

# 基于 Kafka、Disruptor 技术对传统 ETL 的改进

王 梓,梁正和,吴莹莹

(河海大学 计算机与信息学院,江苏 南京 211110)

**摘 要:**ETL 系统是构建和维护数据仓库的基本构件,对异构数据源中的业务数据进行抽取、清洗、转换可通过 ETL 工具将其装载到数据仓库中。但是,当数据量上升到一定程度时,传统的 ETL 在数据处理速度以及数据的准确性方面会大大降低,并且不能满足数据源多种多样的变化需求。针对如何同时具有高效的数据处理能力和通用的数据源访问能力的问题,提出一种对传统 ETL 进行改进的方案。利用 Kafka 和 Disruptor 并发框架相结合,从数据源中抽取数据放入 Kafka 集群,结合 Disruptor 高吞吐和低延迟的特点,实现了数据高效的传输,使数据可以在不同数据源之间进行清洗和转换,同时在数据传输准确性方面有了极大的改进,保证了数据传输的一致性。

**关键词:**大数据;ETL;Kafka;数据仓库;Disruptor

**中图分类号:**TP311.133.1

**文献标识码:**A

**文章编号:**1673-629X(2018)11-0026-04

**doi:**10.3969/j.issn.1673-629X.2018.11.006

## Improvement of Traditional ETL Based on Kafka and Disruptor Technology

WANG Zi,LIANG Zheng-he,WU Ying-ying

(School of Computer and Information,Hohai University,Nanjing 211100,China)

**Abstract:**ETL system is the basic component of building and maintaining data warehouse, and business data in heterogeneous data sources can be extracted, cleaned, and transformed to be loaded into the data warehouse by ETL tools. However, when the data volume rises to a certain extent, the traditional ETL in terms of data processing speed and data accuracy will be greatly reduced, which can't meet the diversified requirements of data source. Aiming at the problem of how to have both efficient data processing and universal data source accessing, we propose an improved scheme for traditional ETL. Combined Kafka with Disruptor concurrent framework, the data is drawn from the data source into Kafka cluster. According to high throughput and low delay for the Disruptor, the efficient data transmission is achieved, enabling data to be cleaned and transformed between different data sources. At the same time, it greatly improves the accuracy of data transmission and ensures the consistency of data transmission.

**Key words:** big data; ETL; Kafka; data warehouse; Disruptor

### 0 引言

大数据时代的到来,信息已成为各行各业发展的决策依据。世界计算机数据信息总量海量增长,已经有越来越多的企业、教育、科研机构 and 生厂商们意识到这些数据中所包含的重要价值,ETL 的发展已成为必然趋势<sup>[1]</sup>。企业在使用各种应用系统期间往往积累了大量的信息、数据资源,计算机技术的快速发展使用户可以对这些信息进行采集、分析、处理,它们成为了企业发展的决策性依据,构成了企业发展的宝贵财富。随着时间的推移,技术的进步,原有旧系统不断升级从而被新系统完全取代,在此使用期间往往会积累大量

珍贵的历史数据,这些数据也是新系统成功启用的关键。同时,这些从旧系统到新系统使用过程中的历史数据也是企业进行决策发展,制定战略方向的重要依据。

如今,随着数据的大量累积,越来越多的企业、厂商都在构建数据仓库(data warehouse)来满足企业在发展过程中的战略需要,他们需要将来自不同的软硬件平台、操作系统、数据模型乃至地理上分布、管理上自治和模式上异构的数据源进行集成。一个或多个不同数据源的相关数据可以进行综合集中放入数据仓库<sup>[2]</sup>,在数据仓库中可以针对不同的主题和

收稿日期:2017-12-04

修回日期:2018-04-05

网络出版时间:2018-05-28

基金项目:国家自然科学基金(61272543)

