DOI:10.11992/tis.201706016
网络出版地址:http://kns.cnki.net/kcms/detail/23.1538.TP.20171021.1350.014.html

基于 SQL-on-Hadoop 查询引擎的日志挖掘及其应用

何明[1],常盟盟[1],刘郭洋[2],顾程祥[2],彭继克[2]
(1.北京工业大学 信息学部,北京 100124; 2. 海通证券股份有限公司 信息技术管理部,上海 200001)

摘 要:随着计算机和网络技术的迅猛发展以及数据获取手段的不断丰富,海量数据的实时处理需求日益增多,传统的日志分析技术在处理海量数据时存在计算瓶颈。大数据时代下,随着开放式处理平台的发展,能够处理大规模且多样化数据的大数据处理系统应运而生。为了让原有的业务能够充分利用 Hadoop 的优势,本文首先研究了基于大数据技术的网络日志分析方法,构建了网络日志分析平台以实现万亿级日志采集、解析、存储和高效、灵活的查询与计算。对比分析了 Hive、Impala 和 Spark SQL 这 3 种具有代表性的 SQL-on-Hadoop 查询系统实例,并展示了这类系统的性能特点。采用 TPC-H 测试基准对它们的决策支持能力进行测试及评估,通过对实验数据的分析和解释得到了若干有益的结论。实现了海量日志数据计算与分析在证券领域的几种典型应用,为进一步的研究工作奠定了基础。

关键词:大数据;日志分析;数据挖掘;Hadoop;查询引擎;数据采集;索引存储;证券行业
中图分类号:TP391 **文献标志码**:A **文章编号**:1673-4785(2017)05-0717-12

中文引用格式:何明,常盟盟,刘郭洋,等. 基于 SQL-on-Hadoop 查询引擎的日志挖掘及其应用[J]. 智能系统学报, 2017, 12(5): 717-728.
英文引用格式: HE Ming, CHANG Mengmeng, LIU Guoyang, et al. Log mining and application based on sql-on-hadoop query engine[J]. CAAI transactions on intelligent systems, 2017, 12(5): 717-728.

Log mining and application based on sql-on-hadoop query engine

HE Ming[1], CHANG Mengmeng[1], LIU Guoyang[2], GU Chengxiang[2], PENG Jike[2]
(1. Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China; 2. Information Technology Management Department, Haitong Securities Co., Ltd., Shanghai 200001, China)

Abstract: With the rapid development of computing and networking technologies, and the increase in the number of data acquisition methods, the demand for real-time processing of massive amounts of log data is increasing every day, and there is a calculation bottleneck when traditional log analysis technology is used to process massive amounts of data. With the development of open processing platforms in the era of big data, a number of big data processing systems have emerged for dealing with large-scale and diverse data. To effectively apply the advantages of Hadoop to the original businesses, in this study, we first investigated network log analysis methods based on big data technology and constructed a network log analysis platform for the acquisition, analysis, storage, high-efficiency and flexible queries, and the calculation of trillions of log entries. In addition, we compared and analyzed three representative SQL-on-Hadoop query systems including Hive, Impala, and Spark SQL, and identified the performance characteristics of this type of system. We used the TPC-H testing reference to test and assess their decision-making support abilities. We drew some useful conclusions from the analysis of the experimental data. We also suggest a few typical applications for this analysis and processing system for massive log data in the securities fields, which provides a solid foundation for further research.

Keywords: big data; log analysis; data mining; Hadoop; query engine; data collection; indexed storage; securities business

随着互联网的飞速发展和逐层推进,企业内部的规模和业务量也不断增加,致使数据量猛增。企业网络中的计算机设备和网络组件持久地记录着海量的网络日志。日志文件是系统软硬件信息和用户行为信息记录的载体,通过日志分析能够实时获取设备、网络运行状态和用户行为交易等信息,有利于保证系统的稳定运行和来往业务的安全性。目前,较为成熟的日志集中管理系统解决了各类设

收稿日期:2017-06-07. 网络出版日期:2017-10-21.
基金项目:国家自然科学基金项目(91646201, 91546111, 60803086);国家科技支撑计划子课题(2013BAH21B02-01);北京市自然科学基金项目(4153058, 4113076);北京市教委重点项目(KZ20160005009);北京市教委面上项目(KM201710005023).
通信作者:何明. E-mail:heming@bjut.edu.cn.

备、服务器和应用日志的采集与格式统一问题,日志分析也从最初简单的正则匹配向结构化查询、报表和预测演进[1]。越来越多的行业领域面临海量(volume)、高速(velocity)和多样(variety)等多V挑战,大数据时代已真正到来[2-4]。

互联网中海量的信息为证券领域日志分析提供了丰富的数据支撑,如何利用大数据分析技术进行实时准确的日志分析成为重要的科学问题。在大型证券公司的内部网络中,随着网络带宽的迅速扩容日志量急剧增长且日志源众多,包括网上交易日志、移动证券日志和网站日志等主要系统的日志。以海通证券为例,目前在全国设有几十个节点,几百台服务器,峰值在线用户约几十万,每个节点各部署了1台负载均衡设备。网上交易应用服务器全天24小时将客户请求数据与应答数据实时或小批量定时写入磁盘日志文件,每台交易应用服务器的日志文件大小为100 MB~3 GB,总计在100 GB左右。同时,每台网上交易应用服务器还会生成一份发送给柜台程序的网关日志数据。此外,各节点负载均衡设备的日志采用SNMP协议进行采集,采集每个站点的网络流量、用户连接数据。每日合计有3亿多条日志,总量共计约300 GB。仅上述3类日志存储一年就将产生约108 TB数据,若接入更多设备、操作系统、业务平台日志,数据规模则更大。传统的日志处理方法在面对海量大数据时,其存储方式和计算能力都受到了限制,因此分布式存储和并行计算成为了新的发展趋势。如何采集、传输、存储、分析及应用大规模的日志数据,已成为证券行业在大数据时代下面临的重大挑战。

Hadoop[5]分布式处理平台为大数据存储和分析提供了有效的解决方案。在大数据应用方面,虽然学术界和工业界对大数据的关注各有侧重,但有一个共同的认识:大数据只有和具体的行业深入结合才能落到实处,才能产生真正的价值。通过前期的积累和算法的升级,大数据应用将对证券行业产生革命性影响。

本文的主要贡献如下:

1)研究基于SQL-on-Hadoop查询系统的性能特点,对比分析了Hive、Impala和Spark SQL这3种具有代表性的SQL-on-Hadoop查询系统实例,构建了海量日志采集与实时计算分析平台;

2)采用TPC-H测试基准对它们的决策支持能力进行测试及评估,通过对实验数据的分析和解释得到了若干有益的结论;

