计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(4):90−95 [doi: 10.15888/j.cnki.csa.006840]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Storm 的大数据指标实时计算方法①

颜　冰, 王钟雷

(中国人民财产保险股份有限公司 大数据中心, 北京 100022)
通讯作者: 颜　冰, E-mail: yanbing@picc.com.cn

摘　要: 大数据时代, 面对爆发式增长的海量异构大数据, 企业指标数据的实时供给能力亟待全面提升. 基于流处理技术的大数据指标实时计算方法, 主要由日志采集、消息管理、协调管理、实时处理等部分构成, 使用 Hadoop、Zookeeper、Storm、Kafka、Redis 等开源软件, 综合应用了数据库日志分析, 流处理、内存计算等技术. 本文详细论述了采用 Storm 技术的大数据指标实时计算方法的技术架构, 实现方法及路径, 同时给出了算法验证的过程和结果分析.

关键词: 大数据; Storm; 统计; 指标; 流处理

引用格式: 颜冰,王钟雷.基于 Storm 的大数据指标实时计算方法.计算机系统应用,2019,28(4):90–95. http://www.c-s-a.org.cn/1003-3254/6840.html

Real-Time Calculation Method of Big Data Index Based on Storm

YAN Bing, WANG Zhong-Lei

(Big Data Center, PICC Property and Casualty Company Limited, Beijing 100022, China)

Abstract: At the age of big data, faced the explosive growth of massive heterogeneous big data, the real-time supply capacity of enterprise index data needs to be improved comprehensively. The real-time computing method of big data index based on stream processing is composed of logs collection, message management, coordination management, real-time processing, and other parts. Not only open source software such as Hadoop, Zookeeper, Storm, Kafka, and Redis are used, but also the techniques of database log analysis, stream processing, and memory calculation are applied. This paper presents the discussion of the technical framework, implementation method and path of the real-time computing method of big data index based on Storm, and provides the algorithm verification process and result analysis.

Key words: big data; Storm; statistics; index; stream processing

大数据时代, 社会的各行各业, 人类的衣、食、住、行、医、娱等, 时时都在产生数据, 这些数据正呈指数级、爆炸式增长, 而且随着信息科技的不断进步, 从海量异构大数据中迅速且高效地挖掘出有效价值, 并将其转化为可靠的决策依据, 已经成为各个行业所面临的重大的挑战, 极大地考验着数据统计分析的能力 [1]. 面对规模大、种类多、变化快等大数据问题, 许多企业通过大规模的硬件资源投入来保障数据的基本处理能力, 从数据采集到生成报表, 仍然采用 *T*+1 日的数据处理机制, 规模较大的企业甚至需要 8 个小时左右才能完成, 传统技术已不能全面满足业务和管理决策的数据时效要求, 企业指标数据的实时供给能力亟待全面提升, 传统统计工作到了必须重视、研究和应用大数据技术的发展阶段.

为提高数据的统计分析处理能力, 很多企业采用了数据一体机方式作为解决方案, 如 Teradata 大数据一体机. 传统架构是主机、存储、网络、管理软件、数据仓库 (数据库或者中间件或者虚拟化软件) 等进行

① 收稿时间: 2018-10-10; 修改时间: 2018-10-30; 采用时间: 2018-11-05; csa 在线出版时间: 2019-03-28

分散管理, 而一体机则是把这些进行集成, 打包形成一体化的解决方案, 来消除传统解决方案中存在的性能瓶颈问题, 比如数据管理, I/O 读写等方面存在的性能瓶颈, 有针对性地提升系统的整体处理能力. 但是随着数据的急剧增长和大数据需求的爆发, 尤其是单一化场景需求逐步向多元化场景需求的转变, 一体机方式从总体成本、扩展能力、配套软件等方面看, 已经逐渐失去竞争优势, 尤其是对于大型 (数据密集型) 企业明显不是最优解决方案. 当前, 越来越多的企业开始探索并通过应用 Hadoop 等大数据技术来提升大数据治理和实时供给能力[2,3].

