jar &
```

3　基于 Parquet 的数据 ETL 流程设计

Apache Parquet 是 Hadoop 生态系统中常用的面向分析型业务的列式存储格式,由 Twitter 和 Cloudera 合作开发,Spark 处理框架默认数据存储格式为 Parquet,所以两者具有很好的兼容性。

根据用户对 CIMISS 系统数据查询的使用场景

进行分析，比如查询多站点长时间序列日值，95%以上的查询需求集中在日值表 200 多个字段中的降水、温度、相对湿度等几个字段，所以采用列式存储是很好的存储方式。为了提升数据的完整性、可用性，本文使用 A 文件（历史地面气象记录月报表）作为数据源，该文件是经过人工审核的数据文件（每个站每月一个 A 文件，包含了当月 70 多个气象要素的日统计值、20 多个小时观测值等数据），具有较高的准确性和完整性，数据稳定后续很少再做修改和删除，规避了 Parquet 格式数据文件在更新、删除操作上的劣势。

以日值为例，本文对 A 文件所辖的 77 个日气象统计要素，增加站号、时间 2 个字段共 79 个字段，以此作为 Parquet 文件的 schema 设计格式（表 3），此格式的 79 个字段虽然比 CIMISS 的日值表所包含的 200 多个字段少了一半，但其所涵盖的主要气象统计要素已能满足日常 95%以上的数据需求（小时值等应用情况类似）。

表 3 日值 schema 表字段格式说明

字段序号	字段名	字段中文解释
1	STATION_ID_C	站号
2	TIME	日期
3	PRS_AVG	日平均气压
4	TEM_AVG	日平均气温
5	PRE_TIME_2008	日夜间降水量
6	PRE_TIME_0820	日白天降水量
7	PRE20	20:00—20:00 降水量
8	PRE08	08:00—08:00 降水量
⋮	⋮	⋮
78	PRS_MAX	日最高气压
79	PRS_MIN	日最低气压

数据 ETL（Extract-Load-Transform）处理是本文数据环境集成的第一步，目的是将 A 文件中的分散、零乱、标准不统一的数据按自定义格式整合到一起，也是构建 Parquet 格式数据集的关键环节。同样以日值数据为例，此过程是归集本省全部站点建站至今所有 A 文件并按站点进行归类，按站点逐个文件对其中包含的 77 个气象要素日值按日抽取后进行数据初级质控，剔除历史错误极值，对特殊字符进行数字化转换等，以保证后续列式数据的同质性，完成 77 个要素的数据抽取清洗之后，将对应站号、日期和 77 个要素日值按日逐行拼接，合并为长时间序列的行格式的大文本 TXT 文件（每个站点对应一个日值合并文件），以充分提高 HDFS 存储块的利用率（HDFS 的存储块默认大小为 128 M），最终转换生成 Parquet 格式的列式数据文件并完成 HDFS 存储转换（存储路径通过文件夹和文件名区分）。上述的数据抽取、清洗、格式转换和存储处理流程就是本文的数据的 ETL 处理流程，因为 Spark 框架已对 Parquet 格式的读写转换进行了封装，一般情况下采用默认的参数设置就能完成 Parquet 格式文件的读写处理。图 2 为日值数据的 ETL 处理流程[13-15]。

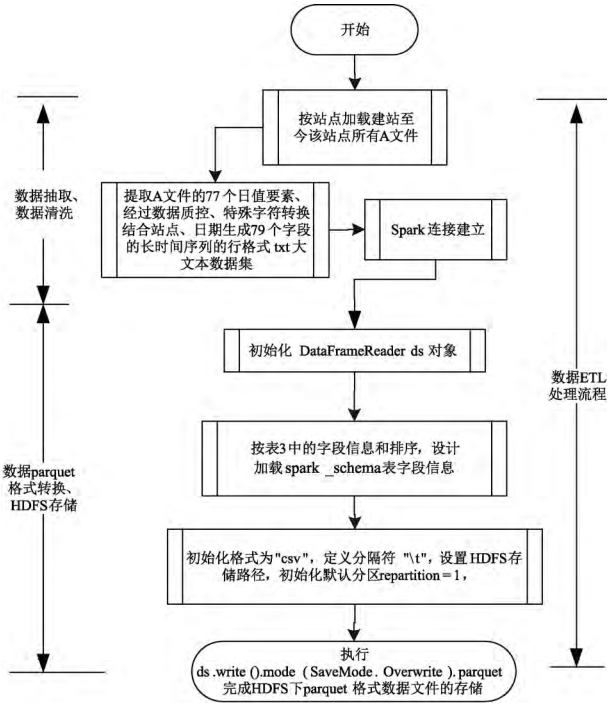

图 2 日值数据的 ETL 处理流程

