Computer Science and Application 计算机科学与应用, 2017, 7(4), 351-358
Published Online April 2017 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2017.74043

Flying Streaming: A Platform for Real Time Multidimensional Statistical Analytics of Large-Scale Log Data

Hongbo Zhao[1], Hua Qin[1], Jianbo Zhao[2]

[1]Department of Information, Beijing University of Technology, Beijing
[2]Beijing 58 Information Technology Co., Ltd., Beijing
Email: zhaohongbo@emails.bjut.edu.cn

Received: Apr. 9th, 2017; accepted: Apr. 22nd, 2017; published: Apr. 27th, 2017

Abstract

At present, it is common that daily increment of log data reaches TB level in domestic internet companies, and the real-time multidimensional statistical analysis of large-scale log data is becoming more and more important for enterprise operation, management and decision-making. However, the current large-scale log data analysis and processing technology is very professional, and business departments and operation and maintenance departments whose demand of data processing is most urgent are difficult to have such capacity. This paper designed a real-time multidimensional statistical analysis platform for large-scale log data through integrating Flume, Kafka, Storm, HBase and so on. The platform is named Flying Streaming. It solves some key technical issues, such as manifold log data access, real-time multidimensional statistical analysis, submitting, updating and deleting tasks by configuration instead of programming. Flying Streaming provides users with the ability of real-time multidimensional statistical analysis without programming. The application effect of Flying Streaming in the Internet enterprise is good, and it can meet the needs for Multidimensional Statistical Analytics of most log Data of business departments and operation and maintenance departments.

Keywords

Storm, Large-Scale Log Data, Real-Time Analytics, Multidimensional Statistical Analytics, General Platform

飞流：基于Storm的大规模日志数据实时多维统计分析平台

赵宏博[1]，秦　华[1]，赵健博[2]

[1]北京工业大学信息学部，北京

文章引用: 赵宏博, 秦华, 赵健博. 飞流: 基于Storm的大规模日志数据实时多维统计分析平台[J]. 计算机科学与应用, 2017, 7(4): 351-358. https://doi.org/10.12677/csa.2017.74043

[2]北京五八信息技术有限公司，北京
Email: zhaohongbo@emails.bjut.edu.cn

收稿日期：2017年4月9日；录用日期：2017年4月22日；发布日期：2017年4月27日

摘　要

目前国内互联网企业单日日志数据增量达到TB级已很常见，大规模日志数据实时多维统计分析对于企业运行、管理和决策越来越重要。但目前大规模日志数据分析处理技术专业性强，企业中数据处理需求最为急迫的业务部门和运维部门都难有这样的技术能力。本论文整合Flume、Kafka、Storm、HBase等开源系统设计了飞流大规模日志数据实时多维统计分析平台，解决了多种日志数据接入、实时多维度统计分析、用户通过提交配置代替大数据编程来提交、更新和删除任务等关键问题，提供了飞流平台上用户不需要编程就能方便使用的大规模日志数据实时多维统计分析的功能。飞流平台在互联网企业中实际应用效果较好，满足了业务部门和运维部门的大部分日志数据多维统计分析需求。

关键词

Storm，大规模日志数据，实时统计分析，多维统计分析，统一平台

1. 引言

互联网企业的主要数据来源是散落在各个业务服务器上的半结构化日志，比如系统日志、程序日志、访问日志、审计日志等。目前国内互联网企业单日日志数据增量达到 TB 级很常见。互联网企业间的竞争非常激烈，实时统计分析日志数据并将结果指导决策能够提高企业的竞争力。因此最原始的日志数据记录具有丰富和巨大的价值。

目前很多开源的系统，如 Flume、Kafka、Storm、HBase 等可以对日志数据进行处理，但是这些系统相互独立，均庞大复杂，需要专门的数据处理人员根据需求编程来使用，而企业的业务和运维部门一般没有专业从事大数据实时处理的人员，因此设计大数据实时处理平台，为用户提供不需要编程就能方便使用的大规模日志数据实时多维统计分析功能，是各个互联网企业的迫切需求，飞流应运而生。

统一的大规模日志数据实时多维统计分析平台需要接入多种来源的日志数据，每种日志记录了各种不同维度的运行数据，用户需要的是灵活的、方便的、多维度的统计分析。因此，飞流平台的设计目标主要是实现大规模日志数据的多源采集和聚合、实时多维度统计分析、用户可以在线通过配置代替大数据编程实现统计分析任务的热提交、热更新和热删除、统计分析结果通过 WebUI 进行展示。