大数据的处理系统大概可以分为两类, 也就是批处理与流处理系统[4]. 批处理大数据系统 (以 Hadoop 为代表) 需先将数据“汇聚成批”, 通过批量的预处理之后, 加载到分析型数据仓库之中, 可以用来进行高性能离线“实时查询”. 这种批处理系统虽然可以对完整地大数据集合实现高效的即时查询, 但却没有办法查询到增量的实时在线数据, 存在数据延迟的问题. 相较于批处理大数据系统, 流处理是一种大数据实时处理技术的典型应用, 它是一个无限增长、没有边界的动态数据集合, 以 Spark Streaming、Storm、Flink 为代表的流处理系统无需存储大数据, 可对数据进行实时高效地在线预处理, 全面、逐条地加载到高性能的内存数据库中供查询. 本文研究的是一种使用数据库日志分析、流处理、内存计算和分布式等技术的指标实时计算模式, 即基于流处理技术的指标实时计算方法.

1 流处理技术概述

流处理系统能满足对进入系统的数据进行即时计算的需要, 相比 Hadoop、Spark 等批处理系统, 在处理方式有非常大的不同. 流处理更加像一个 MapReduce 计算的通用模型, 只不过它的响应时间可以达到秒级甚至是毫秒级. 流处理不需要对完整的数据样本进行计算, 只针对通过系统的每一个数据项进行操作. 流处理系统理论上能够处理无限多、无限大的数据, 但在同一个时间点, 却只能够处理一条 (真正的流处理) 或者少量 (微批处理) 的数据, 在不同记录之间只保持最少量的状态. 流处理属于一般意义上的数据富集、持续处理, 以及对于无界数据的分析过程的组合. 流处理模式适合有近实时处理需求的任务, 如基于网站用户行为实时产品推荐、经营指标实时计算、客户信用审核、业务审核、反欺诈等.

目前, 主要有两种不同的方法来构建流处理系统, 一种属于真正的流处理 (Native Streaming), 所有被输入的记录或者事件都将按照它们进入的先后顺序被逐个处理, 如 Storm、Flink、Samza; 另一种方式是微批处理 (Micro-Batching), 小的“批”由多条输入的记录组成, 它们按照预设好的时间常量创建, 通常是每隔几秒生成一个, 如 Spark Streaming、Trident-Storm. 五种主流的流处理技术对比情况如表 1 所示.

其中, Storm、Spark Streaming 和 Flink 是三种较常用的流处理技术[5].

表 1 五种流处理技术对比情况

项目	技术				
	Storm	Trident-strom	Spark-streaming	Samza	Flink
数据处理模式	流处理	微批处理	微批处理	流处理	流处理
Guarantees	At-least-once	Exactly-once	Exactly-once	At-least-once	Exactly-once
延时	低	中	中	低	低
吞吐量	低	中	高	高	高
项目成熟度	高	高	高	中	低

1.1 Storm[6]

Storm 比较适用于实时处理数据的场景, 它是一套开源、分布式、高容错的实时计算系统, 在多个方面具备很强的优势. Storm 具有较强的容错性, 可以对工作进程及节点的故障进行管理; 流计算可以在多线程、进程以及服务器之间并行展开, 非常易于水平扩展; Storm 的消息处理机制非常地可靠, 不会遗漏信息, 能保证每个消息都能得到完整的处理, 而且在任务失败时也能够从消息源重试这个消息; 底层使用 MQ 作为消息队列, 能够保证消息能得到快速的处理; Storm 具有可靠的事务机制, 即数据的处理完全精准, 而且可以针对高峰、低峰时间段, 动态调整实时计算程序的并行度, 以最大限度利用集群资源; 同时, Storm 的开发和单元测试也比较方便. Storm 目前发展的已经相对比

较成熟, 部署和管理起来也很简单, 性能表现也十分出众, 常常被用于实时分析、持续计算、ETL、在线机器学习、分布式远程调用等.