作者简介:王 梓(1994-),女,硕士,研究生,研究方向为分布式系统;梁正和,博士,副教授,研究方向为 EPR、分布式系统。

网络出版地址:<http://kns.cnki.net/kcms/detail/61.1450.TP.20180525.1610.078.html>

汇总数据进行统计和分析,从而为决策人员提供数据支持。在构建数据仓库的过程中,首先需要将各种分布的、异构的数据源中的数据抽取出来,在此过程中进行清洗、转换、集成最后加载到数据仓库中,这个过程叫做抽取转换装载即 ETL (extraction transformation loading)<sup>[3]</sup>数据迁移。ETL 在构建过程中需要面对传输效率、准确性、数据异构性、多目标等问题。传统的 ETL 在解决不同操作系统之间使用不同的编程语言问题方面,设置一个专有的转换引擎置于数据源和目标数据仓库<sup>[4]</sup>之间,用于运行所有的转换程序。但在数据转换过程中,专有引擎执行所有转换工作成为“瓶颈”。随着企业的发展,数据需要从结构化数据源(关系数据库),非结构化数据源(PDF 文件、邮件等),半结构化数据源(XML 和其他标记语言),遗留系统(主机)、应用程序包(SAP)等异构数据源中提取,同时数据量也呈现出递增式的增长,对数据的存储<sup>[5]</sup>,数据的异构性、并发性进行研究已成为当前的主要研究方向。

在 ETL 构建过程中,尽量降低 ETL 过程的设计与维护代价,提升 ETL 过程的执行效率,是企业在实际项目中重要考虑的问题,因此,设计一种优秀的 ETL 工具对数据仓库<sup>[6]</sup>非常有益。利用 ETL 工具可以对异构数据源中的业务数据进行抽取和转换,并将其装载到数据仓库<sup>[7]</sup>,其主要作用是对各类业务数据的清理、转换和装载,为基于数据仓库的决策分析应用提供高质量的数据。截止目前,生产的数据量大大提高,传统数据处理和数据仓库技术已不能满足海量数据<sup>[8]</sup>处理的现实需求,因此基于 Kafka 和 Disruptor,提出一种对传统 ETL 进行改进的模型<sup>[9]</sup>,并就某教育企业对全国高校系统数据迁移进行了实验研究。

## 1 ETL 过程的特点

ETL 是对数据进行抽取、清洗、转换和装载<sup>[10]</sup>的过程,数据从异构数据源中抽取,迁移到指定的目标库。其间,数据的抽取、清洗、转换和装载形成串行或并行的过程。T 过程是 ETL 的核心,也是数据的转换,而抽取和装载一般可以作为转换的输入和输出,或者作为一个单独的部件,其复杂程度没有转换部件高。ETL 是构建数据仓库的重要组成部分,用户从数据源抽取所需的数据,经过数据清洗,最终按照预先定义好的数据仓库模型,将数据加载到数据仓库中。传统的 ETL 上手快,易操作,可是当数据海量增长时,传统的 ETL 在性能和数据处理的准确性、多样性、并发性等方面却大打折扣。现在数据的 ETL 过程经常会选择 Kafka 作为消息中间件应用在离线和实时的场景中。针对[万东鞠](#)撰写的[文章](#),文中基于 Kafka 和 Disruptor 并发

框架对传统 ETL 进行了改进。

### 1.1 Disruptor 特点

Disruptor 是一个高性能的异步处理框架,一般被设计在生产者—消费者(producer-consumer problem, PCP)问题上,可以获得尽量高的吞吐量(TPS)和尽量低的延迟。针对“并发、缓冲区、生产者—消费者模型、事务处理”这些元素的程序来说,Disruptor 是一种大幅提升性能(TPS)的方案。它本质上是个 ring-buffer,buffer(就是数组)做过优化防止 JVM 伪共享,lock free 是通过 CAS 自旋<sup>[11]</sup>,多线程<sup>[12]</sup>并发获取 buffer 中的序号,这里需要 CAS,把事件放入槽中,工作线程调度交给 jdk 线程池,只要 buffer 中有事件,就不停提交给线程池,不需要锁进程,解决了多线程读写,实现读写同步,解决了数据延迟的问题。

(1)disruptor 没有锁,所以效率高,速度快。

(2)所有访问者都记录自己序号的实现方式<sup>[13]</sup>,允许多个生产者与多个消费者共享相同的数据结构。

(3)在每个对象中都能跟踪序列号(ring buffer, claim Strategy,生产者和消费者),没有伪共享和非预期的竞争。

### 1.2 Kafka 特点

近年来,Kafka 作为一个新兴的分布式消息系统,受到了众多企业、科研机构的青睐。Kafka 在分布式集群应用中作为多种类型的数据管道和消息系统<sup>[14]</sup>而应用广泛。流数据是大多数集群统计和实时数据采集过程中所产生的数据,可能包括页面访问量、物联网传感器采集数据等方式。Kafka 用作 LinkedIn 的活动流(activity stream)和运营数据处理<sup>[15]</sup>管道(pipeline)的基础。在消息保存中 Kafka 根据每个 topic 进行归类,消息发送者称为 producer,接收消息者称为 consumer,此外 Kafka 集群<sup>[16]</sup>由多个 Kafka 实例组成,每个实例(server)称为 broker。

