北京理工大学

本科生毕业设计（论文）开题报告

**学 院：** 软件学院

**专 业：** 软件工程

**班 级：** 08111401

**姓 名：** 黄复贵

**指导教师：** 马锐

**校外指导教师：**

# 选题内容

论文研究课题为《基于电商日志的大数据实时流处理平台设计与实现》，旨在解决当今移动互联网高速发展的今天，如何高效率、有深度地利用并分析大数据量级电商日志的价值问题，实现电商日志数据隐藏价值的最大化利用，从而为商业决策提供有前导性意义的支持，推动社会的可持续发展[1]。

# 研究方案

## 选题主要任务

通过打造一个大数据实时流处理平台，根据已有的由一些购物网站提供的日志或自主生成的日志，模拟出实时产生的电商日志，对其中的信息进行统计分析并可视化输出，为商业决策提供依据。例如统计销售总金额、搜索热词排行、活跃城市分布等。

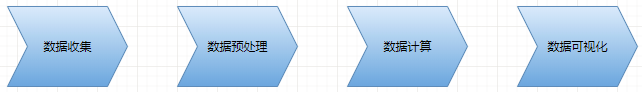
其中软件系统涉及的功能需求如下图1所示，包括：数据收集功能，搭建数据流平台，实现实时日志数据的收集；数据预处理功能，根据后续进行数据分析和统计的需要，对原始数据进行清洗和过滤；数据计算功能，对数据进行有目的性的统计计算；数据可视化功能，将统计计算后的数据以可视化的方法进行展示。

图1 功能需求图

## 技术方案分析与选择

伴随课题产生的电商日志大数据实时流处理平台[2]­，是一个带可视化界面的后台应用，整个应用部署到云服务器上。前端可视化使用阿里云开源可视化工具dataV进行构建，后台使用Go、Scala、Java三种编程语言构建。数据流处理过程为：日志源产生 -> Flume收集 -> Kafka解耦 -> Spark Streaming计算 -> Redis存储结果 -> dataV可视化展示，软件体系架构设计如下图2所示：

图2 软件系统架构图

数据流简述：实时日志源log\_generator会定时生成电商日志数据输出到文件，模拟生产环境中电商应用重要逻辑（例如搜索、购买等）的日志产生；生成的日志被分布式日志收集组件Flume收集后输出到消息队列Kafka；Spark Streaming从Kafka上消费日志数据后进行相应计算，将计算的结果输出到Redis进行存储；最后可视化组件dataV通过定制化Http Api的形式从Redis中取出计算结果数据进行显示。

在大数据的应用中，数据的采集是后续所有数据处理步骤的前提，因此，采集数据是重要的第一步。实时数据是即时数据，是由电商应用实时产生的数据，距离当前时间可能只有1小时或1分钟，大多数应用产生的日志都会追加到文件里，中间可能会有一些日志分片，因此数据的采集从文件开始。采集工具采用Cloudera提供的分布式海量日志收集组件Flume，以监听文件的方式进行采集，一旦有日志数据追加到相应文件，程序就会收集。因此，Flume组件的sources配置为exec tail，channels暂时配置为memory，sinks配置为KafkaSink。

由Flume收集到的日志数据会进入消息队列Kafka，其实Spark Streaming可以直接从Flume上消费数据，但这里还是考虑加入Kafka组件，原因有两点：

1. Kafka可以用于解耦数据处理与数据采集过程。可能后期扩展应用时数据源不一定只有Flume，这时候用Kafka解耦就能统一Spark Streaming接口的书写形式，不同数据源只需要输入到Kafka即可，做到程序的可扩展。
2. Kafka可以起到缓存数据的作用，第一可以避免因为下游Spark Streaming临时出现故障导致错失数据的情况；第二是如果遇见流量洪峰的情况下上游日志数据量太大，基于内存的Spark Streaming来不及计算时，分布式Kafka起到了容错缓存数据的重要作用。

得益于Flume、Kafka都是天生的分布式组件，因此可以做到弹性扩容提高数据收集的能力，同时也提供了在个别节点故障下程序仍然正常运行的高可用能力。

进入Kafka的数据，会被Spark Streaming拉取进行有目的性的实时计算，例如统计销售总金额、消费男女比例、搜索热词排行、活跃城市分布等。计算程序采用Scala语言编写，但在计算过程中会涉及到一些应用工程性质的需求，例如将数据落入NoSQL数据库Redis等，这时候会考虑采用RPC将数据输出给Go语言实现的服务，将数据落入数据库的步骤交给其他服务做，Spark Streaming只做纯粹的映射、过滤等计算。最终计算作业会部署在Spark集群上，理论上可以做到计算能力无限横向扩展。

计算后得到的结果，会按照一定格式存储在NoSQL数据库Redis中，方便与前端dataV可视化界面做交互。考虑到Redis是最终数据可视化的重要依赖，因此该流处理系统中Redis存储这一步需要注意以下两点：