1.2 Spark Streaming

Spark Streaming 能够实现对具有很高的吞吐量, 需要高容错机制的流数据的实时处理, 属于 Spark 核心 API 的扩展内容之一, 支持从 Flume、Kafka、Twitter、Kinesis、ZeroMQ, 以及 TCP sockets 等多种数据源获取数据. 获取到数据源数据后, 能够利用 map、reduce、join 和 window 等高级函数, 处理特别复杂的算法, 同时也可以把处理结果持久化到数据库、文件系统、现场仪表盘等. 但是与 Storm 相比, Spark Streaming 适用于不要求实时处理和完全可靠的事务机制, 不需要动态调整并行度的场景. Spark Streaming 突出的优点是吞吐量 (即单位时间内处理的数据量, MB/S) 高, 是 Storm 的 2–5 倍. 如果除了要进行实时计算外, 还包含批处理 (离线)、交互式查询等需求, 那么就要先选择 Spark 生态, 采用 Spark Core 来实现批处理 (离线) 操作, 使用 Spark SQL 来实现交互式的查询, 再使用 Spark Streaming 来实现流计算, 将三者进行无缝地整合, 能够给系统带来非常高的扩展性能.

1.3 Flink

Flink 目前还属于新兴的项目, 仍处于不断成熟的时期. 它是介于 Spark 和 Storm 之间的一种架构, 采用了原生的流处理系统, 与 Spark Streaming 有相似的主从结构, 与 Storm 相似的数据流, 所以 Flink 兼具了低延迟和高吞吐的特性. 同时, Flink 在 API 和容错性上也有很好的表现, 使用起来相对来说也比较简单. Flink 具有许多特性: 可进行带有事件时间的窗口 (Window) 操作; 能进行高吞吐、低延迟和高性能的流处理; 窗口 (Window) 操作高度灵活; 支持 Exactly-once(有状态计算) 语义; 能进行基于 time、count、session, 以及 data-driven 的窗口操作; 对于基于轻量级的分布式快照 (Snapshot) 实现可以容错; 能运行具备 Backpressure 功能的持续流类模型; 运行时可同时支持 Batch on Streaming 处理以及 Streaming 处理; 可进行迭代计算; Flink 可在 JVM 的内部做到属于自己的内存管理; 程序可自主优化, 这样可以避免在非常特定情况下, 产生 Shuffle 及排序等高代价的操作, 当然中间结果需要进行缓存操作.

2 基于 Storm 的指标实时计算方案

基于流处理技术的指标实时计算, 是通过实时监听和捕获数据库日志, 利用流处理技术对日志和数据库操作指令进行实时解析, 并实时将分析结果用于指标计算的大数据处理模式. Flink 是新兴的项目, 而 Storm 经过多年发展已经比较成熟, 相较于 Spark Streaming 的处理延时更低, 甚至可以到毫秒级, 完全可以满足指标实时更新的需求, 因此我们选择 Storm 作为实时处理的核心技术.