经过 ETL 处理后生成的 Parquet 文件是由一个 header 和一个或多个 block 块组成，以一个 footer 结尾。header 中只包含一个 4 个字节的魔术字“PAR1”用来识别整个 Parquet 文件格式，文件中所有的 metadata（元数据）都存在于 footer 中，footer 中的 metadata 包含了格式的版本信息、schema 信息、block 中的 metadata 信息，footer 中倒数第 2 个字段是一个以 4 个字节长度的 footer 的 metadata，最后一个字段则是与 header 中包含一样的“PAR1”。

Parquet 文件中的每个 block 以 Row Group 的形式存储,因此文件中的数据被划分为一个或多个 Row Group,这些 Row Group 是由一个或多个 Column Chunks 组成的列数据,每个 Column Chunks 的数据以 Page 的作为最小单元进行组织,每个 Page 只包含特定列的值,因为前文在数据 ETL 过程中的数据清洗环节已将特殊字符经过数字转换,使得每个 page 保持了同质性从而具有很好的压缩特性。图 3 为本文日值 Parquet 文件的 Header、Data Block 和 Footer 格式描述。

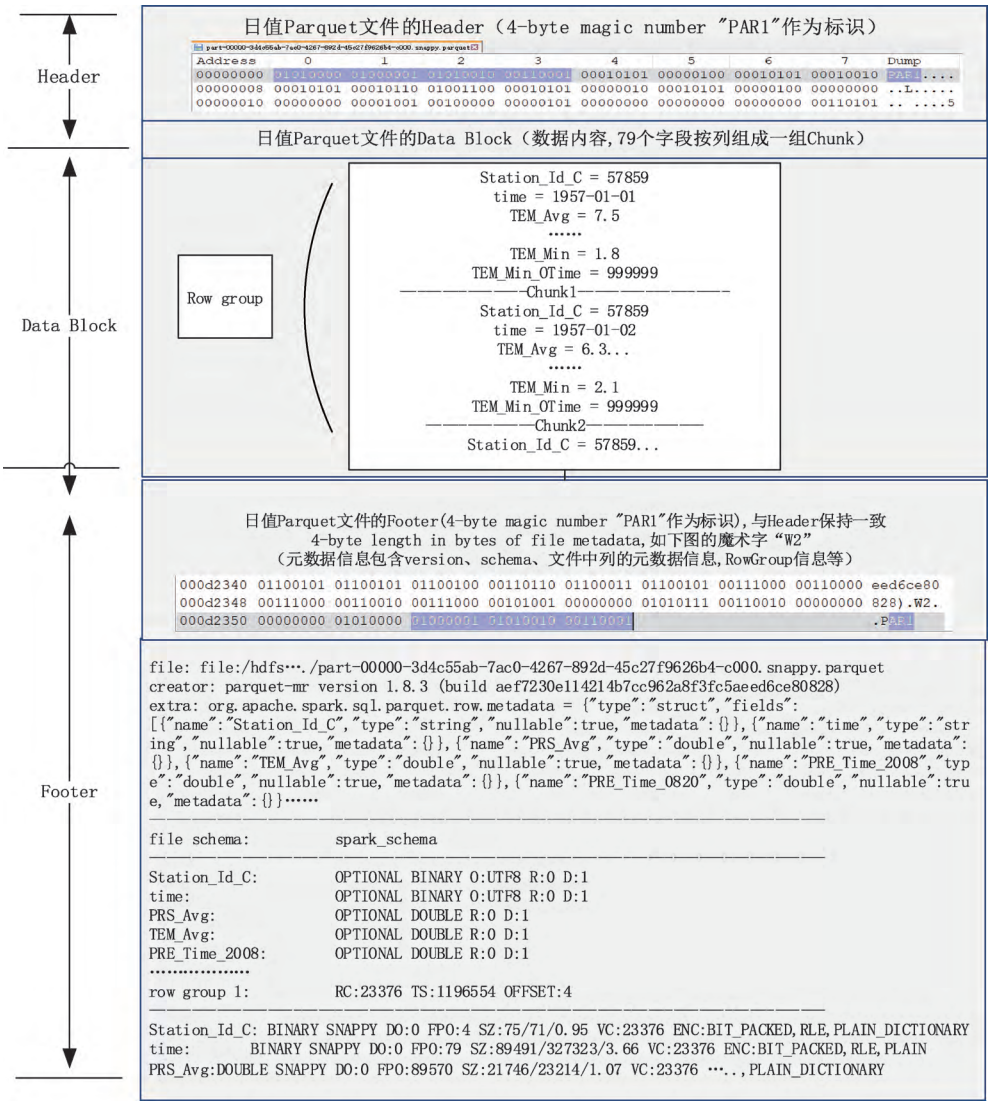

图 3　日值 Parquet 文件的 Header、Data Block 和 Footer 格式

表 4 是包含相同的数据内容的 TXT 格式与 Parquet 格式的文件数据容量对比情况,通过对表 4 进行分析发现,小时值的单个文件数据量最大,压缩率也最大(超过 95%),说明单个文件数据量越大压缩率就越大,证明 Parquet 格式的数据压缩性能非常出色,可以为 Hadoop 集群节省大量的存储空间,使用 Parquet 格式数据可以弥补 Spark 在数据压缩

表 4　格式转换前后存储占用情况

数据说明	文件数	TXT 数据量	Parquet 数据量	压缩率/%
小时值	2700	41.19 G	1.86 G	95.50
日值	2700	4.13 G	260.43 M	93.80
候值	2700	269.07 M	46.38 M	82.76
旬值	2700	135.01 M	35.87 M	73.40
月值	2700	86.94 M	38.30 M	55.90
年值	2700	7.76 M	26.77 M	−245.97

处理上不足，对后续提升 Spark 的处理性能有很好的促进作用。

4 SparkSql 嵌入 Broadcast 处理流程设计

利用 Parquet 格式转换提高了数据文件压缩和存储效率，Spark 从 1.6 版本开始对操作 Parquet 进行了优化，提升了其扫描吞吐量，数据文件的查找速度提高了 1 倍以上，采用 Parquet 有利于 Spark 的调度和执行。

由于在生成 Parquet 数据文件的 ETL 过程中并没有将站名、经度、纬度等台站参数数据列写入 Parquet 文件中，一是为了不增加要素列，二是因为这些参数信息会随着台站的迁移或台站撤销而发生变更，假如 Parquet 文件中写入上述参数数据，在以经纬度为例，如果某站点的经纬度参数发生变更，则需要对 Parquet 文件中相应列的数据进行修改和数据集重构，由于 Parquet 格式不支持实时大批量的数据删除修改操作，所以在实际的数据查询处理中，可以将所需的台站参数数据和经过 Spark 转换后的 DataFrame 主表，通过站号进行关联查询获取最终结果。