3)实现了大规模网络日志数据分析与计算在证券领域的几种典型应用。

1 相关工作

大数据技术在互联网领域海量网络日志分析和处理过程中得到了广泛的应用,日志分析系统主要包括日志同步、数据存储、分布式计算和数据仓库等相关技术。开源的日志分析系统如Facebook的Scribe[6],Apache的Chukwa[7],LinkedIn的Kafka[8],Cloudera的Flume[9]等。Facebook公司庞大的用户群体产生了大量的信息与社交数据,现有8亿多用户的信息需要处理,产生了大规模的数据和日志;同时,离线的大规模数据分析计算已无法满足实时数据分析的用户需求,Scribe结合了Google的分布式文件系统GFS[10](google file system,GFS)。操作流程是收集异构数据源上的日志,集中存储到分布式文件系统,从而在此基础上进行统计分析。Amazon基于S3和EC2,开发了Amazon EMR来提供大数据处理服务,可以将数据分布在可重新调整大小的EC2集群中进行处理,包括日志分析、索引、数据仓库和机器学习等。阿里巴巴集团使用目前国内最大的Hadoop集群"云梯"进行各部门产品的线上数据备份、系统日志以及爬虫数据分析,并建设开放平台为个人和企业提供各种增值服务。腾讯微信等应用产品拥有上亿级别的用户,产生了海量的个人用户日志数据,这些数据中蕴藏着巨大的商业价值,并提出"大数据营销"的概念。人人网基于Hadoop的Hive[11]、HBase[12]和Streaming[13]组件,构建了SNS推荐平台进行分析计算、内容推荐等工作。百度的高性能计算系统规划中的架构将有超过1万个节点,每天的数据生成量在10 PB以上,主要用于日志的存储分析以及统计挖掘等功能。Wei等设计了Analysis Farm摒弃了传统的关系型数据库(relational database management system,RDBMS),利用NoSQL(not only SQL)数据库MongoDB构建了可横向扩展的日志分析平台,以支撑NetFlow日志存储和查询[14]。Rabkin等设计了基于Hadoop的日志收集和分析系统Chukwa,日志处理程序在MapReduce框架上开发[15]。文献[16-17]从原位分析的角度出发,分别实现了针对大规模日志分析的MapReduce(In-situ MapReduce)和Continuous处理机制,但MapReduce模型计算代价很大,并不能

很好地支持迭代运算。

然而 HDFS[18] 和 MapReduce[19] 大数据处理架构主要是针对静态数据的批处理,在运算过程中产生的大量 I/O 操作无法保证处理过程的实时性。针对上述问题,本文将研究基于 SQL-on-Hadoop 查询引擎构建网络日志分析平台,通过使用广泛的标准 SQL 语言来实现快速、灵活的查询性能。通过利用 TB 级日志数据对存储、查询性能进行测试、优化和比较,构建具有稳定性、高性能、可扩展性、易用性和安全性的网络日志统一采集查询和监控平台,以满足对 TB 或 PB 级容量和万亿日志管理的应用需求,为面向证券行业的日志大数据分析及其应用提供技术支撑。

2　基于 Hadoop 的结构化数据处理

网络日志源的种类具有多样性的特点,包括结构化、半结构化和非结构化的数据。不同类型的日志存储方式有所不同。日志管理系统的采集器对不同格式的日志进行标准化处理,从而以结构化的形式进行日志存储和分析。本文所采用的源数据主要分为文本数据、数据库数据和实时/准实时数据等。

2.1　HDFS 数据采集

网络日志的生成是分布式的,与传统的日志管理系统一样,日志采集是本文平台的基础。本文平台采集的日志直接存储在 Hadoop 文件系统(HDFS)中,由于平台构建于 Hadoop 之上,能够处理海量分布式存储的日志数据,同时易于水平扩展,本文的日志数据基本流程按功能可划分为 5 层,如图 1 所示。

1)原始数据层:业务上完成日志格式梳理,系统运行日志支持实时访问和采集接口。

2)数据采集层:主要负责通用的日志数据解析、高效采集和安全可控。

3)数据处理层:主要包括对日志数据的批量式处理和实时处理。

4)数据服务层:主要提供标准的数据访问接口 ODBC、JDBC、HIVE 等。

5)数据展示层:实现实时监控类和报表类数据的展示。

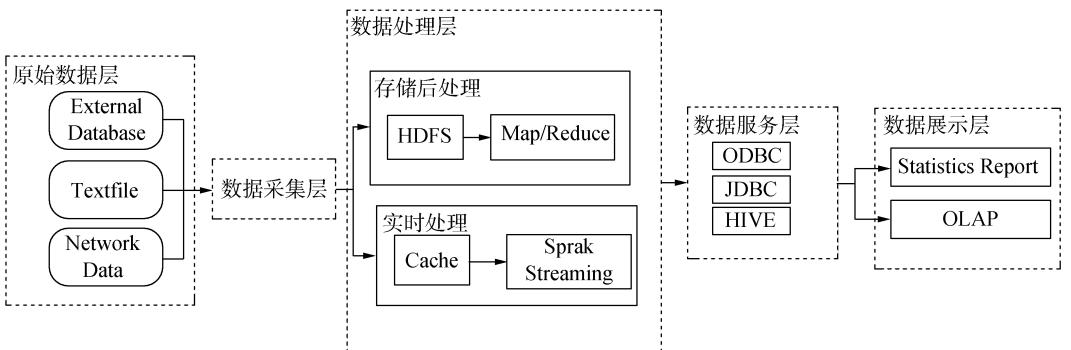


图 1　日志数据处理基本流程

Fig.1　Basic log data processing framework

根据应用需求,本文日志的采集方式分为以下 3 种。

1)文件导入:对已分布在个服务器磁盘的日志文件,经网络文件系统挂载,直接将日志文件导入 HDFS。该方式允许日志文件批量可靠导入,可在网络利用率低谷时段进行传送。

2)流数据导入:基于 Apache Flume[20] 构建,实现多个日志源数据实时汇聚,接收网上交易应用服务器和网络设备发送的日志。

3)RDBMS 导入:为实现与现有日志系统兼容,基于 Apache Sqoop[21],实现与 Oracle、MySQL 和 PostgreSQL 等 RDBMS 对接,支持直接导入存储在上述数据库中的数据记录。Sqoop 同时可以将 SQL-on-Hadoop 处理结果输出到 RDBMS,供现有的日志分析系统进行报表及可视化处理。

2.2　SQL-on-Hadoop 查询引擎

SQL 是结构化数据的查询语言,SQL-on-Hadoop 是构建在 Hadoop 之上的 SQL 查询系统,利用 Hadoop 能够进行海量数据(TB 级别以上)的处理。目前已有的 SQL-on-Hadoop 系统大致可以分为两大类:第一类将 SQL 查询转换为 Map-Reduce job;第二类系统基于 MPP(massively parallel processing)的设计方式,仅仅使用 Hadoop 作为存储引擎,上层自行实现分布式查询的逻辑。第一类系统的代表是 Facebook 的 Hive。Hive 是原始的 SQL-on-Hadoop 解决方案。它是一个开源的 Java 项目,能够将 SQL 转换成一系列可以在标准的 Hadoop TaskTrackers 上运

