在应用日志的行为分析算法方面, 主要基于 Spark 计算框架中的 Spark SQL 模块设计完成, Spark SQL 向用户提供了在大数据集上的类 SQL 查询功能, 同时还支持将原有持久化储存数据迁移到 Spark 环境下进行分析[15]. Spark SQL 的分析的核心模块是 DataFrame. DataFrame 是一个以命名列方式组织的分布式数据集. 它类似于关系型数据库中的一张表. DataFrame 可以由结构化数据、现存在的 RDD 或者从外部的关系数据库导入并转换而来[16]. 其中 DataFrame 包括: 用于描述列字段的集合 Schema 和行数据集 DataSet<Row>, 其中列描述信息用于方便下一步运行 Spark SQL 时查询列的标识, 行信息主要由分析数据信息组成.

根据油田管理评估要求需要统计的应用行为指标包括: 应用每小时的访问量、应用运行安全状况、各模块的使用量、应用模块异常信息、使用次数用户排名等 27 个行为指标. 由于 HBase 根据 rowkey 来检索数据并且支持以字符串匹配方式的扫描方法. 因此将时间和 IP 作为查询条件, 可以在各类应用间进行用户访问行为的关联分析, 进而描绘出用户每天在各类应用的停留时间和访问轨迹并推断出用户访问喜好.

本文在实现应用行为分析算法时, 将这些应用统计指标封装在一个算法内, 因此执行一次算法就可统计出所有应用指标. 在 Spark 执行行为分析时需要确定数据源和具体的分析算法: 其中算法选取由调度引擎来完成并提交给 Spark 集群来; 数据源来自于上一阶段的数据预处理算法处理后储存在 HBase 中的结构化数据, 需要调度引擎将要分析数据的起始行键提交给 Spark 集群. Spark 集群根据 HBase 起始行键拉取数据并执行指定算法程序, 完成处理后返回处理结果. 由于每个分析算法需要完成多个分析指标的统计, 因此需要根据分析指标制作多个 DataFrame 数据集. 因为在数据量过大时, 制作 DataFrame 数据集非常耗时. 因此制作数据集时要尽可能满足多个查询需求, 以减少重复制作数据集的处理时耗. 算法流程如下图 6 所示, 其中每一个分支流程对应一个分析指标.

每个行为指标具体的分析流程如下: 首先选取相应的字段并对字段数据进行格式转换、数据规约, 该过程主要借助于 Spark 提供的算子函数完成; 然后将 RDD 数据集转换成 DataSet<Row>数据集并加入列描述信息; 将数据集注册为临时表并运行查询语句; 最后格式化储存结果. 行为指标分析流程如图 7 所示.

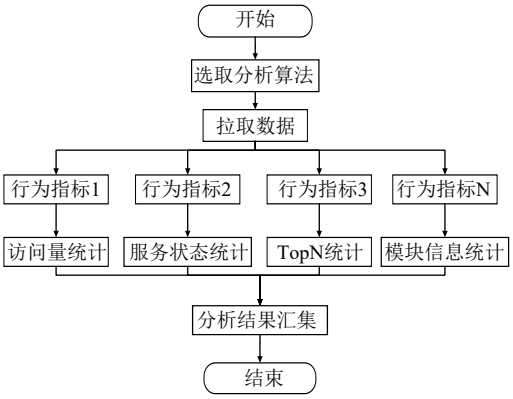

图 6 分析算法流程图

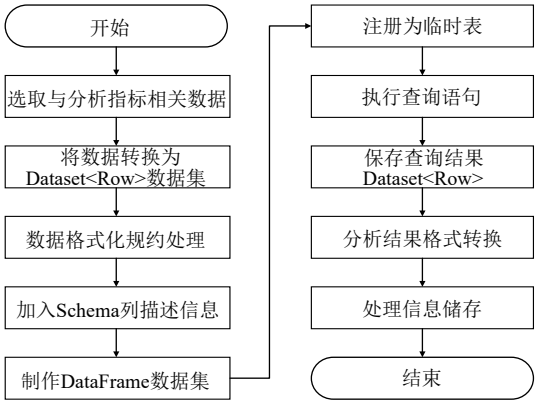

图 7 行为指标分析流程图

4 系统开发环境及实验分析

4.1 系统平台部署

系统平台的主要由三个部分组成: 数据收集层、数据分析层、Web 业务层. 依据实际生产场景, 系统开发环境部署规划如下, 数据收集层由 1 台日志储存服务器组成, 用于部署 Flume 日志收集框架. 数据分析平台是由 1 台主机点和 3 台计算节点组成计算集群, 各节点分别搭建 Hadoop 服务集群、Spark 服务集群、HBase 储存集群, 并在主节点搭建调度引擎程序. 业务层由一台 Web 服务器组成, 用于部署业务管理平台和业务数据库. 系统具体部署规划如图 8 所示.