之前的 Spark 集群在做关联查询时，task 在处理过程中需要使用临时变量（例如参与关联查询的台站参数信息表），因此每个 task 在处理过程中会拷贝一份台站参数信息表作为副本，当遇到大数据量查询时会同时产生几百个 task，就需要同时拷贝几百份副本，使得集群网络 IO 开销短时间内剧增，导致集群的处理效率急剧下降。

因此，本文嵌入 Spark 的 Broadcast 广播变量解决台站参数的关联查询问题。Spark 的 Broadcast 广播变量是提前将一个轻量化的只读变量（数据表或文本）缓存在集群每个节点的 executor 进程中，避免在执行查询时才向所有 task 传递变量。本集群在 Spark 集群初始化时，通过 Spark 的 Broadcast 广播变量预先将台站参数数据拷贝到每个 executor 中，这部分数据只占用 executor 中很小一部分的内存，却能有效减少后续关联查询处理时节点之间的数据交互和归集操作，能将网络 IO 开销缩减 7～8 倍，从而提高查询的处理效率。图 4 为 Spark-Broadcast 关联查询处理流程[16]。

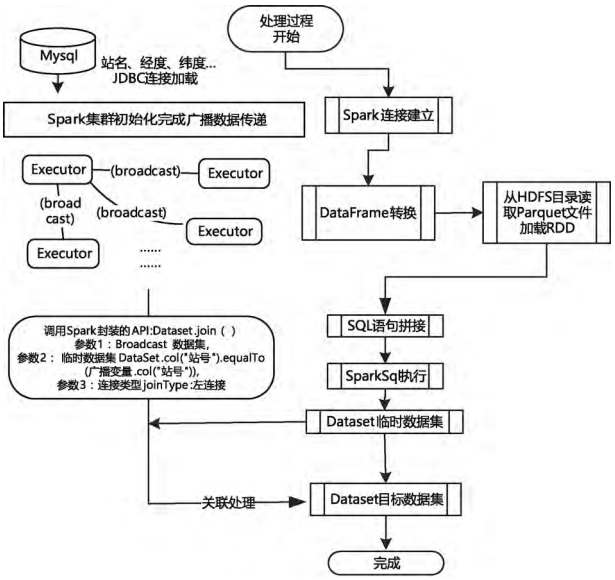

图 4 基于 Parquet 的 Spark-Broadcast 关联查询处理流程

Broadcast 初始化部分代码如下所示：

```
SparkSession SparkSession = SparkUtils.getSparkSession();    //获取 Spark 会话
SparkConTXT SparkConTXT = SparkSession.SparkConTXT();
Dataset<Row> stationInfoRDD = SparkSession.read().format("jdbc") //加载 mysql 库表到 Spark 中
        .option("url", dataUrl)          //jdbc url
        .option("user", username)          //数据库用户名
        .option("password", password)    //数据库密码
        .option("dbtable","station_info").load();//表名
Broadcast<Dataset<Row>>broadcast=SparkConTXT.broadcast(stationInfoRDD, ClassTagMYM.MODULEMYM.apply(Dataset.class));    //将数据库表广播到 Spark 的各个 Executor 中
Dataset<Row> sqlRDD = SparkSession.sql(sql); //Spark 执行 SQL 查询将 sql 查询结果和广播变量(站点信息)关联合并
Dataset < Row > join = sqlRDD.join(broadcast.getValue(),sqlRDD.col("Station_Id_C").equalTo(broadcast.getValue().col("station_code")),"left");
Dataset<Row> resultRDD = join.drop(dropCol);//去除多余的列
```

5 测试结果