Hadoop 是一个批处理系统, 它由于具备数据吞吐量大、自动容错等很多优点, 所以广泛应用在海量异构大数据的处理上. 但是, Hadoop 适合大批量数据的离线处理, 并不擅长实时计算, 因为它本来就是为批处理而开发的, 这也是大家一致的共识. 不过 Hadoop 却可以作为 Storm 等组件运行的基础框架平台. 因此, 整个实时计算系统基于 Hadoop 平台构建, 由日志采集、消息管理 (Kafka)、协调管理 (Zookeeper)、实时处理 (Storm)、内存数据库 (Redis) 等部分构成[7]. 其中, 由日志采集模块 (使用 Shell、C、Java 等脚本开发的插件) 监听数据库日志, 并实时把日志抓取下来推送至 Kafka 分布式消息管理系统; 通过 Storm 系统消费 Kafka 中的消息, 同时通过 Zookeeper 管理期间的消费记录; 由 Storm 根据指标计算逻辑对日志进行定制化分析和处理, 并输出到 Redis 内存数据库中; 最后由应用程序读取 Redis 中的结果并展示给用户, 或转入数据库进行持久化存储. 从技术架构看, 自上而下, 首先是源端数据库, 下一层是日志采集部分, 可以针对多个数据库同时进行采集, 日志采集之下是 Kafka 消息管理系统, Kafka 消息管理系统层之下是 Storm 实时处理层, Kafka 和 Storm 之间的协调管理由 Zookeeper 承担, Storm 流处理层之下是 Redis 内存数据库, 最下层是 Web 或者 App 应用, 也可以包括用于持久化的数据库 (如 Hbase 等列式数据库), 详情如图 1 所示.

2.1 日志采集

日志采集程序或工具有多种可选方式. 在此列举三种, 一种是编写 Shell、C 或 Java 脚本程序, 自编脚本轻量和完全自主可控, 对服务器产生的压力相对较小, 但需要一定的自主开发量; 第二种是采用第三方框架技术直接进行采集, 比如采用 Flume. Flume 属于分布式的、高效的日志采集系统, 可以把分布在不同服

务器上的海量日志文件统一收集到一个集中的存储资源中, 但是 Flume 的配置却不怎么简单, Source、Channel、Sink 的关系交织在配置文件中的话, 非常不便于管理; 还有一种方式是使用 CDC(Change Data Capture) 产品实现源端数据库日志的采集, 但 CDC 通常需要进行单独采购, 同时需要在源端数据库和目标端数据安装软件, 成本较高, 对服务器性能要求也高.

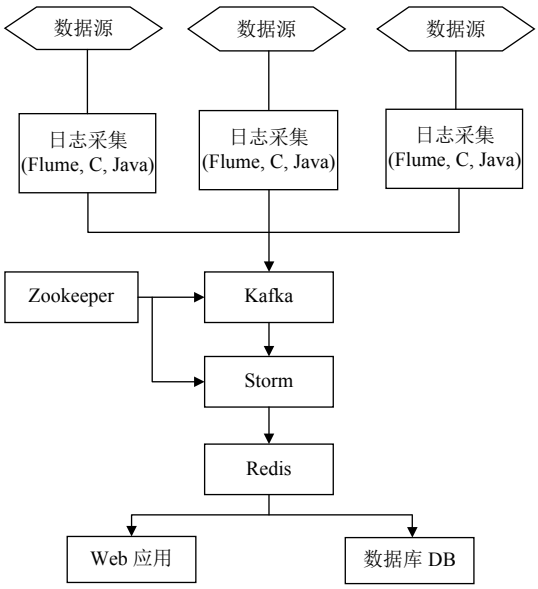

图 1 基于 Storm 的实时分析系统技术架构

本方案采用自行开发的脚本程序进行采集, 直接将采集脚本部署在源端数据库服务器, 实时监听和读取磁盘设备中的日志文件. 此方式相较于另外两种方式有三个优势: 一是部署简单, 直接部署在源端数据库服务器即可, 无需额外地搭建同步服务器, Flume 和 CDC 都需要若干台服务器来部署; 二是响应速度快, 可实现实时捕获增量日志, 而 CDC 的实时同步间隔通常是数秒, 无法满足要求; 三是只涉及磁盘文件读取, 占用源端服务器的 CPU 和内存资源少, 对服务器的运行影响不大.