4.2 实验结果分析

实验分析聚焦在数据分析层上, 主要统计各类算法的分析耗时, 本文的实验环境是由 4 台节点组成的集群环境, 日志文件储存在 HDFS 上, 基于 Spark 框架设计分析算法完成数据的分析, 基于 HBase 储存分析

数据, 并将 Spark 任务直接提交到 Yarn 上, 由 Yarn 完成资源分配和 Spark 任务调度. 其中各节点的环境信息和部署组件信息如表 2 所示.

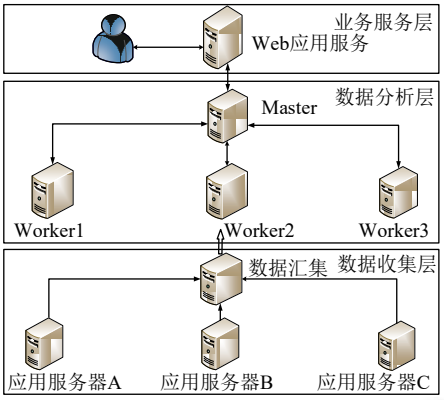

图 8 系统部署图

表 2 Spark 集群运行环境

主机名	操作系统	处理器	内存	硬盘	Hadoop	HBase	Spark
Master	CentOS7	4 核	8 GB	1 TB	2.7.3	1.2.0	2.1.0
Work1	CentOS7	4 核	8 GB	1 TB	2.7.3	1.2.0	2.1.0
Work2	CentOS7	4 核	8 GB	1 TB	2.7.3	1.2.0	2.1.0
Work3	CentOS7	4 核	8G	1T	2.7.3	1.2.0	2.1.0

实验分别在单节点环境和四节点组成的集群环境下测试了 2 个典型算法的耗时, 测试的算法为: 日志文本数据的预处理算法和应用行为指标分析算法 A(该算法主要用于统计 IIS 类型应用日志的行为指标, 包括统计每小时 IP 量、总 UV 量、每小时 PV、总 PV 量、各模块的访问量、TOPN 用户等 27 个行为指标). 日志预处理算法选取了某油田企业内部具有代表性的应用日志数据, 日志数据格式为 IIS W3C 格式. 选取并整理的单个日志数据大小依次为 106 MB、511 MB、1.1 GB、5.1 GB、9.8 GB、20 GB. 实验对比结果如图 9 所示.

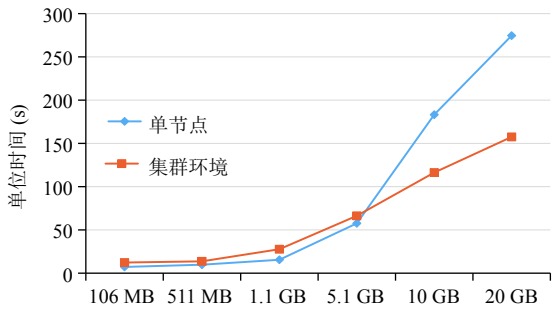

图 9 预处理算法时长对比图

由实验结果可以看出, 当数据量较小时, 单节点的处理时长较短; 当数据容量大于 5 GB 时, 集群环境下的处理时长远小于单节点的处理时长.

算法 A 的实验数据为储存在 HBase 中的结构化数据, 分别选取的数据集分别为: 956 887 条、1975 511 条、5911 511 条、29 479 329 条、63 906 591 条数据. 这里统计数据算法 A 的耗时为从数据加载到内存到预处理数据分析完成的时间 (不包括将数据写回数据库中的时间), 结果如表 3. 该算法的时间消耗主要在于: 制作 DataFrame 数据集的耗时和运行查询 SQL 的耗时, 算法 A 完成 27 个指标的统计, 需要制作 9 个 DataFrame 数据集, 运行了 35 次 SQL 查询.

表 3 算法 A 处理时长对比

数据量 (条)	单节点处理时长 (s)	集群处理时长 (s)
956 887	5.8	9.6
1975 511	13.3	15.8
5911 511	52.2	46.6
29 479 329	135.7	73.6
63 906 591	337.3	107.6

从实验结果可以看出, 当数据集增长到一定程度, 采用集群环境的处理耗时远低于单机处理耗时.

