计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(10):68−75 [doi: 10.15888/j.cnki.csa.008094]

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

面向物流服务的海量日志实时流处理平台①

梁方玮, 薛　涛

(西安工程大学 计算机科学学院, 西安 710600)
通讯作者: 梁方玮, E-mail: 15029232505@163.com

摘　要: 随着电商平台的快速发展, 物流行业增长迅猛, 其中物流服务平台的访问日志能够反映用户的行为规律, 从而挖掘潜藏信息助力物流服务平台优化业务已至关重要. 目前, 针对于此类大规模日志数据处理提出了更高的实时性需求, 本文综合考量多种实时计算的流处理框架、大规模存储数据库以及日志采集工具等, 选取 Flume 及 Kafka 作为日志采集工具与消息队列, 并利用 Flink 及 HBase 进行流数据实时计算以及大规模数据存储. 同时, 对平台设计了数据去重、异常告警、容错策略以及负载调度的功能. 经实验测试证明, 本处理平台可以有效处理物流服务平台的日志数据, 具有较强的创新思路以及实际价值.

关键词: 日志处理; Flink 流处理框架; 数据实时处理; 异常告警; HBase

引用格式: 梁方玮,薛涛.面向物流服务的海量日志实时流处理平台.计算机系统应用,2021,30(10):68–75. http://www.c-s-a.org.cn/1003-3254/8094.html

Real-Time Stream Processing Platform for Massive Logs of Logistics Services

LIANG Fang-Wei, XUE Tao

(School of Computer Science, Xi'an Polytechnic University, Xi'an 710600, China)

Abstract: With the rapid development of e-commerce platforms, the logistics industry is at a high rate of growth. The access logs of the logistics service platform can reflect user behavior, so it is very important to tap the hidden information to help the logistics service platform optimize the business. At present, higher real-time requirements are imposed on large-scale log data processing. This study comprehensively considers a variety of stream processing frameworks capable of real-time computing, large-scale storage databases, log collection tools, etc. It chooses Flume and Kafka as the log collection tools and message queues and uses Flink and HBase for real-time calculation of streaming data and large-scale data storage. At the same time, the functions including data deduplication, abnormal alarms, fault tolerance strategy, and load scheduling are designed for the platform. Experimental tests have proved that this processing platform can efficiently process log data of the logistics service platform, with innovative ideas and practical value.

Key words: log processing; Flink flow processing framework; real-time data processing; malfunction alarm; HBase

随着我国经济的进步与发展, 电商平台成为人们购销的首选, 物流公共服务平台则作为信息化与效率的保障支撑着电商平台的正常运转. 为了更好地收集用户需求、降低网站的开发维护成本、提供高质量的物流服务. 日志信息毫无疑问是其提取宝藏的矿石. 因为日志全方面地记录了平台服务性能、接口调用记录以及用户行为的信息, 具有较强的追溯作用以及实际价值[1]. 因此, 对物流服务平台的日志数据进行实时采集并挖掘内在信息具有较高的现实意义与理论价值. 目前, 机器学习[2]、文本挖掘[3] 等技术已经被应用于日

① 基金项目: 陕西省 2020 年技术创新引导专项 (基金)(2020CGXNG-012)
Foundation item: Year 2020, Fund of Technical Innovation Guidance of Shaanxi Province (2020CGXNG-012)
收稿时间: 2021-01-03; 修改时间: 2021-01-29; 采用时间: 2021-02-02

志处理领域中, 并形成了较为坚实的研究基础. 例如, 实现多源异构日志的日志采集技术、针对大规模及高并发日志数据存储和并行计算技术等[4–6]. 但这些研究均持续时间较长, 不能够很好地满足实时日志分析以及异常识别等需求. 针对此需求, 相关学者提出利用低时延的流处理框架来保障大规模日志数据实时处理, 其中 Storm、Flink 以及 Spark Streaming 处理效果最为突出[7,8]. 且 Flink 相较于 Storm 以及 Spark Streaming 具有更精确的数据计算性能和完善的窗口支持特性, 更适于进行海量日志实时处理. 此外, 为了保障对海量物流服务平台日志数据进行安全存储以及高效的实时读写, 本文将引入 HBase 数据库进行数据存储, 相较于 Redis、MangoDB 以及 ElasticSearch 等具有高并发、实时处理数据以及适于存储大规模数据的特性. 因此, 本文将围绕海量日志实时处理并借助 Flink、HBase 等技术进行平台的设计与实现.