行的 MapReduce 任务。如图 2 中的 Hive 架构部分所示,Hive 通过一个 metastore(本身就是一个数据库)存储表模式、分区和位置以期提供像 MySQL 一样的功能。它支持大部分 MySQL 语法,同时使用相似的 database/table/view 约定组织数据集。Hive 内部机制是基于 MapReduce,从而导致了计算过程中消耗大量的 I/O,降低了运行效率。Impala[22] 是由 Cloudera 构建的一个针对 Hadoop 的开源的 MPP (massively parallel processing)“交互式”SQL 查询引擎。Impala 同样提供了一种 SQL 查询方法,如图 2 中的 Impala 架构部分所示,与 Hive 不同的是,Impala 并没有使用 MapReduce 执行查询,而是使用了自己的执行守护进程操作本地磁盘文件。由于没有 MapReduce 开销以及磁盘 I/O、查询语句编译等一系列优化,Impala 通常要比 Hive 具有更快的数据访问性能[23]。Impala 共享 Hive 的 metastore,可直接与 Hive 管理的数据互操作。Spark[24] 使用轻量级的线程作为执行器,减少了执行作业的开销,同时提高了调度的响应速度,如图 2 中的 Spark 部分所示。Spark SQL 是在 Spark 之上搭建的 SQL 查询引擎,支持在 Spark 中使用 Sql、HiveSql、Scala 中的关系型查询表达式。

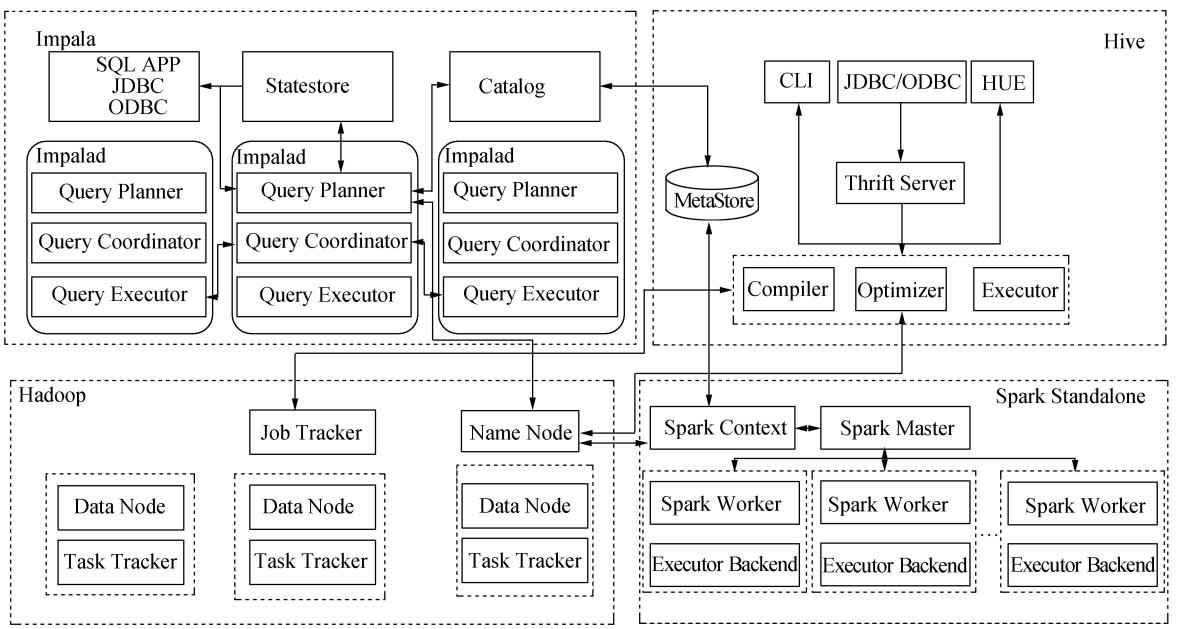


图 2 Hadoop、Hive、Impala 与 Spark 执行结构图

Fig.2 Structure for implementation of Hadoop, Hive, Impala and Spark

2.3 结构化数据存储与压缩

目前,很多研究者提出了在 Hadoop 中优化结构化数据存储的方法。He 等[25] 提出的 RCFile 格式旨在提高数据导入和处理效率。它首先将数据水平分割为多个行组(row-group),然后对每个组内的数据垂直分割成列存储。列存储将数据表同一列的数据连续存放,当查询只涉及部分列时,可大幅减少所需读取的数据量。ORC(optimized RCFile)是对 RCFile 的改进,解决其在数据类型和性能上的多个局限性,改善查询和空间利用效率。Parquet 是 Hadoop 生态圈中一种新型列式存储格式,灵感来自于 2010 年 Google 发表的 Dremel 论文[26],它可以兼容 Hadoop 生态圈中大多数生态框架(Hadoop、Spark 等),被多种查询引擎支持(Hive、Impala、Spark SQL、Drill 等),并且它与语言和平台无关的。表 1 比较了本文 2.2 节描述的 3 种查询引擎从 HDFS 上读取多种格式的数据格式的支持。Text 是原始的文本数据,通常为 CSV 或其他特定字符分隔。Hive 的格式支持更为全面,由于 Impala 和 Hive 共享 metastore,因此本文平台实际应用中通常由 Hive 导入数据而后台使用 Spark SQL 查询。

表 1 Hive、Impala 和 Spark SQL 数据格式支持比较

Table 1 Data format comparison of Hive, Impala and Spark SQL

数据格式	Hive		Impala		Spark SQL	
	查询	插入	查询	插入	查询	插入
Text	√	√	√	√	√	√
RCFile	√	√	√	—	—	—
ORC	√	√	—	—	—	—
Parquet	√	√	√	√	√	√

数据压缩是另一种性能优化方法。压缩一方面节省存储空间,另一方面在相同磁盘 I/O 速度可读写更多记录。Hive、Impala 和 Spark SQL 均支持直接查询压缩的数据文件,常用压缩算法有 Gzip/Zlib 和侧重于解压缩速度的 Snappy。ORC 格式本身已内嵌轻量级的压缩机制。