1. Redis是基于内存的数据库，在服务器内存不足的情况下，操作系统会自动Kill掉Redis节点，这样就会造成内存中计算好的数据丢失。虽然Redis有默认的持久化机制，节点重启后会自动恢复数据，但默认的机制是基于快照.rdb的save 900 1、save 300 10、save 60 10000，即900秒之内，如果超过1个key被修改，则发起快照保存；300秒内，如果超过10个key被修改，则发起快照保存；1分钟之内，如果1万个key被修改，则发起快照保存。这种机制会丢失一定量的数据，因此要采用基于追加文件appendonly yes的持久化机制，并且配置为appendfsync always，即只要有一个计算结果的写命令，都要进行磁盘持久化，不允许数据有任何丢失。
2. 虽然Redis性能极好，一般一个节点就能处理全部读写命令。但如果Redis单点部署，就有单点故障的可能，虽然Redis也有基于Sentinel或Cluster的高可用实现，但这种机制要经过心跳检测、故障发现、Master选举、故障自动转移，通知客户端更新连接池等步骤，从故障发生，到故障转移程序自动恢复的延迟会比较大。因此，这里采用基于Supervisor的方式管理Redis节点，一旦出现节点发生故障异常退出，Supervisor就会感知并且快速重启Redis进程。

可视化dataV通过定制化Http Api方式轮询访问Redis中计算好的数据，这样就能呈现最后计算得到的结果，输出数据的价值。定制化的Http Api即通过接收前端dataV的轮询请求，从Redis中取出相应的数据按照一定的格式返回。这里又会涉及到Http Api服务的单点问题。解决方案是增加一层Nginx代理，一个节点服务请求，再热备份两个等价的节点，在服务节点挂掉的情况下，Nginx会将请求打到热备份的节点，解决单点故障的问题，保证服务高可用。

## 技术实施条件

该系统的实施需要软硬件的配合，软件需要JDK、Flume、ZooKeeper、Kafka、Spark、Redis等、硬件则需要至少3台（云）服务器的支持，具体版本如下表1所示：

表1 实施条件表

|  |  |
| --- | --- |
| 名称 | 版本 |
| >=3台（云）服务器 | CentOS 7.3 64位 |
| JDK | java version "1.8.0\_151" |
| Flume | 1.8.0 |
| ZooKeeper | 3.4.10 |
| Kafka | 1.0.0 |
| Spark | 2.2.1 |
| Redis | 4.0.8 |

## 目前主要问题

Flume、Kafka、Spark发布版本已经很多，各个版本之间的兼容性是个主要的问题。可能出现Flume已与Kafka联调好数据流，但此时Spark又不支持该版本Kafka的某种数据消费方式，因此要更换Kafka版本，此时Flume的版本有可能也要一起更换。

整个数据流的延迟问题。从日志产生源被Flume收集进入Kafka，Spark Streaming拉取消费计算，将结果落入Redis，最后dataV通过定制化的Http Api轮询获取Redis中的数据进行呈现，这中间每批数据从产生到最后可视化呈现，除去必要的计算时间，可能会额外多2~3秒的延迟时间。

Spark Streaming消费Kafka的方式问题。目前Spark Streaming消费Kafka数据有两种方式，基于Receiver与基于Direct方式[3]。采用Receiver的方式编程简单但会有性能问题，可能与zk中offset不同步的重复消费问题，还可能存在WALs机制下的数据冗余问题；采用Direct的方式消除了上述困扰，但编程复杂，需要用户自己维护offset记录，目前这两种实现方案在实际生产环境中的优劣比较，业界还不是很明确。

## 预期研究目标

实现上述设计图中的大数据实时流处理平台，包括从数据采集、预处理，到统计计算入库并最后可视化呈现的全过程。与此同时，系统设计要遵循高内聚、低耦合，具备可扩展性的软件体系结构设计原则，全链路保证高性能、高可用的数据服务。

# 课题计划进度表

表2 计划进度表

|  |  |
| --- | --- |
| 任务信息 | 时间节点 |
| 准备服务器，各个技术组件的初步使用 | 第1周 |
| 确定日志格式，实时日志产生器的开发 | 第2周 |
| 使用Flume收集产生的日志数据至Kafka，Kafka终端测试消费数据 | 第3周 |
| Spark Streaming程序从Kafka消费日志数据，计算统计前五活跃城市与地理位置，计算结果落入Redis | 第4、5周 |
| Spark Streaming程序计算统计交易总金额、消费男女比例、搜索热词排行，计算结果落入Redis | 第6、7、8周 |
| 开发所有Spark Streaming计算结果的dataV数据Api | 第9、10、11周 |
| 打包程序各模块可执行文件，编写部署命令，在服务器上整体部署运行  梳理总结、论文撰写 | 第12周  第13、14、15周 |

# 参考文献

[1]Hu H, Wen Y, Chua T S, et al. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial[J]. IEEE Access, 2017, 2(1):652-687

[2] https://www.csdn.net/article/2014-06-12/2820196-Storm

[3]http://spark.apache.org/docs/latest/streaming-kafka-0-8-integration.html