1 海量日志实时处理

大数据时代发展至今, 数据量呈指数级增长, 大数据中的日志处理也就逐渐得到重视. 目前, 针对海量日志数据的处理方式可以分为离线处理与实时处理两种类型, 其基本流程均为日志数据采集、日志数据归一化、日志数据存储、日志数据预处理、离线计算/实时查询和结果展示[9]. 但是, 随着需求方对于数据实时性要求的不断提升, 如用户的实时行为、程序实时监控告警、任务实时调度、业务综合推荐等, 实时处理成为日志数据处理领域当前亟待优化的关键性课题[10,11], 本节将从实时处理概念与流程、实时流处理框架以及实时存储数据库角度进行具体的日志数据实时处理分析.

1.1 大数据实时计算与处理

大数据的实时计算与处理可以针对性解决数据实时性的需求, 结合流处理框架以及实时存储与高并发数据库可以更好地解决面向大数据的高吞吐率和并发性需求. 相较于传统的离线批数据处理, 大数据实时计算与处理的思路可以更好地解决基于海量日志的数据处理课题. 实时计算与处理的流程主要包括数据的实时采集、数据的实时计算以及数据的实时下发. 数据的实时采集包括用户行为记录的数据采集、网站请求的数据采集、机器 CPU/MEM/IO 记录的数据采集等. 数据的实时计算是针对采集的数据进行实时计算, 计算的结果包括用户信息统计、请求数量及请求 IP、单位内机器 CPU/MEM/IO 的均值、Error 级日志数据过滤等. 数据的实时下发则是将数据交由下游进行处理, 下游包括数据信息监测告警、数据信息实时存储与检索等[12]. 实时处理流程如图 1 所示.

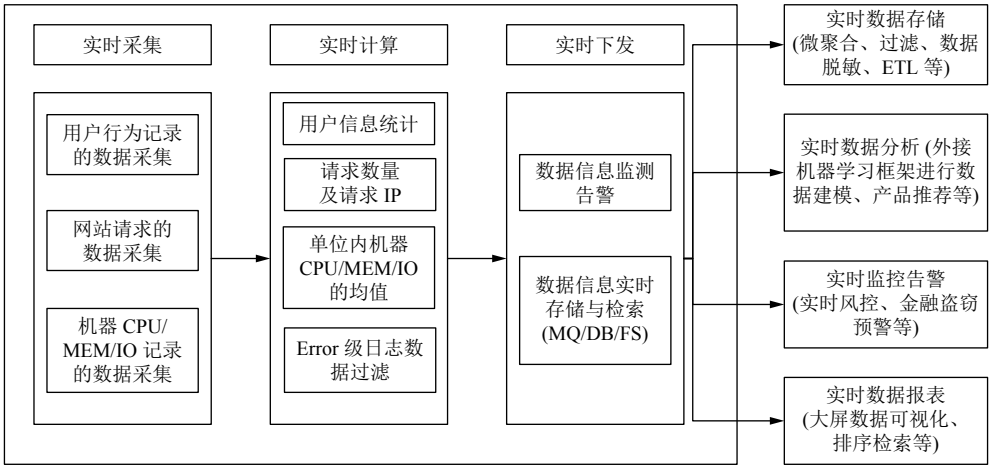

图 1 大数据实时处理流程及应用场景

1.2 Flink 流处理框架

针对大数据的实时处理, 采用的是 Flink 流处理框架进行实时计算, 相较于离线批数据处理框架可以很好地解决实时监控告警、窗口聚合、故障溯源等情况. 实时计算是不断地从 MQ (Message Queue, 消息队列) 中读入采集的数据, 并在计算后存入数据库中. 面向的是数据量未知、计算操作简单且需要实时响应的情况. 而离线计算则是从数据存储中读取固定量的数据进行

复杂的计算并生成详细报表. 所以, 基于实时计算的流处理框架 Flink 可以保障数据的实时处理、不断进行数据迭代与更新.

Flink 是针对流数据与批数据的分布式处理框架, 可以兼顾有界批数据和无界实时数据的处理. Flink 的核心为分布式运行, 主体部分包括 Program code (Flink 程序代码)、Job client (执行起点, 负责接收用户代码并创建流数据)、Job manager (作业管理器, 负责进行调度)、Task manager (接受上一级传递的 Task, 是 JVM 中的一个或多个线程中执行任务的工作节点)[13]. Flink 的工作流程如图 2 所示.