(1)持久化能力的高效性。对 TB 级以上的数据也可以在常数时间复杂度读取和写入硬盘。

(2)支持 broker 间的消息分区,并保证分区中消息读取的有序性。

(3)分布式系统,易于向外扩展。所有的 producer、broker 和 consumer 都会有多个,均为分布式的。无需停机即可扩展机器。

(4)消息被处理的状态是在 consumer 端维护,而不是由 server 端维护。当失败时能自动平衡。

(5)异步:Kafka 分布式<sup>[17]</sup>消息系统采用异步通信机制,消息进入系统缓存后系统无需立刻应答或处理,可以根据用户需求和配置情况选择。

(6)持久性、可靠性:消息被持久化到本地磁盘,并且支持数据备份,防止数据丢失。

2 改进后的 ETL 模型

基于 Kafka 和 Disruptor 数据处理技术对传统 ETL

的改进如图 1 所示。

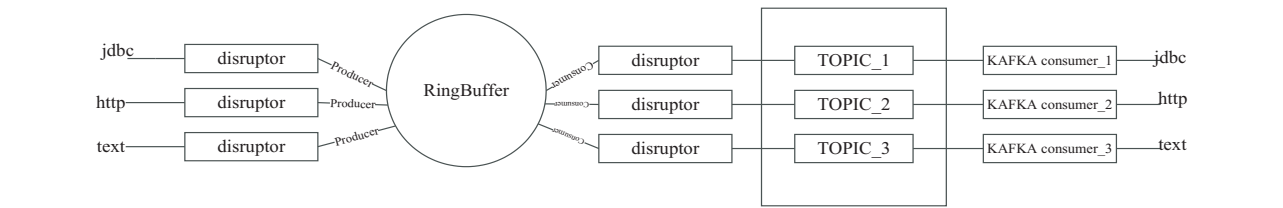


图 1 改进的 ETL 模型

实验中,Kafka 结合 Disruptor 技术,数据通道在启动后,会启动一个 restful 数据被动接收接口,前端埋点或其他的数据收集服务会将收集到的数据以 restful+json 的方式传到接口,接口在接收到数据后,会根据配置的类型,将数据发送到指定的接收源,接口源可以是配置的一个 read-type,是 Kafka 的 job,实现数据之间的快速传输。从多种数据源中抽取数据,将数据发送到 Disruptor 的 RingBuffer 环形区域,Disruptor 数据消费者发送数据到 Kafka 服务器,由于 Disruptor 高性能<sup>[18]</sup>、低延迟的特点,从而提高了目标源数据到 Kafka 中数据的速度,因此在传输速度方面对传统的 ETL 有很大的改进,极大节约了时间<sup>[19]</sup>。再利用 Kafka 高吞吐和异步的特性,Disruptor 可持续发送数据到 Kafka 中,Kafka 消费者处理数据,节省了数据等待传输时间,同时该模型可接受多种不同数据源,实现数据多样性的传输。

3 实验验证过程

启动数据通道,在通道启动一个 restful 数据被动接收接口,解析配置文件,程序初始化操作→启动消费者线程→生产者进程发布事件到 Disruptor,从目标源中读取数据到 Disruptor 的 RingBuffer 环形队列中,Disruptor 消费者将 RingBuffer 环形队列<sup>[20]</sup>收集到的数据通过 restful+json 的方式传到接口,接口在接收到数据后将数据发送到 Kafka。由于 broker 的增加或者减少都会触发 Consumer Rebalance,数据通过 Kafka Consumer 开始处理 partition 里面的 message,实验中接收源为 jdbc,数据发送到指定的接收源过程中对数据进行了清洗和装载。由于 Kafka 高吞吐、异步性的特点,可以将数据存放在 Kafka 服务器端,随时处理服务器端的数据。