2.4　结构化数据处理算法

RDD 数据集包含对父 RDD 的一组依赖,这种依赖描述了 RDD 之间的传承关系。RDD 将操作分为两类:Transformation 与 Action。Transformation 操作不执行运算,只有当 Action 操作时才触发运算。在 RDD 的实现机制中,基于迭代器的接口实现原理使得数据的访问更加高效,同时避免了大量中间结果对内存的消耗。Spark SQL 包含了结构化数据和数据之上进行运算的更多信息,Spark SQL 使用这些信息进行优化,使得结构化数据的操作更加高效和方便,基于 Spark SQL 的数据操作流程如下。

算法 1　SparkSQLonRdd(<input>,<context>)

输入　Kafka 输入数据流 input, Spark 上下文 context;

输出　分布式集合 dataframe。

1) DStream line:Kafka->DStream(input);
2) 获取 Kafka 流数据输入;
3) SqlContext sc = new SqlContext(context);
4) DStream<Row> rdd=line.map;
5) new Function;
6) public Row call(T) {};
7) 创建 Row 对象;
8) List<StructField> sf = new; List<StructField>();
9) Struct Fields.add(CreateDataType(<Column>));
10) 重复步骤 9) 创建逻辑表结构;
11) Struct Type st: DataTypes.CreateStructType(sf);
12) DataFrame df :
13) sc->DataFrame(rdd, st);
14) df.RegisterTable(<Table Name>);
15) DataFrame dataframe=sc.sql(<Sql Query>);
16) Return dataframe。

算法 2　RddProcessing(<input>)

输入　Kafka 输入数据流 input;

输出　数据集对象 record。

1) 数据采集与预处理
①SparkConf conf = new SparkConf();
②创建上下文对象;
③StreamingContext(conf, Interval);
④Map<E,T> Offsets=kafka.getOffset();
⑤获取 kafka 读取偏移量;
⑥DStream stream;
⑦KafkaUtils.createDStream(input);
⑧Return stream。
2) RDD 数据处理
①stream.foreachRDD;
②new VoidFunction<RDD>>();
③call(RDD<MessageAndMetadata> rdd);
④HasOffsetRanges offrange = rdd.rdd();
⑤合并请求应答,并解析存储数据;
⑥rdd.mapPartitionsToPair;
⑦ new FlumeKafkaFunction();
⑧foreachPartition(ProceFunction());
⑨kafka.setOffset(offrange);
⑩保存 kafka 读取偏移量。
3) ProceFunction 数据后处理
①Iterator<Tuple2<T, KafkaData>> iter;
②while (iter.hasNext());
③KafkaData data = iter.next()._2();
④json = data.getData();
⑤Record record =Object(json, class);
⑥record.setCollect_time;
⑦data.getExtData(TIME));
⑧Utils.save(item_topic, record);
⑨Return record。

其中,RDD 根据数据记录的 key 对结构进行分区。分片数据采用迭代器 Iterator 流式访问,hasNext 方法是由 RDD lineage 上各个 Transformation 携带的闭包函数复合而成,使得对象被序列化,通过网络传输到其他节点上进行装载运算。Iterator 每访问一个元素,就对该元素应用相应的复合函数,得到的结果再流式地存储。

3　平台架构与集群环境部署

3.1　平台架构与处理框架

本文基于 Hadoop,构建证券交易应用服务器和网络设备海量日志采集、解析、存储与实时计算分析平台,平台的核心架构如下。

1)数据采集层:负责实时采集来自通达信、恒生、核新的网上交易应用服务器全天 24 小时的客户

请求应答数据以及网络设备日志数据,为大数据分析平台提供数据源。

2)数据汇集层:将各个数据采集节点的日志数据源源不断地汇集到各自的集群。

3)数据缓冲层:根据不同的 Topic 对海量日志数据进行缓冲,有助于控制和优化数据流经过系统的速度。

4)数据分发与解析处理层:负责数据的解析、勾对、计算和分发。

5)数据存储与计算层:用于存储、管理日志数据,支持多维检索、统计分析和查询处理。

6)应用层:负责面向终端用户提供日志分析与管理的泛在接入,提供实时运维监控、实时预警、明细毫秒级查询以及实时报表输出等应用。

可以看到,在这个大数据分析体系结构中,系统支持 TB 级、PB 级或者更大规模数据的分析和处理;系统可以处理结构化数据、非结构化和半结构化数据,有良好的扩展性。基于上述平台结构,本文设计了能够有效地利用大数据技术解决海量系统访问日志多条件实时快速查询的处理框架,如图 3 所示。

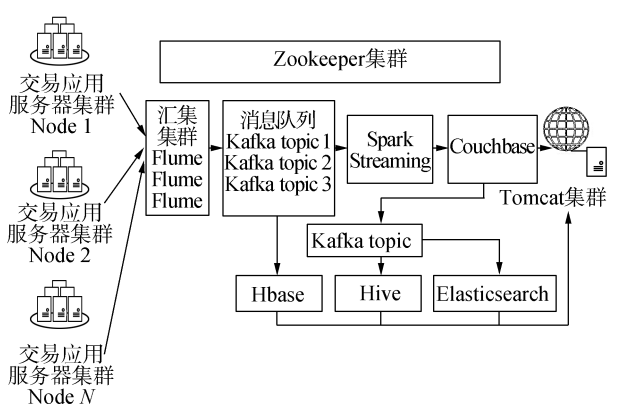


图 3 处理框架

Fig.3 Processing framework

该处理框架能够保证平台系统如下的几个特性。

1)实时性:实时采集 Agent 包,从产生时刻起到实时采集,再到传输到数据中心,整个时间间隔控制在 1 s 内实时勾对、解析等计算,并保存到数据中心的集群,这个过程的时间间隔控制在 3~5 s。

2)准确性和完整性:传输通道实现不重传、不漏传、断点续传,保证数据完整性。

3)安全性:非对称加密算法对传输的日志数据进行加密,使用 SSL/TLS 协议,保障网络传输通道的安全性。

4)稳定性和可靠性:基于成熟的、经过实践验证稳定可靠的 Hadoop 技术组件服务器节点非常容易实现横向扩展,分布式环境保障集群中的任意一台服务器出现宕机时不影响系统的稳定可靠运行。

3.2 环境部署

基于 Hadoop 的网络日志分析平台在海通证券网络信息中心的搭建部署,如图 4 所示。共 42 台服务器,其中 11 台是 Flume 汇聚节点(256 GB 内存,2×600 GB,RAID1 阵列),5 台 Kafka 节点(256 GB 内存,2×600 GB,RAID1 阵列),3 台 Couchbase 节点(512 GB 内存,2×600 GB,RAID1 阵列),5 台 Zookeeper 节点(256 GB 内存,2×600 GB,RAID1 阵列),2 台作为 Namenode(256 GB 内存,2×600 GB,RAID1 阵列),14 台是 Datanode 节点(256 GB 内存,2×600 GB,RAID1 阵列,2×600 GB,RAID1 阵列 +6×2 TB,RAID0 阵列),2 台 Tomcat(256 GB 内存,2×600 GB,RAID1 阵列)。