2.2 消息管理 (Kafka)

Kafka 是基于日志文件的消息系统, 消息能够持久化存储到硬盘中, 数据不容易丢失. Kafka 可以保存消息的进度及位置, 对于用户来说, 也可以自行定义消费的起始点, 可以实现消息的重复和多次消费. Kafka 同时具有队列和发布订阅两种消息消费模式, 可以保证消息队列中的消息能按照顺序被消费并且与 Storm 的契合度很高. 此外, Kafka 的 Consumer 是 pull-based 模型, 该模型可以缓解日志产生速度快于消费速度的压力, 使消费速度合理匹配生产速度. 把 Kafka 消息系统放置在日志采集和 Storm 模块中间, 是防止在突发的、高并发的情况之下, 由于日志可能会出现井喷式的增长, 如果这时候 Storm 的消费速度不能快于日志的产生速度, 就会导致大量消息处理滞后, 进而导致丢失, 所以加入了 Kafka 消息系统作为数据缓冲区.

2.3 协调管理 (Zookeeper)

Zookeeper 是一个针对分布式系统的高可靠地协调系统, 它可以让分布式系统在大多数情况下正常运行. 一是可以提供分布式的锁服务[8]. 分布式集群系统中, 读取与分析等操作会分散到不同的节点之上进行, 所以在数据操作的过程中就有可能发生一致性问题. Zookeeper 提供的这种锁服务就很好地解决了此问题, 保证了进行分布式数据运算时的数据操作的一致性; 二是能够为分布式的系统提供故障恢复的支持. Storm 中 master 节点运行的守护进程“Nimbus”和 worker 节点运行的守护进程“Supervisor”之间的协调工作是通过 Zookeeper 来管理的, Nimbus 和 Supervisor 自身在集群上都是无状态的, 它们的状态都保存在 Zookeeper 中, 所以任何节点的宕机和动态扩容都不会影响整个集群的工作运行; 三是 Zookeeper 也可以管理 Kafka 的消费记录, 即使遭遇 Kafka 宕机, 在进行重启之后也能定位上次的消费记录, 从宕机点继续进行消费, 实现了“断点续传”.

2.4 实时处理 (Storm)

Storm 能够相对比较简单地实现对复杂实时计算的编写以及扩展. 数据库数据的实时处理会使用 Storm, 就像好比离线数据批处理常常使用 Hadoop 一样, 而且 Storm 能保证没有遗漏, 保证每一个消息都能被处理, 速度也比较快. 在一个相对较小的集群中, 可以使用多种语言编程, 如使用 Java、Payson 等语言进行开发, 每秒能处理百万级别的消息. Storm 作为整个指标实时计算模式的功能核心和技术核心部分, 主要完成三个方面的工作, 即日志解析、指令解析、实时计算. 该部分的日志处理能力主要受单一数据库只能采用单线程处理的限制, 实战时要注意避免该问题成为整体处理能力的提升的瓶颈, 但是不同的数据库日志可以并发处理.

2.4.1 日志解析

不同类型的数据库产品的日志编码规则和存储逻

辑各不相同, 解析日志需要首先研究数据库日志的编码和存储等规则, 这是整个计算模式能否正常运行的前提之一, 否则无法将日志解析和转换为易于识别的信息. 日志解析程序在接收到日志消息后, 将根据数据库的日志规则, 自动切分日志, 识别日志类型, 剔除回滚等不改变数据的日志类型, 仅保留增、删、改等操作产生的日志, 并将该部分日志的每一页内容由十六进制编码转换为“标准和可用”数据用于下一步进行指令解析. 整个日志解析过程可划分为捕获、切分、识别和转换等四个部分, 如图 2 所示.

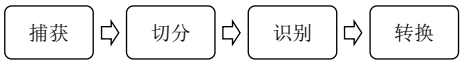

图 2 日志解析过程图

2.4.2 指令解析

基于日志解析部分的结果, 指令解析部分将按照拟统计的大数据指标的算法要求, 从日志中筛选出指标计算涉及到的所有操作信息, 解析每个增、删、改等操作涉及的指令所影响的数据表以及字段信息, 抽取出用于数据筛选和计算的信息, 尤其是相应的数据增量变化信息. 简单来说, 就是通过指令解析从中获取指标计算所需要的全部信息, 然后将解析结果推送至内存计算程序进行下一步处理. 指令解析过程可划分为识别、筛选、解析和推送等四个部分, 如图 3 所示.