Disruptor 读取源数据和发送数据给 Kafka broker 服务器,在程序中的主要流程图如图 2 所示。

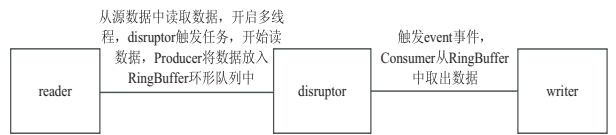


图 2 Disruptor 读写过程

Kafka 原理结构如图 3 所示。

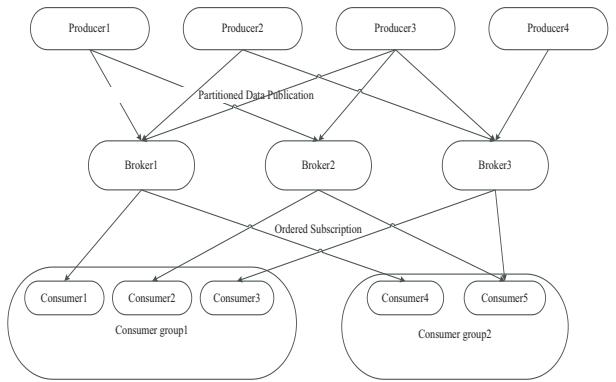


图 3 Kafka 原理

4 实验结果分析

4.1 实验结果

图 4 所示为改进后的 ETL 将 MYSQL 数据库中的数据迁移到 Postgresql 数据库中。t\_ampa\_useraction 分别在 MYSQL 不同的数据库下,在 job 配置文件中配置了目标源和目标数据库,启动项目,开始数据迁移。实验中使用了 4 张表同时进行,由开始时间和接入时间可见,极大地节约了数据传输时间。

数据库名称	开始时间	结束时间	记录条数
Mysql t_ampa_useraction	19:28:00		740 156
pg 数据库 t_ampa_useraction_new		19:34:12	740 156
Mysql t_ampa_useraction_01	19:29:00		503 015
pg 数据库 t_ampa_useraction_01_new		19:34:12	503 015
Mysql t_ampa_useraction_02	19:30:00		561 326
pg 数据库 t_ampa_useraction_02_new		19:34:12	561 326
Mysql t_ampa_useraction_03	19:31:00		1 014 000
pg 数据库 t_ampa_useraction_03_new		19:34:12	1 014 000

图 4 MYSQL 中数据导入 Postgresql

图 5 所示为传统 ETL 将 MYSQL 数据库中的数据迁移到 Postgresql 数据库中,两个 ETL 同时启用。

数据库名称	开始时间	结束时间	记录条数
Mysql t_ampa_useraction	19:28:00		740 156
pg 数据库 t_ampa_useraction_new		19:55:27	740 100
Mysql t_ampa_useraction_01	19:29:00		503 015
pg 数据库 t_ampa_useraction_01_new		19:49:03	503 015
Mysql t_ampa_useraction_02	19:30:00		561 326
pg 数据库 t_ampa_useraction_02_new		19:53:12	561 326
Mysql t_ampa_useraction_03	19:31:00		1 014 000
pg 数据库 t_ampa_useraction_03_new		20:08:22	1 011 200

图 5 传统 ETL MYSQL 中数据导入 Postgresql



4.2 实验分析

通过实验对比,现在数据的 ETL 过程经常会选择 Kafka 作为消息中间件应用在离线和实时的场景中,结合 Kafka 的特性和 Disruptor 高并发、高吞吐的特点,Disruptor 消费者发送数据到 Kafka 服务器中,实现数据的高效传输。传统的 ETL 当数据量上升到一定程度时,传输的记录有时会缺失,处理时间过长,无法实现并发存储,对此做出了改进。在实际开发过程中,因为同时对各个高校的数据进行迁移,数据量急剧增长,因此使用 Kafka 高吞吐量、低延迟、异步的特性,可以极大提高数据的传输效率。

5 结束语

文中利用 Kafka 和 Disruptor 并发框架两种数据处理技术快速构建数据 ETL 通道,凭借高吞吐量、低延迟的特点,极大节约了数据之间的传输时间。实验采用分布式消息系统作为大规模流数据的缓存,提高了平台对动态流数据输入数据量突发性变化的适应能力。针对多种数据源如 http、txt、jdbc 等的处理,对传统 ETL 进行了改进,实现对大量数据的并发处理,使不同数据库之间的数据能够快速、同步、多样地传输。

虽然对传统 ETL 在处理速度和吞吐量方面进行了改进,但是在排序、分页等功能上做得还不够完善,当 job 上升到 10 个以上时,xml 文件解析便容易出错,对这些问题将有待进一步完善。

参考文献:

[1] 钟华,冯文澜,谭红星,等.面向数据集成的 ETL 系统设计与实现[J]. 计算机科学,2004,31(9):87-89.