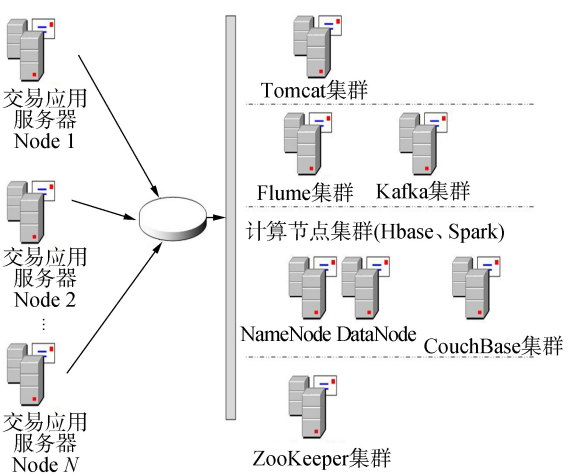


图 4 集群拓扑图

Fig.4 Cluster topology

所有节点通过 10 GB 以太网互联。Hadoop 部署采用 Cloudera 的发行版,版本为 CDH5.5.0,HDFS 总容量近 60 TB。接入日志分析平台的数据来自网上交易应用服务器日志数据和网络设备日志数据。网上交易日志每天产生的记录数约 1.2 亿条,体积约 100 GB;网络设备日志数据日志每天的记录数约 650 万条,体积约 6 GB。

4 实验与性能评估

4.1 实验环境与数据集

我们采用的实验环境为 7 台物理测试机构建的集群,选取 2 台机器作为主节点,其余作为计算节点进行 SQL-on-Hadoop 实验,测试集群拓扑如图 5 所示。

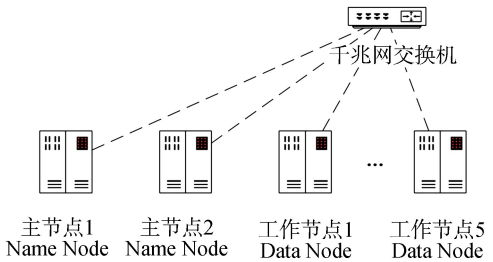


图 5　测试环境拓扑图

Fig.5　Test environment topology

实验采用针对 OLAP 应用的 TPC-H 测试基准来评估执行引擎的性能。TPC-H 面向商务采购应用,其数据库模式遵循第三范式。性能评测基准定义了 22 个复杂 SELECT 语句和 2 个更新数据语句,遵循 SQL-92 标准。数据库的规模由自带的扩展因子(scale factor,SF)决定,有 10 个级别,从 1 GB 到 100 TB 不等供用户选择。TPC-H 基准以每小时内执行的查询数作为度量标准,在工业和科研领域当中应用广泛。

文献[23]讨论了 ORCFile 和 Parquet 两种列式存储格式的性能差别,通过空间使用和查询性能比较,认为 Parquet 针对文本文件压缩率较高,从而节省了 HDFS 的存储空间,同时减小了磁盘 I/O 的开销,但是解压缩会占用部分计算资源,对性能有一定影响。因此,本文采用 Parquet 紧凑的列存储格式,并选用了压缩比和解压速度较为均衡的 Snappy,相对原始文本日志节省了近 70%的空间。

本文实验使用 TPC-H 作为测试数据集,在 SF = 300 的数据规模上进行测试,其描述和相关压缩处理如表 2 所示。

表 2　实验数据集

Table 2　Experimental dataset

Table	Rows	Volume	Parquet/Snappy
Lineitem	1 799 989 091	187.8 GB	50.8 GB
Order	450 000 000	43.6 GB	15.8 GB
Partsupp	240 000 000	32.2 GB	12.2 GB
Part	60 000 000	8.7 GB	2.3 GB
Customer	45 000 000	7.5 GB	3.9 GB
Supplier	3 000 000	0.4 GB	0.23 GB
Nation	25	3.1 KB	2.1 KB
Region	5	620 B	390 B
总计	2 597 989 121	280.2 GB	85.2 GB

4.2　性能评估

本文选择 SQL-on-Hadoop 作为基础查询引擎,对 3 个引擎的处理时效性进行分析。从表 3 中各个引擎的总运行时间可以看出,Impala 比 Hive 快了 1.5 倍,Spark SQL 比 Hive 快了 2.7 倍。

表 3　查询执行时间比较

Table 3　Comparison of query runtime

TPC-H 查询子句	执行时间/s		
	Hive	Impala	Spark SQL
Q1	115	25	14
Q2	161	146	18
Q3	274	173	83
Q4	152	211	126
Q5	342	326	64
Q6	146	38	12
Q7	678	339	87
Q8	468	331	90
Q9	1104	735	467
Q10	298	163	40
Q11	246	35	30
Q12	126	22	14
Q13	293	142	66
Q14	107	101	21
Q15	277	16	12
Q16	408	267	54
Q17	523	428	254
Q18	449	549	121
Q19	243	226	283
Q20	419	170	91
Q21	681	713	792
Q22	82	67	92
Total	7 592	5 223	2 831

实验结果表明,Impala 在 Q1、Q6、Q12、Q15 上的性能优于 Hive,查询语句结构如下:

```
SELECT
{
field1, field12,
SUM(field3) as alias1,
```

```
SUM(field4) as alias2,
……,
AVG(field5) as alias3,
……,
COUNT( * ) as alias4
}
FROM TableExpression
WHERE
[field6 <= date'yyyy-mm-dd'-interval '[DELTA]' day (3)]
GROUP BY [field6, field7]
ORDER BY [field6, field7]
```

根据该语句的执行计划,可以判断查询时对整个表进行了遍历。对于 Spark SQL 而言,其在大多数查询上的表现优于 Hive 和 Impala。由于 Spark 的接口丰富和 SQL 优势,在执行查询时的速度较快。

4.3 Q22 资源消耗情况

Q22 的查询语句如下:

```
SELECT
cntrycode, COUNT( * ) as numcust, sum(c_acctbal) as totacctbal
FROM (
SELECT substring(c_phone from 1 for 2) as cntrycode,
c_acctbal FROM customer
WHERE substring(c_phone from 1 for 2) in
('[I1]','[I2]','[I3]','[I4]','[I5]','[16]','[I7]')
and c_acctbal > (
SELECT AVG (c_acctbal)
FROM customer WHERE c_acctbal > 0.00
and substring (c_phone from 1 for 2) in
('[1]','[12]','[13]','[14]','[15]','[16]','[I7]'))
and not exists (SELECT * FROM orders where o_custkey=c_custkey)
) as custsale
Group BY cntrycode ORDER BY cntrycode;
```