1.3 HBase 实时处理数据库

除了对于流处理框架的选择外, 平台要求数据库可以进行数据实时存储、高并发以及查询操作. HBase 数据库相较于传统的 RDBMS (关系型数据库) 等类型的数据库, 可以很好地解决实时读写、大规模数据存储以及随机访问等场景. HBase 是通过从下到上线性地增加节点来进行数据库拓展, 将大规模且稀疏的数据表构建在服务器集群上. 但是 HBase 的实时计算的特性是由其自身的架构与数据结构所决定的, 其底层架构为 LSM-Tree+HTable (region 分区) + Cache, 客户端可以直接定位到待查询数据所处的服务器位置, 并在服务器上进行数据匹配, 整个过程由 Cache 缓存完成, 这也是 HBase 虽然部署于批处理系统的集群上却可以进行实时计算的缘由. 除此上述的关键特性外, HBase 的系统架构中较为关键的组件包括: Client (客户端, 包含访问数据库的接口)、Zookeeper (数据库依赖, 对 Region 相关信息进行存储)、HMaster (进程管理, 用于管理 Region server 的负载调度) 等[14]. 其具体的工作流程图如图 3 所示.

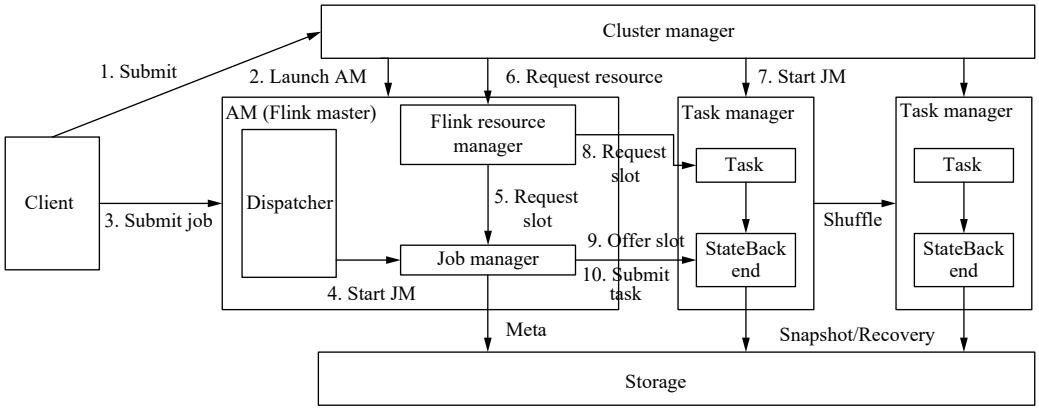

图 2 Flink 流处理框架工作流程

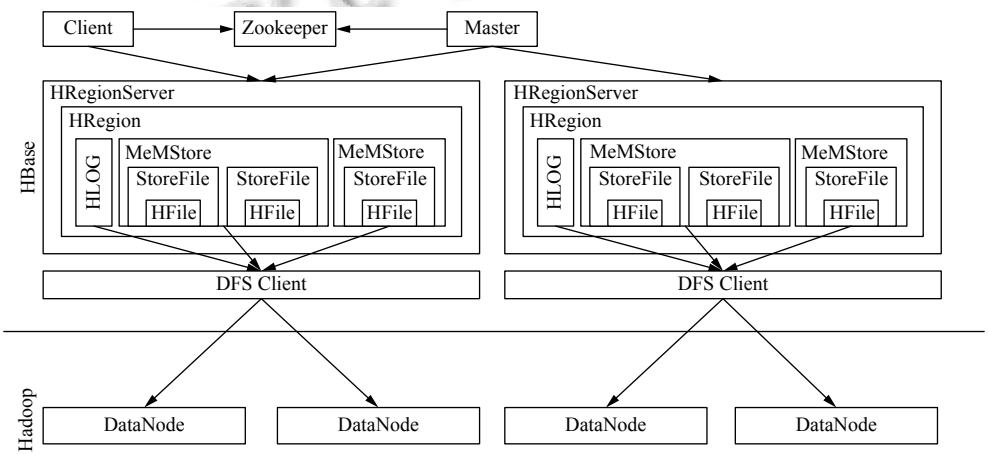

图 3 HBase 数据库工作原理

2 日志实时处理平台

当前, 访问量 10 万级的物流服务平台的日志数据量就超过 1 GB, 单机服务器已经不能满足海量日志存储的需求, 所以需要进一步引入大规模数据存储 HBase, 并且为了更好地降低日志数据信息时延的问题、提高日志的时效性, 需要进一步引入流处理框架 Flink 解决这一问题, 故本节将结合 Flink 与 HBase 以及其他相关技术对海量物流服务平台的日志数据实时处理平台的架构以及功能进行介绍, 并进一步分析架构中各部分集群的连通方式与各技术的工作原理.