2. 相关工作

目前国内外工业界和学术界在大规模日志数据分析领域做了很多研究和实践，尤其是工业界推动的开源社区非常活跃。面对数据规模大这个问题，广泛采用的解决办法是使用分布式的方案。

多源采集和聚合。Flume [1]是 Apache 基金会下面的一个分布式、可靠和高可用的，从不同源采集、

聚合和传输大量日志数据到一个集中的数据仓库的系统。文献[2]的数据表明 Flume 满足大规模数据实时处理需求。直接将 Flume 收集的数据传递给数据分析系统会带来数据在传递和分析阶段可能丢失，数据量剧增时没有缓冲等问题，所以在 Flume 和数据分析系统的中间加一层消息系统就很有必要。实时的生产日志是流式数据，Kafka [3]是 Apache 基金会下面的一个分布式的、基于发布/订阅的，主要用于流式数据的消息系统，满足大规模数据实时处理需求[4]。

实时计算。满足低延迟大规模流式数据分析的分布式计算系统主要有 Apache 基金会下的 Spark Streaming [5]和 Storm [6]，按照它们提供的编程模型就可以写出分布式的低延迟数据处理程序。Spark Streaming 的处理模型类似批处理，需要收集数据的时间，造成几秒到几分钟的延迟，只能做到准实时的数据处理。Storm 是消息流处理模型，采用服务型作业，数据流实时采集，省去了作业调度时间，满足大规模数据实时处理需求[7] [8]。日志数据分析完后，如果直接做展示，分析结果只能被利用一次。HBase [9]是 Apache 基金会下的分布式的，支持随机实时读/写访问的数据库，可用于存储大规模数据并实时存取[10] [11]。分布式计算系统将分析结果写到 HBase，数据可视化系统随机读取 HBase 存储的分析结果并以某种方式展示出来。

依靠用户大数据编程才能进行多维统计分析。北邮陈任飞等[12]设计并实现的日志数据处理系统，通过整合 Flume、Kafka、Spark Streaming 这一方法，解决了多源采集和聚合、大规模数据的处理两个问题。延迟在秒级，为准实时级别。该系统完全依靠各个部门的各个用户分别使用 Spark 的 API 进行大数据编程来进行日志数据的多维统计分析。北邮薛瑞等[13]设计并实现的日志数据处理平台，通过整合 Kafka、Spark Streaming、HBase 这一方法解决了大规模数据的处理这一个问题。延迟在秒级，为准实时级别。该平台还通过提供一些日志数据统计分析的编程接口简化了用户的编程工作，但仍需要依靠各个部门的各个用户分别进行大数据编程来进行日志数据的多维统计分析。以上两个平台或系统，都只是提供了依靠用户分别进行大数据编程来进行日志数据准实时处理的功能，没有实现实时多维度统计分析、用户可以在线通过配置实现统计分析任务的热提交、热更新和热删除、统计分析结果通过 WebUI 进行展示这些目标。

飞流通过整合 Flume、Kafka、Storm 和 HBase，并在此基础上实现了一个 Storm topology，解决了多种日志数据接入、实时多维度统计分析、用户简单配置即可提交、更新、删除统计分析任务等问题。

3. 飞流

3.1. 飞流架构

首先从整体上看下飞流的架构，如图 1。

为了支持大规模日志数据的集中多源采集和聚合，飞流架构采用 Flume 将不同源的日志数据采集、聚合和传输到一个集中的数据仓库，飞流选择 Kafka 作为这个集中的数据仓库暂时缓存数据，可以有效避免数据在传递和分析阶段丢失和数据量剧增时没有缓冲的问题。大规模日志数据的实时统计分析计算的工作以 Storm 分布式计算系统为基础。飞流的核心就是多个 Storm topology。一个 Kafka topic 对应一个类型的所有日志。一个 Storm topology 订阅一个 Kafka topic。启动一个 Storm topology，就可以承载一个 Kafka topic 对应的日志数据的所有多维统计分析任务。多个 Kafka topic，对应启动多个 Storm topology，就可以承载所有日志数据的多维统计分析任务。启动 topology 是由数据平台部门做的，对于用户是透明的。用户看到的，是一个统一的大规模日志数据实时多维统计分析平台。每个 topology 从 Kafka 拉取各自订阅的 topic 的日志数据。