如图 6 所示,Q22 中作业由 3 个子查询组成。子查询 S1 对 customer 表进行扫描并将结果保存到临时表 Temp1 中;子查询 S2 对 Temp1 进行聚集操作 AGG1 后将结果保存到临时表 Temp2 中;子查询 S3 在与表 Orders 执行聚集操作 AGG2 后依次与 Temp1 和 Temp2 进行关联操作求笛卡尔乘积 AGG3 然后排序。

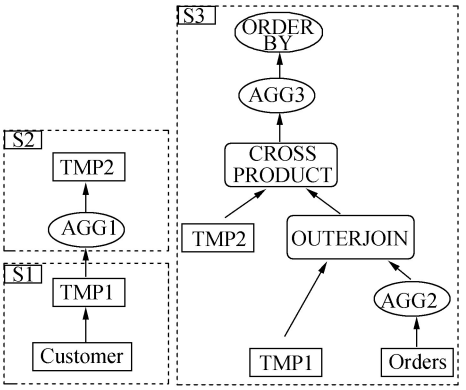


图 6 Implementation of Q22

Fig.6 Q22 的执行过程

实验分析对比了不同的查询方式在运行 Q22 时集群资源使用情况(如图 7~11 所示),包括 CPU、内存、网络、磁盘 I/O。注意到,在查询 Q22 执行过程中,Impala 对集群资源的占用是最少的,其次是 Hive,Spark SQL 占用资源最多。由于 Spark SQL 是基于内存计算的框架,所以在内存占用方面和磁盘读取上更为明显。

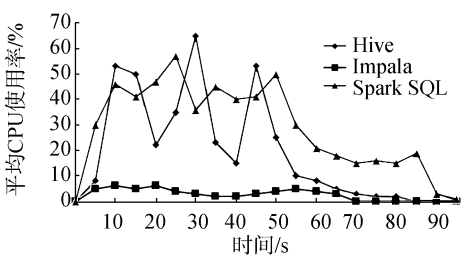


图 7 集群平均 CPU 使用率

Fig.7 Average cluster CPU usage

由于 Hive 和 Spark SQL 均在 JVM 之上运行,对 CPU 和内存的使用依赖于 JVM。如图 7 所示,Impala 的 CPU 占用时间要明显少于 Hive 和 Spark SQL,这是由于 Impala 在执行查询过程中,在每个计算节点上运行只占用一个 CPU 线程。而 Hive 和 Spark SQL 在 CPU 使用上的优化完全依赖于 JVM。如图 8 所示,Impala 和 Hive 内存使用率明显小于 Spark SQL,同时使用线程来执行耗费资源较多的 Executor Backend 进程。

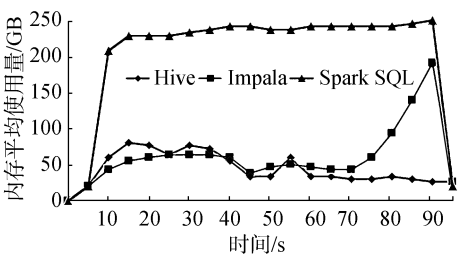


图 8 集群内存平均使用量

Fig.8 Average cluster memory usage

在磁盘性能方面,Impala 和 Hive 的磁盘读取速率优于 Spark SQL。从图 9 可以看出,Hive 和 Impala 数据访问量在 S1 时相对一致,高于 SparkSQL。在 S2 中,Impala 数据访问量较小,Hive 次之,SparkSQL 最高。在图 10 中,Impala 在 S1 和 S2 执行结束后将结果写入 HDFS 时,对磁盘的写入速率迅速增加。

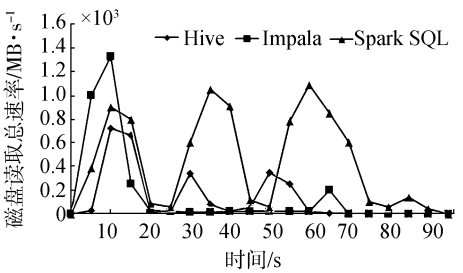


图 9　集群磁盘读取总速率

Fig.9　Cluster disk read speed

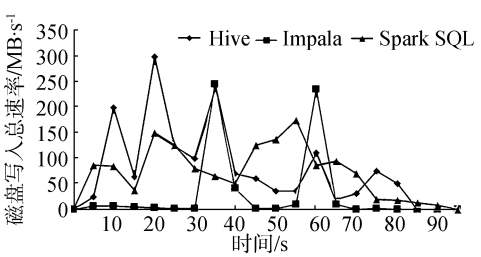


图 10　集群磁盘写入总速率

Fig.10　Cluster disk write speed

在图 11 中,Impala 在最后一个阶段的网络流量迅速增长,主要由于执行过程中的内部表连接产生的结果通过网络传输给其他节点导致。

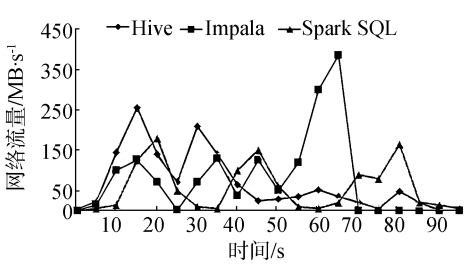


图 11　集群网络流量

Fig.11　Cluster network traffic

综上所述,Spark SQL 执行效率相对较快,它的查询速度比 Hive 要快 2.7 倍。然而,当查询总大小超过内存大小时,Impala 则无法查询。Hive 处理的结果准确率较高,处理速度较慢。因此,Hive 比较适用于批处理应用;Impala 适合交互式查询,系统的稳定性还有待提高;Spark SQL 能够降低 Hive 的延迟,比较适合多并发和流数据处理场景。

通过以上的实验分析和比较,从文件格式角度来讲,本文选择能够更好地适配 Spark SQL 的 Parquet 列式存储格式,以便快速地从 HDFS 中扫描找到相应的数据;从压缩角度来看,本文采用 Snappy 压缩方式,以减少数据输入量和加快查询速度;从 Spark SQL 自身特性分析,Spark SQL 基于内存计算执行速度快,可以操作 Hadoop 上多样化格式的数据并进行高效的结构化分析与处理,提供可用性更好的 API 进行数据分析,更加灵活且易扩展。鉴于此,综合考虑以上几个方面并结合应用驱动的大数据分析与计算实际需求,本文选择 Spark SQL 作为 SQL-on-Hadoop 查询系统,以适应快速证券大数据分析与计算场景下的高并发实时查询应用需求。

5　实际应用

在本文实现基于 SQL-on-Hadoop 网上交易日志实时分析与计算平台上,目前已存储约 60 TB 的网上交易日志,并开发和移植了实时监控、统计分析、明细查询等实际应用。