2.1 日志实时处理平台架构设计

本文在传统的大数据实时处理流程中嵌入针对日志数据的相关技术, 并对传统流程的结构进行了优化使得架构更适用于海量日志数据的实时处理进程. 平台架构主要包含 5 层, 日志采集层利用 Flume 进行数据采集, Flume 是以一个典型分布式的日志采集系统[15]. 日志削峰层是利用 Kafka 对日志数据进行削峰处理. 日志处理层利用 Flink 对采集的数据进行实时处理与计算, 包括数据清洗、实时 ETL (Extract-Transform-Load) 以及实时预警. 日志存储层利用 HBase 进行数据存储. 日志展示层利用 Kibana 进行数据与计算结果的展示, 平台架构如图 4 所示.

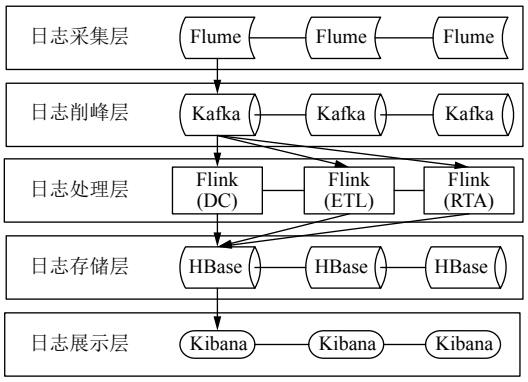

图 4 海量日志实时处理平台架构图

2.1.1 日志采集层

日志采集层将 Flume 与 Kafka 集群整合 [15], 其中, Flume 是数据的传输者, 可以源源不断地将日志数据采集到端口上, 但不会持久性地对数据进行保存, 仅以一个临时性的缓存进行保存, 并利用 sink 将数据落地传输到 Kafka 消息队列当中. 而 Kafka 作为消息队列, 除了接收来自 Flume 的日志数据外, 还需对数据进行削峰平谷.

该部分的整合思路是利用 Flume 对日志数据进行采集并将其发送到 Kafka 消息队列当中. 平台采用 Flume 集群的方式进行配置, 核心是在 Kafka 创建一个实时处理平台的 Topic 并将 Flume 采集到的日志数据发送到该 Topic 上即可.

具体的整合过程为: ① 首先设定两个 Flume agent 并部署于服务器上进行数据采集. 其次, 将日志数据以下沉的方式发送到另一个新的 Flume agent 服务器上. ② 配置完成后即可启动 Flume agent 对日志数据进行监听与传输; ③ 在 Kafka 中创建一个 Topic 来接收采集到的日志数据. 整合的原理如图 5 所示.

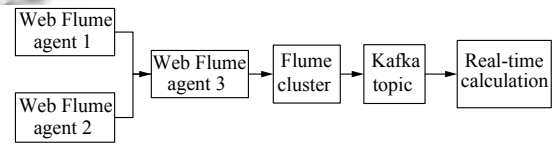

图 5 Flume 与 Kafka 整合工作流程图

2.1.2 日志处理层

日志处理层是将 Kafka 与 Flink 结合, 其原理是 Flink 对外提供了 Kafka connectors 用于读取 Kafka topic 中的日志数据, 并且 Kafka consumer 与 Flink 的 checkpoint 机制相互结合后可以保障 exactly-once 的数据处理. 为了能够进一步保证数据处理的唯一性, Flink 自身也并非完全依赖于 Kafka consumer 的跟踪模式, 而是在 Flink 内进行跟踪与检查从而保证日志数据处理的唯一性.

具体的整合过程为: ① Kafka consumer 提供了一个 (或多个) 针对 Kafka topic 的访问接口, 其构造函数对外接收相关的配置参数 (包括: Kafka topic 信息、Kafka brokers 列表、Zookeeper 服务器列表等, 以保障连通性稳定). ② Kafka consumer 建立一个到 Client 端的连接来查询 Topic 内的日志数据, 完成后启用 Flink 的 checkpoint 机制. ③ Kafka consumer 会从 Kafka topic 中的未消费的日志数据为起始, 单向周期性地扫描 Kafka 的消息偏移量以及其他操作的状态. ④ Flink 将上述的内容以流数据的形式存储到 checkpoint 当中, 并继续读取 Kafka 队列中新的数据. 整合过程如图 6 所示.