基于优化后的 Spark 集群和嵌入 Broadcast 广播变量后的 SparkSql 处理流程，本文将以客户端查询大数据量的响应时效快慢为指标，对 Parquet 与 TXT 两种格式的处理响应时效进行比较，通过从客户端触发查询事务，到全部查询结果输出转换为可下载文本的全过程所耗费的时间，可以直观地对比出这两种方式的处理效率。

从表 5 的测试结果可以看出,当查询站点数、总数据量以及查询的目标列数较少时,Parquet 与 TXT 两种格式查询耗时区别很小;当需要查询的数据量呈倍增趋势时,以日值查询为例,Parquet 格式的查询耗时比 TXT 格式缩减了 1 倍以上;对于小时值查询,Parquet 格式的查询耗时比 TXT 格式缩减 4～5 倍,说明对于单个数据容量越大的文件,Parquet 格式比 TXT 格式查询效率提高更明显。从每种格式的查询耗时可以看出,Parquet 格式的查询耗时总体波动很小,而 TXT 格式查询耗时总体则波动很大。

表 5　Parquet 与 TXT 格式查询时效比对(建站至今)

统计类型	站数	要素/个	耗时/s		数据量/万条
			TXT 格式查询	Parquet 格式查询	
日值	5	1	0.7	0.6	12.7
		12	0.9	0.9	
		77	1.5	1.4	
	91	1	2.6	1.4	232.5
		12	2.9	1.6	
		77	4.1	1.9	
	2700	1	22	9	1611
		12	24	10	
		77	30	14	
小时值	91	1	13	3.1	5580
		7	20	3.2	
		22	23	3.7	
	2700	1	82	22	38692
		7	101	23	
		22	123	24	

从表 6 的并发查询结果可以得知,Parquet 格式在处理 100 用户的并发情况下效率提升更加显著,非常适用于海量数据的查询处理,这是因为 Parquet 格式文件的同列数据类型的同质性使其拥有高效的压缩编码,所以在 Spark 解码时可以快速跳过不符合条件的数据,有效减少冗余数据列,只读取目标列,提高了数据扫描和并行处理的效率。

表 6　100 用户并发 Parquet 与 TXT 格式查询时效比对

统计类型	站数	要素	时限/a	耗时/s		数据量/万条
				TXT	Parquet	
日值	91	12	70	28	9	230
	2700	12	10	202	112	23650
小时值	91	7	70	251	48	5580

6　结论与讨论

本文基于原有 Hadoop 大数据架构,对 Parquet 列式存储技术、Spark 分布式集群的并行处理配置参数以及关联查询流程进行深入研究与应用,实现了对海量气象数据的高效存储、查询和高并发访问,弥补了 CIMISS 支撑能力的不足。利用 Parquet 格式构建基于 A 文件的历史地面观测数据集,其高可用性可作为 CIMISS 和天擎数据环境下此类数据的备份,为气象业务提供更可靠的数据保障,对提升气候预测、评价和可行性分析等传统气象业务的综合应用能力具有重要意义。Parquet 格式实现代码简捷高效,特别适用于数据很少更新、处理时效要求不高的海量历史数据如海量网站日志的分析处理等应用场景,结合数据挖掘和人工智能技术,可衍生出更多的大数据处理模型。

在后续的工作中,将开展 Spark 即席查询的研究与应用,拓展更多的统计处理流程和发布更多数据产品的众创接口,以满足用户更多的数据服务需求,取得更好的数据应用效益。

参考文献

[1] 熊安元,赵芳,王颖,等. 全国综合气象信息共享系统的设与实现[J]. 应用气象学报,2015,26(4):500-512.