5.1　实时运维监控

对分布在全国各个节点和服务器的状态实时监控并对各种状态进行及时判断和处理,能够对整个系统的使用状况有宏观的把控。实时运维监控主要包括技术指标监控、业务指标监控和客户分布。

1)如图 12 所示,技术指标监控主要针对实时的请求延迟、成功率、系统冗余(带宽流量/在线数)等指标进行监控。数据从千万级别的当日日志数据中实时提取,从采集到存储达到秒级实现。延迟情况主要包括登录、委托、查询(资金查询)和转账这 4 类业务单位时间段内的平均耗时和峰值情况。系统冗余用于指示系统的资源使用情况,包括系统容量情况和系统带宽使用情况,能实时展示系统当前冗余,有助于系统管理员及时掌握当前系统的使用情况。成功率主要包括登录、委托、转账这几类业务单位时间段内的处理成功率情况。

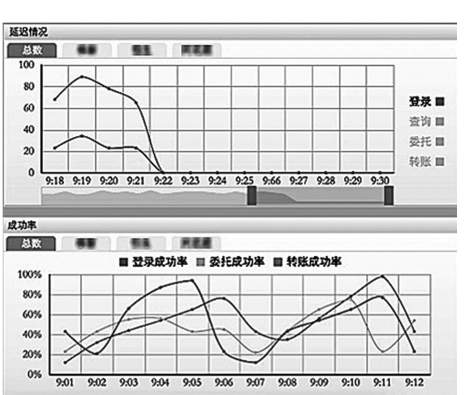


图 12　技术指标监控

Fig.12　Technical index monitor

2)如图 13 所示,业务指标监控主要针对登录情况、转账情况、委托情况和实时在线人数等指标进行监控。可以从千万级别的当日数据中实时观察当前系统的客户登陆数、系统发生的交易数量和转账金额等情况,整个过程实现秒级响应。

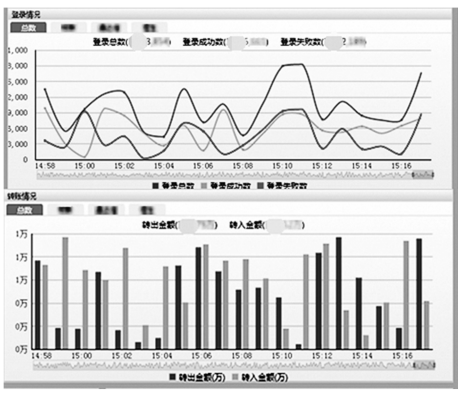


图 13 业务指标监控

Fig.13 Business index monitor

3)如图 14 所示,客户分布主要针对实时在线客户与委托分布,站点和来源省份的分布。在线分布从千万级别的数据中指示一段时间客户的登陆来源和委托来源分布。系统通过登陆和委托源 IP 关联全球 IP 分布区域得出客户的分布情况, IP 分布来自 10 亿级别的 IP 分布数据源。

图 14 客户分布

Fig.14 Customer distribution

5.2 客户交易行为监控

日志分析更有价值的应用在于发现客户的异常行为。如图 15、16 所示,通过大数据平台,可以实时掌握不同区域和营业部活跃用户的分布,为业务部门做绩效考核、精准营销等提供数据支撑。

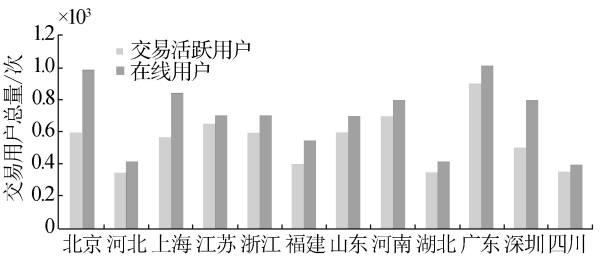


图 15 活跃用户分布

Fig.15 Active user distribution

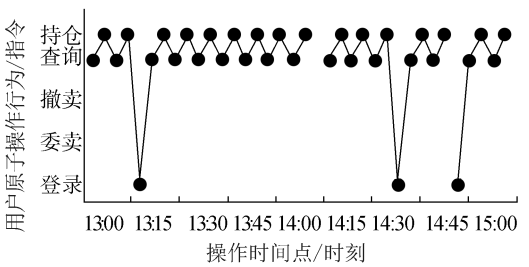


图 16 行为轨迹分析

Fig.16 Behavioral traces analysis

5.3 业务统计查询与分析

1)统计报表:如图 17 所示,统计报表是实际日志处理中的一项重要需求和应用,对计算的性能和实时性要求更高。根据具体业务功能需求,按照指定周期完成系统资源、登录、委托和业务监控等统计任务。统计报表主要是提供给系统管理员分析系统的资源使用情况和系统健康状态,以便做好相应的措施和规划,同时为管理者决策提供数据支撑和参考依据。

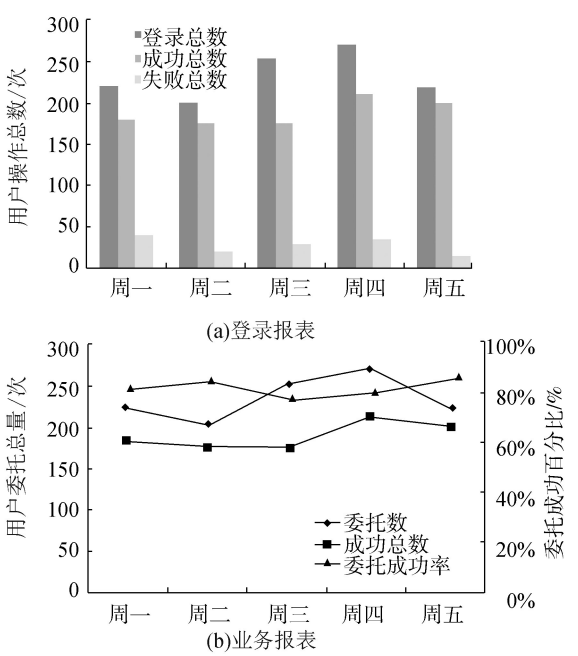


图 17 业务报表

Fig.17 Business report

2)明细查询:如图 18 所示,对网上交易日志的明细查询,实现千亿级数据秒级查询响应。主要用于实时明细查询,根据时间、系统、站点等多维条件查询从 TB 级别的日志数据中快速准确地找到所需数据,查询时效均能达到秒级响应。极大地方便了运维管理人员的工作,在节约大量的时间的同时提高了问题排查效率。

图 18　明细查询

Fig.18　Query details

6　结束语

本文研究了 SQL-on-Hadoop 技术在网络日志分析中的应用。我们选取了其中最有代表性的 3 种 SQL 查询引擎——Hive、Impala 和 Spark SQL,并使用 TPC-H 的测试基准对它们的决策支持能力进行测试及评估。构建面向证券行业的网络日志分析平台,实现万亿级日志存储和高效、灵活的查询系统,为海量日志集中分析与管理系统应用提供支持。目前 SQL-on-Hadoop 系统还存在若干问题有待解决,在有限的资源使用情况下和特定数据分布场景下提高查询处理效率等问题都有待进一步的研究。

参考文献:

[1]OLINER A, GANAPATHI A, XU W. Advances and challenges in log analysis [J]. Communications of the ACM, 2012, 55(2): 55-61.