除了上述的基本集群整合外, 为了保障平台的优势性能, 还需对 Flink 进行额外的配置: ① 由于海量日志数据的采集源各不相同, 为了便于后续存储需要在

Flink job 中对日志数据进行统一化. 具体步骤是利用 grok (基于正则表达式) 解析原日志数据的 message 字段并建立新的结构来存储统一的日志数据. 统一化后需要对数据进行去重、脏数据等数据清洗, 并将其转换为 LogEvent 格式便于后续的存储. ③ 为了能够对日志数据中的异常情况进行实时告警, 还需在 Flink 框架中设定 Filter 算子进 Error 级日志过滤, 过滤后的 Error 级日志数据将封装为新的 Event (即告警信息), 再由 sink 调用下游服务进行告警. ④ 为了进一步优化 Flink 针对海量日志处理的性能, 在 Flink 中嵌入负载预测模型以及负载预测网络便于对数据流量进行预测以及资源的优先调度. ⑤ 为了能够有效保障告警机制的正确运行, 平台在 Flink 源码中设定乐观容错机制保障日志数据处理的适当容错性.

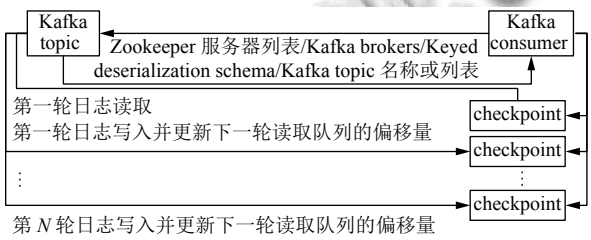

图 6 Kafka 与 Flink 整合工作流程图

2.1.3 日志存储层

日志存储层利用 HBase 进行数据存储. 除了对于日志数据的采集以及实时计算与处理外, 还需将日志数据进行存储便于后续使用, 所以该层将 Flink 与 HBase 进行交互. Flink 对此提供 Flink HBase connector 用于 Flink 从 HBase 数据库中读取或写入数据, 其中 Flink 利用 TableInputFormat 读取 HBase 中的批量数据, 以及利用 TableOutputFormat 向 HBase 中写入数据.

2.2 物流服务平台海量日志实时处理平台功能介绍

海量日志实时处理平台除了日志的采集、传输、实时计算、存储以及展示外, 并基于 Flink 设计了日志数据去重、异常检测及告警、容错机制以及负载预测等功能.

2.2.1 日志数据实时去重

实时去重模块通过选取 Flink 状态后端的 RocksDBStateBackend 进行数据集合的维护, 其状态数据均存储在 Task manager 本地机器的内存和磁盘之上, 便于对数据进行实时的操作. 为了保障大规模数据的全局去重, 针对 Flink 的状态需设置为 KeyedState, 设置完成后 Flink 将会对处理的数据通过 RocksDB 进行扫描并利用 KeyedState 进行日志数据主键的比对, 从而保障日志数据处理的实时性以及唯一性.

2.2.2 异常日志检测及实时告警

异常日志检测及告警是利用 Flink 作业去实时处理 Kafka 队列中的数据来进行异常检测的计算, 具体是利用 filter 算子进行异常日志的过滤, 核心代码为:

```
.filter(logEvent -> "error".equals(logEvent.getLevel()))
```

过滤完成后将异常日志数据构建成一个新的事件信息封装成告警内容, 在下游进行告警消息的发送, 发出的应用异常日志告警消息中会携带一个链接, 通过该链接可以跳转到对应的异常日志信息.

2.2.3 日志数据处理容错效率策略

日志数据处理容错效率策略是基于补偿函数的乐观容错机制. 相较于 Flink 系统分布式快照的悲观容错机制, 乐观容错机制无须耗费额外的时间开销去进行阶段性检查, 从而具有更高的使用效率. 同时, 本文选用 PageRank 全量迭代算法的数据收集及数据处理接口, 可以直接在 Flink 系统中调用, 该算法机制的整体流程如图 7 所示.