用户通过 Web UI 输入任务的配置信息(配置信息包含该任务分析的日志数据属于的 Kafka topic 信息、维度信息、度量信息和度量周期 4 部分，飞流目前可配置的统计分析功能包括求和、取平均、最小、最

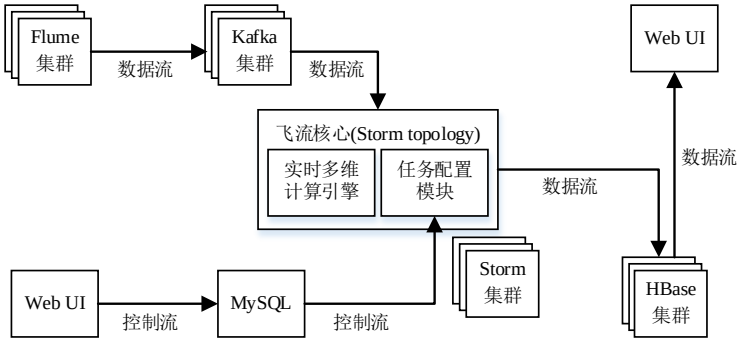


Figure 1. Architecture of Flying Streaming
图 1. 飞流架构

大、计数、唯一计数)等，此外还提供包括提交新任务、更新旧任务和删除旧任务在内的控制信息配置。用户通过 WebUI 配置的任务提交后被写到 MySQL 数据库的任务配置表中，MySQL 会自动为每一个配置信息生成一个 id，该 id 作为配置信息对应的任务的 id。

每个 topology 动态地从任务配置表中读取与自己订阅的 topic 对应的任务配置信息，按照配置好的任务需求分析日志数据，topology 将分析的结果实时持久化存储到 HBase。Web UI 动态地从 HBase 读取最新的分析结果，刷新展示结果的图表。

3.2. 关键技术

3.2.1. 多维统计分析

飞流的可配置的多维统计分析功能是多维分析理论[14]在互联网企业大规模日志数据实时处理场景中的新实践，包括数据源接入、多维数据的提取、多维聚合计算、多维聚合结果持久化等过程，其主要机制如下：

一个用户任务的配置信息中的度量信息由一个或多个<维度名，维度的值数据提取方式，黑/白名单>三元组表示(设置黑/白名单是为了解决任务只处理用户关心的该维度的所有维度值组成的集合的一个子集对应的日志数据的问题)。

假设任务 i 的配置信息中的维度信息里有 n 个<维度名，维度的值数据提取方式，黑/白名单>三元组，以第 j (1 <= j <= n)个三元组为例，维度名对应的所有维度值构成的集合记作 Aj。如果是黑名单，黑名单包含的所有维度值构成的集合记作 Cj，Cj 对于 Aj 的补集 Bj = Aj – Cj (如果是白名单，白名单包含的所有维度值构成的集合记作 Cj，记 Bj = Cj)。

笛卡尔乘积 B1 × B2 × B3 × ⋯ × Bj × ⋯ × Bn 中的元素用 d_k 表示(1 <= k <= B1 × B2 × B3 × … × Bj × … × Bn 中的元素的个数)，d_k 是一个 n 元组，表示成<v_1, v_2, …, v_j, …, v_n>。

任务 i 的数据源接入，就是从其他系统拉取待统计分析的日志数据。

任务 i 的多维数据的提取就是对于每一个到来的日志数据，从里面提取任务 i 配置的 n 个<维度名，维度的值数据提取方式，黑/白名单>三元组对应的 n 个维度的值数据。如果这些值数据都通过各自黑/白名单的筛选，这些值数据按照任务配置信息里维度的先后顺序构成的 n 元组就对应一个 d_k。

对应一个 d_k 的日志数据需要聚合到一起按照任务 i 配置信息里的度量类型和度量周期做周期性的度量计算，这是任务 i 的一个子任务。

任务 i 的多维聚合计算就是计算任务 i 的所有子任务。

任务 i 的多维聚合结果持久化比较简单，就是将多维聚合计算的结果持久化存储到 HBase 中。

这四个过程，通过编写一个 Storm topology 来具体实现。图 2 是 topology 的设计。

1) 数据源接入。这部分在 KafkaSpout 中实现。Topology 订阅哪个 Kafka topic、topology 的并行度等信息是在 topology 的配置文件里设置的，这样可以使 topology 适配各种 Kafka topic。KafkaSpout 从 Kafka partition 拉取订阅的数据，然后发到下游的 ExtractBolt，Tuple(依然是原始日志数据)从 kafkaSpout 到 ExtractBolt 之间是通过 ShuffleGrouping 策略发送的，即一个 Tuple 会被发送到一个随机选择的 ExtractBolt task。