[2]李国杰,程学旗. 大数据研究:未来科技及经济社会发展的重大战略领域——大数据的研究现状与科学思考[J]. 中国科学院院刊,2012, 27(6): 647-657.
LI Guojie, CHENG Xueqi. Research status and scientific thinking of big data[J]. Bulletin of Chinese academy of sciences, 2012, 27(6): 647-657.

[3]王元卓,靳小龙,程学旗. 网络大数据:现状与展望[J]. 计算机学报, 2013, 36(6): 1125-1138.
WANG Yuanzhuo, JIN Xiaolong, CHENG Xueqi. Network big data: present and future [J]. Chinese journal of computer, 2013, 36(6): 1125-1138.

[4]孟小峰,慈祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-149.
MENG Xiaofeng, CI Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of computer research and development, 2013, 50(1): 146-149.

[5]JOSHI S B. Apache hadoop performance-tuning methodologies and best practices[C]//Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. New York, USA, 2012: 241-242.

[6]LAMB W. The storyteller, the scribe, and a missing man: hidden influences from printed sources in the gaelic tales of duncan and neil macdonald [J]. Oral tradition, 2012, 27(1): 109-160.

[7]Apache.org. Apache Chukwa [EB/OL]. [2017-06-07]. http://chukwa.apache.org/

[8]GOODHOPE K, KOSHY J, KREPS J, et al. Building LinkedIn's real-time activity data pipeline [J]. Data engineering, 2012, 35(2): 33-45.

[9]APACHE ORG. Apache Flume [EB/OL]. [2017-06-07]. https://flume.apache.org.

[10]GHEMAWAAT S, GOBIOFF H, LEUNG S T. The Google file system[C]//Proc of the 19th ACM Symp on Operating Systems Principles. New York, USA, 2003: 29-43.

[11]THUSOO A, SARMA J S, JAIN N, et al. Hive—a petabyte scale data warehouse using Hadoop [C]// Proc of 2010 IEEE 26th International Conference. Piscataway, NJ, 2010: 996-1005.

[12]APACHE ORG. Apache HBase [EB/OL]. [2017-06-07]. https://Hbase.apache.org.

[13]APACHE ORG. Hadoop Streaming [EB/OL]. [2017-06-07]. http://hadoop. apache. org/docs/r1. 2. 1/streaming. html.

[14]WEI J, ZHAO Y, JIANG K, et al. Analysis farm: A cloud-based scalable aggregation and query platform for network log analysis [C]//International Conference on Cloud and Service Computing. Hong Kong, China, 2011: 354-359.

[15]RABKIN A, KATZ R H. Chukwa: a system for reliable large-scale log collection[C]// International Conference on Large Installation System Administration. New York , USA, 2010: 163-177.

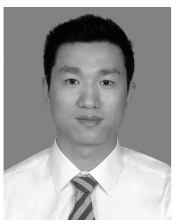
[16]LOGOTHETIS D, TREZZO C, WEBB K, et al. In-situ mapreduce for log processing [C]//Usenix Conference on Hot Topics in Cloud Computing. Berkeley, USA, 2012: 26-26.

[17]TREZZO C J. Continuous mapreduce: an architecture for large-scale in-situ data processing[J]. Dissertations and theses-gradworks, 2010, 126(7): 14.

[18]Apache.org. HDFS Architecture Guide [EB/OL]. [2017-06-07]. http://hadoop.apache.org/docs/r2.7.2/hadoop-

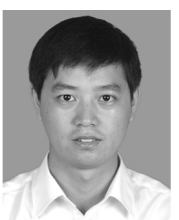
project-dist/hadoop-hdfs/HdfsUserGuide.html.
[19] DEAN J, GHEMAWAT S. Mapreduce: simplified data processing on large culsters[C]//Proc of the 6th Symp on Operating System Design and Implementation. San Francisco, USA, 2004: 137-150.
[20] HAN U G, AHN J. Dynamic load balancing method for apache flume log processing[C]//Information Science and Technology. Shenzhen, China, 2014: 83-86.
[21] Apache.org. Apache sqoop [EB/OL]. [2017-06-07]. http://sqoop.apache.org/.
[22] BITTORF M, BOBROVYTSKY T, ERICKSON CCACJ, et al. Impala: a modern, open-source SQL engine for Hadoop [C]//Proceedings of the 7th Biennial Conference on Innovative Data Systems Research. CA, USA, 2015: 4-7.
[23] FLORATOU A, MINHAS U F, OZCAN F. SQL-on-Hadoop: full circle back to shared-nothing database architectures [J]. Proc of the VLDB endowment, 2014, 7 (12): 1199-1208.
[24] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets [J]. Book of extremes, 2010, 15(1): 1765-1773.
[25] HE Y, LEE R, HUAI Y, et al. RCFile: a fast and space-efficient data placement structure in MapReduce-based warehouse systems. [C]// Proc of 27th IEEE Int Conf on Data Engineering. CA: IEEE Computer Society, 2011: 1199-1208.
[26] MELNIK S, GUBAREV A, LONG J J, et al. Dremel: interactive analysis of web-scale datasets[J]. Communications of the Acm, 2011, 3(12):114-123.

作者简介：



何明，男，1975 年生，博士，主要研究方向为大数据、推荐系统、机器学习。

常盟盟，男，1987 年生，硕士研究生，主要研究方向为数据挖掘、机器学习。



刘郭洋，男，1986 年生，硕士研究生，主要研究方向为大数据、数据挖掘。

2017 IEEE 第 2 届信息技术，电子与自动化控制会议（ITNEC2017）
2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC2017)

2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2017) will be held during December 15-17, 2017 at Chengdu China. This conference is sponsored by IEEE, IEEE Beijing Section, Chongqing Global Union Academy of Science and Technology. Submitted papers will be reviewed by at least two technical program committee members and reviewers assigned by the TPC members of the Conference. All accepted papers will be published by IEEE, which will be indexde by EI、CNKI. Excellent papers will be selected and published under SCI or EI journals.

The aim of ITNEC 2017 is to provide a platform for researchers, engineers, academics as well as industrial professionals from all over the world to present their research results and development work in Information Technology, Networking, Electronics and Automation Control. It provides opportunities for the scholars to exchange new ideas and application experiences, to extend business or research relations and to identify global partners for future collaboration.

Website: http://www.itnec.org/