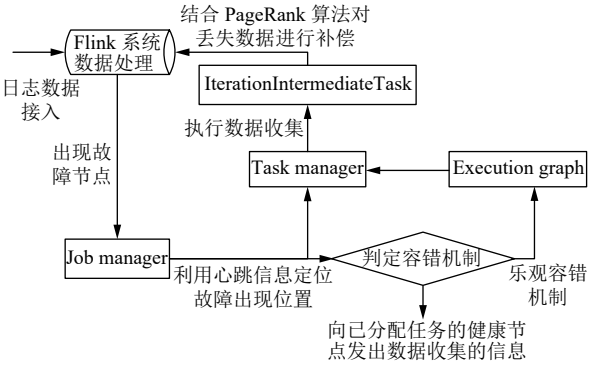

图 7 基于补偿函数的乐观容错机制工作流程图

2.2.4 实时负载预测及资源调度

为了能够有效应对实时计算负载的波动, 本文通过量化集群资源来得出本阶段内的日志数据量, 再利用负载预测模型来预测下一阶段的日志数据量变动, 最后针对下一阶段的日志数据量进行系统资源分配. 为了在 Flink 的基础上实现负载预测以及资源调度, 需要在原 Flink 架构上进行优化, 具体的优化方案为:

1) 添加负载监控节点: 用于实时计算日志负载量, 存入本地用于后续负载预测;

2) 添加负载预测节点: 执行负载预测算法接口, 结

合得到的负载序列计算下一阶段的负载波动情况;

3) 添加资源调度节点: 由 Zookeeper 集群承担, 用于存储负载预测的结果及资源调度方案, 并实时执行调度方案;

4) 添加负载迁移节点: 根据资源调度方案对节点状态进行修订, 从而完成资源在线调度.

3 实验与结果分析

3.1 实验环境

结合对于海量日志实时处理平台的整体架构, 平台的具体部署环境如表 1 所示.

平台部署完成后, 对平台的基本功能进行测试, 经过测试可以得出平台在日志数据采集、日志数据在消息队列中的传输、日志基本实时处理以及日志存储这几个方面响应正常且执行次寻准确. 除以上的基本功能外, 本文接下来将就其余创新功能进行更为详细的测试介绍.

表 1 平台部署环境表

配置项	配置参数
系统内存	32 GB×6
机器节点数量	1个主节点 6个从节点
操作系统	CentOS7
JDK	jdk1.8.0-181-Linux-X64
开发环境	Intellij IDEA
Apache Flink	1.6.0
Apache Kafka	2.10-0.8.2.0
Apache Zookeeper	3.4.10
Apache Flume	1.9.0
Apache Hadoop	2.7.4
Apache HBase	2.8.1
Kibana	7.6.1

3.2 海量日志去重效果测试

前文的功能设计中可以看出日志数据的去重是由日志数据“键”的冲突来进行重复检验的, 并利用 RocksDB 进行日志数据的本地存储, RocksDB 具体的测试参数设置如表 2 所示.

表 2 测试参数表

RocksDB参数列表名称	参数设置	备注
state.backend.rocksdb.block.cache-size	512 MB	表示用于读取日志数据的Cache容量
state.backend.rocksdb.thread.num	1	表示后台用于清洗与合并日志数据的线程数量
state.backend.rocksdb.writebuffer.size	32 MB	表示一组同时处理日志数据的writebuffer容量
state.backend.rocksdb.writebuffer.count	9000条	表示writebuffer的组数目(与日志长度有关)
state.backend.rocksdb.writebuffer.num	500条	表示合并后存入磁盘的日志数据数量
state.backend.local-recovery	8 s	表示本地回复间隔

完成基础的测试参数设定后, 利用脚本随机生成 1 万条、10 万条、100 万条简易日志数据, 分别对基于 Flink 状态后端 RocksDB 的去重框架、基于 MapReduce 与 HDFS 的去重框架、基于 Spark Streaming 与 MapReduce 的去重框架以及基于单一 HBase 的全局去重框架进行测试, 测试的结果由 (重复数据量, 响应时间) 表示, 测试结果如表 3 和表 4 所示.

从表 3 和表 4 中我们可以看出基于 Flink 状态后端 RocksDB的去重框架相较于基于 MapReduce 与 HDFS 的去重框架、基于 Spark Streaming 与 MapReduce 的去重框架具有更高的去重准确率, 且相较于直接存储于数据库进行主键对比的 HBase 具有更快的响应速度, 故通过测试结果可以看出本文基于 Flink 状态后端 RocksDB 的去重框架具有较为优良的效率.