2) 多维数据的提取。这部分在 ExtractBolt 中实现。ExtractBolt 从 MySQL 读取与该 topology 订阅的 topic 对应的任务配置信息。在 ExtractBolt 里，KafkaSpout 每传递过来一个 Tuple，配置的所有任务就会对该日志数据轮流解析，如图 3 的 a 部分。以一个任务 i 为例，如图 3 的 b 部分。

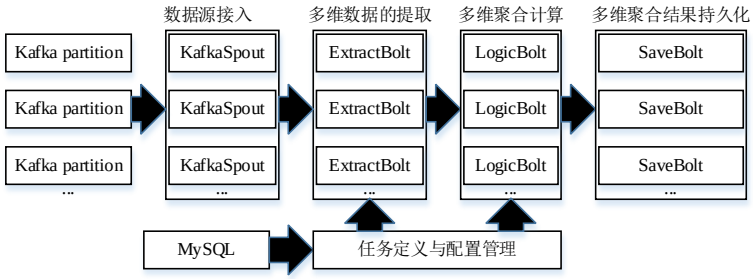
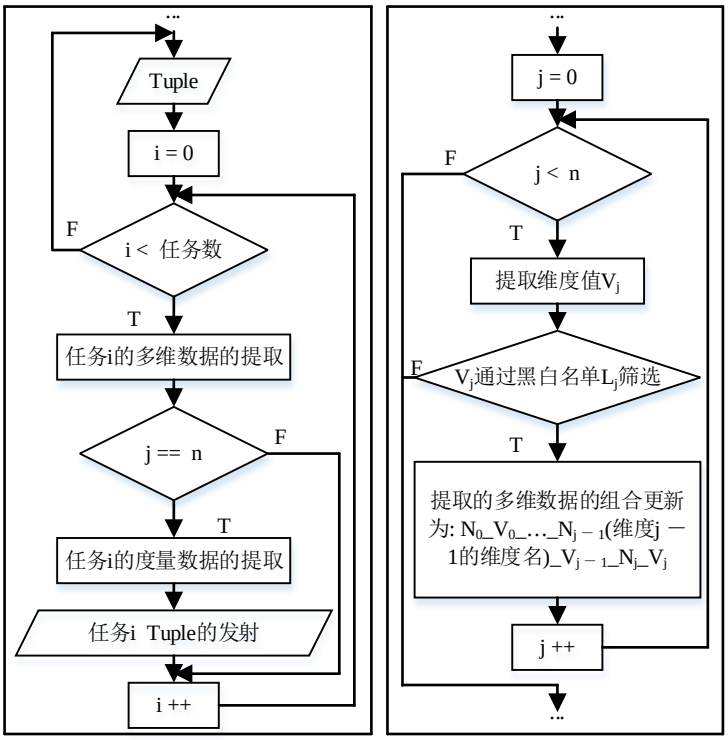


Figure 2. A storm topology
图 2. 一个 Storm topology



a. ExtractBolt的多任务　　b. 任务i的多维数据的提取

Figure 3. Extracting of multidimensional data
图 3. 多维数据的提取

对于第 j 个<维度名，维度的值数据提取方式，黑/白名单>三元组，按照维度的值数据提取方式从日志数据里提取维度的值数据，提取方式有分隔符、正则、先分隔符后正则三种。接着用黑/白名单筛选该值数据，黑/白名单可以用定值、正则或集合实现。若筛选通过，接着进行第 j + 1 个三元组对应的工作。如此，提取完任务 i 配置的 n 个维度的值数据，并且所有维度的值数据都通过黑/白名单的筛选后(这些值数据构成的重组就对应一个 d_k)，还需要根据任务 i 配置信息里的度量信息里的度量数据提取方式，从日志数据里提取度量数据，如图 3 中的 a 部分。接着构造 Tuple(任务 id_维度 1 名_维度 1 值数据_..._维度 n 名_维度 n 值数据，度量数据，度量类型，度量周期)，发射到下游 LogicBolt。Tuple 从 ExtractBolt 到 LogicBolt 是通过 FieldsGrouping 策略发送的，设置的 field 是 Tuple 中第一个字段，则任务 id 相同且维度的值数据相同的 Tuple 会发送到相同的一个 LogicBolt task 中，这样确保对应一个 dk 的所有日志数据能够聚合到一起进行接下来的多维聚合计算，即任务 i 的一个子任务的聚合计算。任务 i 的多维数据的提取完成后，进行第 i + 1 个任务的多维数据的提取。ExtractBolt 在多个 Executor 线程并行执行，每个线程处理不同的数据。