[2] 宋旭东,闫晓岚,刘晓冰,等.数据仓库 ETL 元模型设计[J]. 计算机仿真,2010,27(9):109-111.

[3] 吴远红. ETL 执行过程的优化研究[J]. 计算机科学,2007,34(1):81-83.

[4] 郭志懋,周傲英. 数据质量和数据清洗研究综述[J]. 软件

学报,2002,13(11):2076-2082.

[5] 张宁,贾自艳,史忠植. 数据仓库中 ETL 技术的研究[J]. 计算机工程与应用,2002,38(24):213-216.

[6] 庞金香. 浅谈高校的数据清洗与整合[J]. 计算机时代,2017(8):39-42.

[7] 赵龙. 数据仓库与 CRM[J]. CAD/CAM 与制造业信息化,2002(10):25-28.

[8] 蒋海波. 海量数据存储系统的高可靠性关键技术研究与应用[D]. 成都:电子科技大学,2013.

[9] HAN Jiawei, KAMBER M. 数据挖掘概念与技术[M]. 范明,孟小峰,译. 北京:机械工业出版社,2001.

[10] LABIO W, YANG J, CUI Y, et al. Performance issues in incremental warehouse maintenance[C]//Proceedings of the 26th international conference on very large data bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000:461-472.

[11] 毛斐巧,齐德昱,林伟伟. 一种解决构件连接死锁问题的方法[J]. 软件学报,2008,19(10):2527-2538.

[12] 谭玉靖. 基于 ZooKeeper 的分布式处理框架的研究与实现[D]. 北京:北京邮电大学,2014.

[13] 赵革科,常炳国. 一种面向服务的异步消息中间件的设计[J]. 计算机应用,2009,29(8):2312-2314.

[14] 徐高潮. 分布计算系统[M]. 北京:高等教育出版社,2004.

[15] 鲍玉斌,孙焕良,冷芳玲,等. 数据仓库环境下以用户为中心的数据清洗过程模型[J]. 计算机科学,2004,31(5):52-55.

[16] 项凯. 面向海量高并发数据库中间件的研究与应用[D]. 上海:上海交通大学,2015.

[17] 张勇. 基于 Java 多线程同步的安全性研究[J]. 河北工程大学学报:自然科学版,2011,28(2):105-108.

[18] 周庆岳. Java 集合框架的线程安全[J]. 厦门科技,2006(1):54-56.

[19] 夏魏,邵清. ETL 在超市大数据量中的应用研究[J]. 信息技术,2013,37(11):117-120.

[20] 张鑫. 深入云计算: Hadoop 源代码分析[M]. 北京:中国铁道出版社,2013:63-70.

(上接第 25 页)

2015,319:289-313.

[12] DEVENEMA Y. Lectures on the modal  $\mu$ -calculus[R]. Beijing: Renmin University in Beijing, 2008.

[13] WINSKEL G. Event structure semantics for CCS and related languages[C]//Proceedings of the 9th colloquium on automata, languages and programming. Berlin: Springer, 1982:561-576.

[14] TARSKI A. A lattice-theoretical fixpoint theorem and its applications[J]. Pacific Journal of Mathematics, 1955, 5(2): 285-309.

[15] WILKE T. Alternating tree automata, parity games, and modal  $\mu$ -calculus[J]. Bulletin of the Belgian Mathematical Socie-

ty, 1993, 23(2): 359-391.

[16] 刘光武. 自动机状态复杂度及模型研究[D]. 武汉: 华中科技大学, 2007.

[17] WANG Zhe, WANG Kewen. Uniform interpolation for  $\mathcal{AL}\mathcal{C}$  revisited[C]//Australasian joint conference on artificial intelligence. Berlin: Springer, 2009:528-537.

[18] MAKSIMOVA L L. The lyndon property and uniform interpolation over the Grzegorzczuk logic[J]. Siberian Mathematical Journal, 2014, 55(1): 118-124.

[19] KOOPMANN P, SCHMIDT R A. Count and forget: uniform interpolation of SHQ-ontologies[C]//International joint conference on automated reasoning. Berlin: Springer, 2014: 434-444.