表 3 不同框架下的数据重复量 (条)

测试量(×10^4)	MapReduce + HDFS	Spark Streaming + MapReduce	HBase	Flink-RocksDB
1	3000	3000	3000	3000
10	25 900	26 840	30 000	30 000
100	0	454 728	700 000	247 043

表 4 不同框架下去重的响应时间 (ms)

测试量(×10^{4}条)	MapReduce + HDFS	Spark Streaming + MapReduce	HBase	Flink-RocksDB
1	7.365	9.932	12.134	0.863
10	36.694	47.891	59.047	14.674
100	194.098	510.984	304.983	98.675

3.3 异常日志检测及告警效果测试

本文对 Error 级别的日志数据处理采用了在 Flink 中内嵌 filter 算子进行告警的方式, 告警的信息是通过 Flink 下游的封装后通过邮件的方式进行告警. 实验选取 Crontab 指定脚本定时执行 Error 级日志编写与发送, 实时监测邮件端的报警情况, 以其中一次的邮件报警结果为例, 如图 8 所示.

结合 100 轮次的脚本测试结果, 可以得出基于 Flink 内嵌 filter 算子的告警模式响应平均时间在 7.8 ms 且邮件发送间隔在 3 s 内, 测试结果可以满足“实时告警”的响应需求, 相较基于 Filebeat+Kafka+Storm 的日志实时处理架构的告警响应速度提升幅度接近一倍.

图 8 日志处理平台实时告警测试结果图

3.4 日志数据处理容错效率测试

本文对 Flink 底层源码进行调整后进行了容错效率的测试, 测试所采用的数据集为 Facebook 页面日志数据集 gemsec-Facebook、霍林斯大学教育网页日志数据集 Hollins 以及维基百科网页的日志数据集 Wiki-topcats, 3 个数据集均符合补偿迭代算法 PageRank 的计算范畴. 本文从正确性测试、恢复性能测试以及迭代恢复时长测试这 3 个角度对乐观容错机制及 Flink 原有的悲观容错机制进行比对, 测试的具体结果如表 5 所示. 从测试结果可以看出, 经过优化的容错机制相较于原 Flink 内嵌的悲观容错机制具有更高的恢复准确性、更短的恢复迭代次数以及更短的恢复时间.

表 5 日志数据处理容错效率测试结果表

测试内容	测试内容说明	乐观容错机制测试结果	悲观容错机制测试结果
正确性	模拟故障发生后重启作业恢复数据的正确性	各数据集模拟50次故障发生, 恢复正确率近似100%, 且节点个数越多收敛越快	各数据集模拟50次故障发生, 恢复正确率近似95%, 且节点个数越少收敛越快
恢复性能	模拟故障发生后重启作业至收敛状态的时间	各数据集模拟50次故障发生, 至收敛状态平均迭代次数为32次	各数据集模拟50次故障发生, 至收敛状态平均迭代次数为47次
迭代恢复时长	模拟故障在恢复迭代执行一半时产生从而估测恢复时间	各数据集模拟50次故障发生, 平均恢复时间为238 s	各数据集模拟50次故障发生, 平均恢复时间为387 s

3.5 实时负载预测及资源调度效果测试

本文对 Flink 底层源码进行优化后进行了负载预测及资源调度测试, 选用 1 个 Job manager 节点、6 个 Task manager 节点、3 个节点构成的 Kafka 与 Zookeeper 集群, 具体的测试参数设置如表 6 所示.

从预测算法性能以及调度响应时间这两个测评标准对优化的 Flink 调度策略、原 Flink 调度策略以及 Elastic Nephel 调度策略进行测试, 测试的环境是利用脚本编写了负载波动的日志传输情况, 具体的测试结果如图 9 所示.

从图 9 的测试结果可以看出, 优化后的 Flink 调度策略相较于原 Flink 的调度策略从算法的偏差值方面具有低的偏差值, 且在调度响应时间角度 Flink 优化调度策略具有更低的响应时间.