3) 多维聚合计算。这部分工作在 LogicBolt 中实现。LogicBolt 有个哈希表结构的属性，记录被该 LogicBolt 实例对应 Executor 线程处理的每个子任务的当前度量周期的度量结果和开始时间等信息。LogicBolt 收到一个 Tuple，首先解析 Tuple 里的度量类型，根据不同度量类型进入到不同算法中。每个算法的大致结构为：查看度量结果哈希表里面有没有该子任务的项，如果没有则表示这是用户提交的新任务或更新的旧任务的子任务，对 Tuple 里的度量数据做度量计算，最后用该计算结果、0 值分别作为当前度量周期的度量结果和开始时间构造表项并 put 到结果哈希表中。如果有，则读取度量结果，并结合 Tuple 里的度量数据计算本度量周期到目前为止的度量结果并 put 到度量结果表中。度量结果的表示根据度量类型的不同而不同，比如取平均度量类型，需要保存当前度量周期内的所有 Tuple 的度量数据的总和以及 Tuple 的总数。退出算法，接着处理下一个到来的 Tuple。LogicBolt 的 prepare 方法启动一个计时线程，计时线程周期性(周期的值由用户在 topology 的配置文件设置，该值就是度量周期的分度值)扫描度量结果哈希表每个表项的开始时间字段，并和当前时间比较，如发现有子任务的度量周期到点，则计时线程读取该子任务的度量结果并构造新的 Tuple(任务 id_维度 1 名_维度 1 值数据_..._维度 n 名_维度 n 值数据，度量类型，时间戳，度量结果)发送到 SaveBolt，最后将度量结果哈希表中该子任务当前度量周期的度量结果和开始时间清零，表示下一个度量周期的开始。Tuple 从 LogicBolt 到 SaveBolt 同样是通过 FieldsGrouping 策略发送的，设置的 field 是 Tuple 中第一个字段。LogicBolt 在多个 Executor 线程并行执行,每个线程处理不同子任务。

4) 多维聚合结果持久化。这部分工作在 SaveBolt 中实现。HBase 表的 rowKey 设计成“任务 id_维度 1 名_维度 1 值数据_..._维度 n 名_维度 n 值数据_度量类型_时间戳”，value 就是度量结果。SaveBolt 使用 LogicBolt 传递下来的 Tuple 信息构造 rowkey 和 value，持久化存储到 HBase 中。SaveBolt 在多个 Executor 线程并行执行，每个线程处理不同子任务。

3.2.2. 通过配置提交、更新、删除任务

设计如图 4，分为任务配置层、配置信息持久化层和分布式计算层。

1) 任务配置层。由前端用户系统实现，飞流使用 Web UI 实现，一个任务的配置信息和控制信息通过表单实时提交给后台配置信息持久化层。热提交、热更新和热删除。

2) 配置信息持久化层。由 MySQL 实现，有一张任务配置信息表，表里的一个元组表示一个任务的配置信息，有多少个元组，就有多少个任务。控制信息结合配置信息被转化成数据库的相应 INSERT、UPDATE 和 DELETE 操作，以实现插入、更新、删除一个元组。