从两个分析算法的耗时统计可以得出: 当数据量大小在单节点处理能力范围内, 单节点处理时长要小于集群环境下处理时长; 若数据量过大, 采用集群环境的处理耗时要小. 这是由于集群环境下涉及到数据的分片, 任务间的通信, 代码序列化分发, 如果数据储存不在本地, 还会涉及到数据的移动问题, 此外处理时长还受主机磁盘 IO 传输速率、网络带宽的传输速率的影响, 这些多方面的因素都会影响处理时长. 因此集群环境在处理大批量数据时才会发挥优势.

5 结论

面对油田应用部署分散、种类繁多、数量庞大的复杂场景. 本文借助于各类主流的大数据处理框架实现对海量数据收集和储存; 在数据处理分析方面, 本文基于 Spark 计算框架设计了应用日志行为分析系统, 并设计了应用的安全状况分析和行为指标分析的算法; 此外为了方便运维人员使用该系统, 又基于 Web 设计了可视化的管理平台实现了各类框架的集成与管理. 该系统解决了油田进行海量应用数据分析的滞后性难题; 为油田迅速评估各类应用系统的运行状况和安全

状况提供了决策依据; 并为油田快捷高效的管理各类业务系统带来了一系列巨大优势.

参考文献

1 Wang GY, Butt AR, Pandey P, *et al*. Using realistic simulation for performance analysis of MapReduce setups. Proceedings of the 1st ACM Workshop on Large-Scale System and Application Performance. New York, NY, USA. 2009. 19–26. [doi: 10.1145/1552272.1552278]

2 Yang HC, Dasdan A, Hsiao RL, *et al*. Map-Reduce-merge: Simplified relational data processing on large clusters. Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. New York, NY, USA. 2007. 1029–1040. [doi: 10.1145/1247480.1247602]

3 Olston C, Reed B, Srivastava U, *et al*. Pig Latin: A not-so-foreign language for data processing. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. New York, NY, USA. 2008. 1099–1110. [doi: 10.1145/1376616.1376726]

4 Thusoo A, Sarma JS, Jain N, *et al*. Hive–a petabyte scale data warehouse using Hadoop. Proceedings of 2010 IEEE 26th International Conference on Data Engineering. Long Beach, CA, USA. 2010. 996–1005. [doi: 10.1109/ICDE.2010.5447738]

5 Thusoo A, Sarma JS, Jain N, *et al*. Hive: A warehousing solution over a Map-Reduce framework. Proceedings of the VLDB Endowment, 2009, 2(2): 1626–1629. [doi: 10.14778/1687553]

6 Koliopoulos AK, Yiapanis P, Tekiner F, *et al*. A parallel distributed WEKA framework for big data mining using Spark. Proceedings of 2015 IEEE International Congress on Big Data. New York, NY, USA. 2015. 9–16. [doi: 10.1109/BigDataCongress.2015.12]

7 Apache Spark is a fast and general-purpose cluster computing system. http://spark.apache.org/docs/latest/. [2017-12-30]

8 Quartz is a richly featured, open source job scheduling library. http://www.quartz-scheduler.org/documentation/. [2017-12-16].

9 Apache Felix is a OSGi framework and service platform. http://felix.apache.org/documentation.html. [2017-12-30].

10 顾兆军, 李晓红, 王伟, 等. Web 日志挖掘中的会话识别方法研究. 计算机技术与发展, 2012, 22(4): 45–49.

11 Facca FM, Lanzi PL. Mining interesting knowledge from weblogs: A survey. Data & Knowledge Engineering, 2005, 53(3): 225–241.

12 高彦杰, 倪亚宇. Spark 大数据分析实战. 北京: 机械工业出版社, 2016: 6–7.

13 Zhang F, Liu M, Gui F, *et al*. A distributed frequent itemset mining algorithm using Spark for big data analytics. Cluster Computing, 2015, 18(4): 1493–1501. [doi: 10.1007/s10586-015-0477-1]

14 张树壮, 罗浩, 方滨兴, 等. 一种面向网络安全检测的高性能正则表达式匹配算法. 计算机学报, 2010, 33(10): 1976–1986.

15 Armbrust M, Xin RS, Lian C, *et al*. Spark SQL: Relational data processing in spark. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. New York, NY, USA. 2015. 1383–1394.

16 Peng P, Zou L, Özsu MT, *et al*. Processing SPARQL queries over distributed RDF graphs. The VLDB Journal, 2016, 25(2): 243–268. [doi: 10.1007/s00778-015-0415-0]