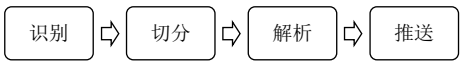

图 3 指令解析过程图

2.4.3 实时计算

以数据统计类指标计算为例. 传统的处理方式是首先同步源端数据库和目标端数据库的数据, 待增量数据在目标端数据库同步且入库完成之后, 再调用程序根据指标算法对全量样本数据进行聚合计算, 计算结果保存到数据库中等待应用读取, 数据的时效性、连续性比较差. 基于流处理的实时计算则是对数据库数据进行实时、在线、同步处理, 无需入库, 无需对样本进行全量计算, 直接在上一次计算结果基础上进行处理, 理论上可提供毫秒级的实时计算能力.

实时计算程序需要根据指标算法预设某指标的完整计算规则, 基于日志和指令解析的结果, 自动适配该指标的计算逻辑. 在筛选出需要参与计算的信息后, 针对 Insert、Update、Delete 指令选择相应的处理方式进行计算. 如果是 Insert 插入指令, 可以在上一次统计结果基础上直接进行“加法”操作; Update 更新指令在日志中会记录该表原始数据块和该表最新数据块, 可根据数据实际变化情况进行“加法或者减法”操作; Delete 删除指令进行“减法”操作. 对上一次计算得出的结果进行相应的“加法或者减法”计算后, 将得出的最新数值写入内存数据库中供应用调用. 当然, 也可以保存所有增量变化的信息, 持久化到备份库或者宽表中满足不同场景的需要.

有些指标的计算需要考虑复杂的筛选条件, 有的可能需要进行复合或混合运算, 只要获取了“字段”的变化情况, 无非是计算的复杂度得到了增加, 但是复杂的判断和计算势必会影响实时处理的效率, 这点需要在数据库建模时统筹进行考虑, 并在处理逻辑上进行优化. 此外, 需要注意的是, 内存数据库中的数据应当定期进行转储, 比如可以将转储的数据保存至 HBase 列存储数据库中, 这样就可以在系统宕机后对数据重新进行初始化.

2.5 方案优势

上述基于流处理技术的指标实时计算方案, 采用了较成熟的主流技术和工具, 能够实现对大数据的实时、在线以及持续地处理, 可以满足业务和管理决策对数据的实时性需求. 该方案与传统数据仓库、BI、数据采集 (如 CDC 等) 等技术相比, 具有五大优势. 一是处理高效, 从捕获数据库日志到完成指标实时计算的时耗能达到毫秒级; 二是对源端数据库服务器影响小, 因直接读取服务器磁盘日志文件, 不涉及数据库系统级的管理和交互, 所以基本不占用源端服务器的 CPU 和内存资源; 三是可靠性高, 消息队列 (kafka) 和协调系统 (Zookeeper) 保障了日志能够逐条被处理, 并且整个集群在宕机后能够快速恢复; 四是成本低, 流处理集群基于 X86 架构服务器搭建, 价格低, 采购、维护和升级简单; 五是可扩展性强, 整个系统采用 Hadoop 分布式集群架构, 通过增加硬件设备可实现处理能力的线性提升.