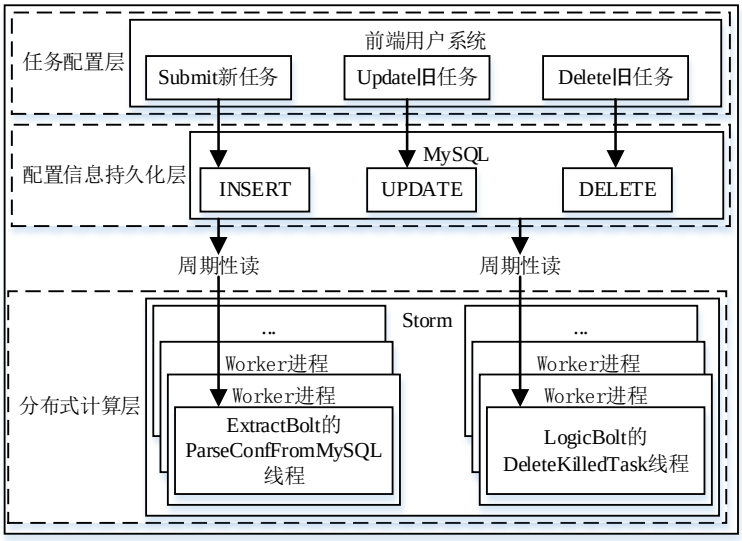


Figure 4. Hot submitting, hot updating and hot deleting
图 4. 热提交、热更新和热删除

3) 分布式计算层。该层是一个 Storm topology。topology 由多个 Worker 进程执行，每个进程可以有一个或多个 Executor 线程，一个 Executor 线程对应一个 Bolt 实例。飞流的设计里，ExtractBolt 实例对应的 Executor 线程执行数据处理之前，在 prepare 方法里启动一个 ParseConfFromMySQL 线程，ParseConfFromMySQL 线程周期性(周期的值，由用户在 topology 的配置文件里设置)读取 MySQL 的任务配置信息表，用最新的任务配置信息更新 ExtractBolt 实例中相应的任务配置信息属性，ExtractBolt 根据这些最新的配置信息进行多维数据的提取时就可以响应新任务的提交、旧任务的更新或删除。LogicBolt 对新任务的提交和旧任务的更新的响应在多维聚合计算部分已描述。LogicBolt 实例对应的 Executor 线程执行数据处理之前，在 prepare 方法里启动一个 DeleteKilledTask 线程，该线程周期性读取 MySQL 的任务配置信息表，并和度量结果哈希表比较，如果某个子任务存在于度量结果哈希表中，而在任务配置信息表中不存在该子任务对应的原任务的配置信息，则表示此原任务刚刚被用户删除，则从度量结果哈希表中删除此子任务对应的表项，这样就响应了用户旧任务的删除。这样的设计，做到了将任务的逻辑和程序本身解耦，程序不和具体的某个任务相关，而是灵活的适配各种任务。

4. 结论

本论文整合 Flume、Kafka、Storm、HBase 等开源系统设计了飞流大规模日志数据实时多维统计分析平台，解决了多种日志数据接入、实时多维度统计分析、用户简单配置即可提交、更新、删除统计分析任务等关键技术问题，提供了飞流平台上不同用户通过 web 配置界面同时提交、同时执行、同时删除不同的统计分析任务的功能，不需要用户进行大数据分析处理编程。飞流平台在互联网企业中实际应用效果较好，满足了业务部门和运维部门的大部分日志数据多维统计分析需求。未来，支持更多的度量类型和支持 SQL，还需要做一些工作。

参考文献 (References)

[1] Apache Flume (2017) Flume Homepage. http://flume.apache.org/

[2] Percy, M. (2017) Flume NG Performance Measurements. https://cwiki.apache.org/confluence/display/FLUME/Flume+NG+Performance+Measurements

[3] Apache Kafka (2017) Kafka Homepage. http://kafka.apache.org/

[4] Kreps, J. (2017) Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines). http://kafka.apache.org/performance

[5] Apache Spark Streaming (2017) Spark Streaming Homepage. http://spark.apache.org/streaming

[6] Apache Storm (2017) Storm Homepage. http://storm.apache.org/

[7] Naik, R. and Amin, S. (2017) Microbenchmarking Apache Storm 1.0 Performance. https://hortonworks.com/blog/microbenchmarking-storm-1-0-performance/

[8] 李川, 鄂海红, 宋美娜. 基于 Storm 的实时计算框架的研究与应用[J]. 软件, 2014, 35(10): 16-20.

[9] Apache HBase (2017) HBase Homepage. http://hbase.apache.org/

[10] Misty (2017) Testing HBase Performance and Scalability. https://wiki.apache.org/hadoop/Hbase/PerformanceEvaluation

[11] 张智, 龚宇. 分布式存储系统 HBase 关键技术研究[J]. 现代计算机, 2014(32): 33-37.

[12] 陈任飞, 吕玉琴, 侯宾. 基于 Flume/Kafka/Spark 的分布式日志流处理系统的设计与实现[EB/OL]. http://www.paper.edu.cn/html/releasepaper/2015/07/130/

[13] 薛瑞, 朱晓民. 基于 Spark Streaming 的实时日志处理平台设计与实现[J]. 电信工程技术与标准化, 2015(9): 55-58.

[14] 廖开际. 数据仓库与数据挖掘[M]. 北京: 北京大学出版社, 2008: 79-86.