表 6 实时负载预测及资源调度效果测试

参数列表名称	参数设置	备注
JobManager.heap.size	2048 MB	主节点容量
TaskManager.heap.size	2048 MB	工作节点容量
TaskManager.numberOfTaskSlots	2	执行线程数
high-availability	Zookeeper	HA模式设置
state.backend	RocksDB	状态后端
TaskManager.network.memory.fraction	0.2	缓冲区大小
TaskManager.network.memory.max	500 MB	缓冲区上限
TaskManager.memory.segment-size	32 768	内存分块大小

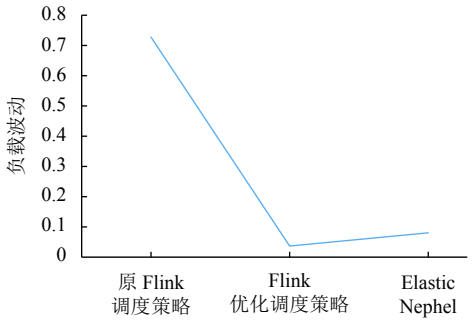

图 9 日志容错效率测试结果图

4 结论与展望

本文实现了基于 Flink+HBase 的物流服务平台海量日志数据实时处理平台. 平台除了实现基本的数据采集、消息队列传输、日志实时处理 (ETL) 以及日志实时存储外, 还在日志处理环节对 Kafka 及 Flink 的具体连接方式以及底层源码进行了优化, 并设计了大规模日志数据去重、实时告警、乐观容错机制以及弹性调度的功能, 使得平台更加完善. 相较于原 Flink 的基础框架以及功能, 平台具有更优良的性能及更快速的响应时间, 并添加了新的功能模块设计. 在下一步的研究工作中, 将会利用实际日志数据替代脚本数据进行测试, 并根据实际测试结果进行深入的优化.

参考文献

1 王玉真. 基于 Flink 的实时计算平台的设计与实现 [硕士学位论文]. 南昌: 南昌大学, 2020.
2 Lavanya K, Venkatanarayanan S, Bhoraskar AA. Real-time weather analytics: An end-to-end big data analytics service over apach spark with Kafka and long short-term memory networks. International Journal of Web Services Research, 2020, 17(4): 15–31. [doi: 10.4018/IJWSR.2020100102]
3 李钦, 杨程. 基于 ELK 的日志分析平台搭建与优化. 现代信息科技, 2019, 3(15): 193–194. [doi: 10.3969/j.issn.2096-4706.2019.15.072]
4 王帅. 基于日志的微服务化系统监测与故障预测的研究与实现 [硕士学位论文]. 成都: 西南交通大学, 2019.
5 郭文鹏, 赵宇海, 王国仁, 等. 面向 Flink 迭代计算的高效容错处理技术. 计算机学报, 2020, 43(11): 2101–2118. [doi: 10.11897/SP.J.1016.2020.02101]
6 Ye F, Liu ZH, Liu QH, *et al*. Hydrologic time series anomaly detection based on Flink. Mathematical Problems in Engineering, 2020, (1): 3187697.
7 蔡波. 云环境中用户日志采集和处理算法的研究与实现 [硕士学位论文]. 南京: 南京邮电大学, 2019.
8 李梓杨, 于炯, 卞琛, 等. 基于流网络的 Flink 平台弹性资源调度策略. 通信学报, 2019, 40(8): 85–101.
9 Fegaras L. Compile-time query optimization for big data analytics. Open Journal of Big Data, 2019, 5(1): 35–61.
10 Sahal R, Khafagy MH, Omara FA. Big data multi-query optimisation with Apache Flink. International Journal of Web Engineering and Technology, 2018, 13(1): 78–97. [doi: 10.1504/IJWET.2018.092401]
11 卜梓令, 吕明. 基于 HBase 存储的机器学习平台的研究. 工业控制计算机, 2020, 33(10): 63–64. [doi: 10.3969/j.issn.1001-182X.2020.10.023]
12 陆世鹏. 基于 Spark Streaming 的海量日志实时处理系统的设计. 电子产品可靠性与环境试验, 2017, 35(5): 71–76. [doi: 10.3969/j.issn.1672-5468.2017.05.014]
13 唐立, 李亚平, 曲金帅. 基于 HBase/Spark 的教学大数据存储及索引模型研究. 云南民族大学学报 (自然科学版), 2020, 29(5): 486–492, 507.
14 党引, 吴旻荣, 李强. 基于 HBase 的海量数据分布式序列存储策略优化. 自动化技术与应用, 2020, 39(8): 39–43. [doi: 10.3969/j.issn.1003-7241.2020.08.010]
15 Karunaratne P, Karunasekera S, Harwood A. Distributed stream clustering using micro-clusters on Apache storm. Journal of Parallel and Distributed Computing, 2017, 108: 74–84. [doi: 10.1016/j.jpdc.2016.06.004]