[2] 赵芳,熊安元,张小缨等. 全国综合气象信息共享平台架构设计技术特征[J]. 应用气象学报,2017,28(6):750-757.

[3] 杨润芝,马 强 ,李德泉,等. 内存转发模型在 CIMISS 数据收发系统中的应用[J]. 应用气象学报 ,2012,23(3):377-384.

[4] 赵文芳,刘旭林. SparkStreaming 框架下的气象自动站数据实时处理系统[J]. 计算机应用,2018,38(1):38-43+55.

[5] 陈东辉,曾乐,梁中军,等. 基于 HBase 的气象地面分钟数据分布式存储系统[J]. 计算机应用,2020,48(6). 2617-2621.

[6] George L. HBase: the definitive guide [M]. Sebastopol: O'Reilly Media, 2011.

[7] 贺正红,周娅,文缔尧,等. 面向 HBase 的大规模数据加载研究[J]. 计算机系统应用,2016,25(6):231-237..

[8] 黄志,詹利群,任晓炜,等. Hadoop 环境下基于 SparkSQL 海量自动站数据查询统计的初探 [J]. 气象科技,2019,38(1):768-772,871.

[9] 冯兴杰,王文超. Hadoop 与 Spark 应用场景研究[J]. 计算机应用研究,2018. 35(9):2561-2566.

[10] 季永华,孙超,刘一鸣,等. CIMISS 中气象观测资料处理入库

效率优化方法[J]. 气象科技,2017,45(2):29-34.
[11] 宋智,徐晓莉,张常亮,等. 应用分布式存储技术优化省级CIMISS数据服务能力[J],气象科技, 2019,47(3):433-438.
[12] 于俊. Spark 核心技术与高级应用[M]. 北京:机械工业出版社,2016.
[13] 肖卫青,杨润芝,胡开喜. Hadoop 在气象数据密集型处理领域中的应用[J]. 气象科技,2015. 43(5):823-828.
[14] 丁祥武,解书亮,李继云. 基于 Spark 的并行 ETL[J]. 计算机工程与设计,2017,38(9): 2580-2585..
[15] 罗祖兵 ,杨 晓 敏 ,严 斌 宇. 基于 Hadoop 和 Spark 的雷达数据序列模式挖掘系统[J]. 计 算 机 应 用 ,2019, 39(增刊 2): 169-174.
[16] 李涛,刘斌. Spark 平台下的高效 Web 文本分类系统的研究[J]. 计算机应用与软件,2016,33(11):33-36.

Research and Application of Spark Performance Optimization Analysis in Big Data Environment

HUANG Zhi SU Chuancheng SU Xiaohong
(Guangxi Meteorological Information Center, Nanning 530022)

Abstract: Aiming at a large amount of data query requirements of long-time series, multi-sites and multi-meteorological elements, the supporting capacity of the existing CMISS (China Integrated Meteorological Information Sharing System) is seriously insufficient. In this study, the monthly report data of historical surface meteorological records since the establishment of the meteorological stations in Guangxi and existing Hadoop cluster physical resources are used to redesign the ETL process, construct the Parquet format dataset, and complete HDFS conversion storage. Besides, the Broadcast variable of Spark is embedded to optimize the execution parameters of the Spark cluster, which improves the processing parallelism of the cluster and the association query efficiency of SparkSql. The results show that the maximum compression ratio of the Parquet format data set was more than 95%; the query efficiency of the one-time large amount of data was 1 to 5 times higher than the original and supported high concurrent access, providing effective technical support for the development of various related forecasting services.

Keywords: Hadoop; Spark; ETL (Extract-Transform-Load); Parquet; column store; Broadcast