3 应用实践

3.1 流处理技术实战

基于 Storm 的大数据指标实时计算模式已经在某省级单位进行了实践. 数据库产品的型号及版本为

IBM Informix 11, 在局域网 (千兆) 内架设基于 X86 架构 PC 服务器的 Hadoop(开源) 集群作为运行平台. 日志采集使用 Java 插件完成, 采用 Hadoop、Kafka、Zookeeper、Storm、Redis 等开源产品和组件完成消息管理、协调管理、流计算、内存计算等工作. 整个实战环境不超过 10 台 PC 服务器, 可配置 4–6 个计算节点和 2–4 个管理节点, 主要用于处理本地的两个 Informix 数据库日志. 基于以上的环境和配置构建的流处理计算系统, 实现了两个大类, 不少于 3 个维度的统计指标的实时计算, 达到了毫秒级的准实时计算效果. 值得一提的是, 在每个数据库日志 200–500 MB 大小的情况下, 各台服务器 CPU 占用率仅为 5% 左右, 剩余空闲资源可利用空间还非常大, 还可以增加更多的指标进行处理, 当然也要考虑日志解析节点的综合处理能力, 不能盲目的增加过多的计算内容.

3.2 应用成果

该单位的数据发布平台对接了流处理系统的实时指标数据. 依托于流处理分布式集群的快速处理能力, 流处理系统的计算结果通过消息机制实时推送至发布平台. 以该单位的保险费指标为例, 实时数据支持日期、机构和产品三个维度, 可满足各级管理人员实时监控和分析业务情况的需要. 下一步, 该单位计划逐步开发赔款、赔案、实收保费、应收保费等更多指标的实时展示功能, 进一步增强平台的业务和管理支撑能力.

4 存在的主要问题

目前, 基于 Storm 的大数据指标实时计算方法存在两个相对比较大的问题. 一是数据库产品升级可能带来日志格式的变化. 数据库产品手册并未说明日志的编码规则, 日志的类型等信息需要自行研究, 如果数据库版本变化导致日志编码或存储规则发生变化, 这种情况就需要在升级之前重新研究日志规则, 然后对应调整日志解析算法; 二是指标口径调整可能引起系统处理逻辑的变更. 如果指标口径调整导致算法逻辑发生变化, 比如统计的字段或者数据筛选条件发生改变, 就需要调整指令解析和实时计算程序. 这两种情况尤其是第二种问题如果频繁发生, 可能要耗费大量的时间和人力成本完成相应改造工作.

5 结束语

基于 Storm 的大数据指标实时计算方法, 尚处于研究阶段, 仍需要进一步的测试和优化, 稳定性有待进一步提高, 处理能力也还有挖掘的空间. 就现阶段应用实践效果来看, 在不需要大量资金投入的情况下, 满足数据规模适中的企业的少量指标的实时计算基本没有问题, 但是数据日志规则研究和单个数据库日志的解析效率问题, 目前仍然是实现大批量指标计算的掣肘, 所以大规模应用的基础仍需要进一步的夯实. 但是随着大数据技术日新月异的不断进步, 更成熟和强大的组件或产品也会不断涌现, 该技术在将来通过持续地升级和调整, 流处理能力也必将会越来越强大.

参考文献

1 胡伟. 大数据分析浅析. 科学与财富, 2014, (5): 94, 93.

2 White T. Hadoop 权威指南: 大数据的存储与分析. 王海, 华东, 刘喻, 等译. 4 版. 北京: 清华大学出版社, 2017.

3 孙婕. 基于云计算的广域量测系统数据存储与安全方法研究[硕士学位论文]. 北京: 华北电力大学, 2016.

4 陈纯. 流式大数据实时处理技术、平台及应用. 大数据, 2017, 3(4): 1–8.

5 CSDN. 大数据流处理平台的技术选型参考. https://blog.csdn.net/uxiAD7442KMy1X86DtM3/article/details/79554077, 2018-03-14.

6 Anderson Q. Storm 实时数据处理. 卢誉声, 译. 北京: 机械工业出版社, 2014.

7 CSDN. Flume+Kafka+Storm+Redis 实时分析系统基本架构. https://blog.csdn.net/ymh198816/article/details/51998085, 2016-07-24.

8 赵玉京. 基于 Zookeeper 的分布式范围锁的设计与实现[硕士学位论文]. 武汉: 华中科技大学, 2015.