计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(9):93−97 [doi: 10.15888/j.cnki.csa.005968]

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

# 数控机床传感器数据分析中 ETL 系统改进①

刘　峰[1], 蔡明高[1,2], 于　波[1], 于碧辉[1]

[1](中国科学院 沈阳计算技术研究所, 沈阳 110168)
[2](中国科学院大学, 北京 100049)

**摘　要**: ETL(Extract-Transform-Load)系统是数控机床传感器数据分析中不可或缺的一个重要组成部分. 针对近年来数据量越来越大和实时性要求越来越高的问题, 本文对 ETL 系统中传统的变更数据捕获方案做了改进, 设计了适用于该数据的实时数据抽取方案, 并详述了整个分布式架构的设计. 实验表明, 该 ETL 系统在数控机床传感器数据的实时处理上, 具有较高的效率.

**关键词**: 数控机床; 传感器; ETL; 实时

引用格式: 刘峰,蔡明高,于波,于碧辉.数控机床传感器数据分析中 ETL 系统改进.计算机系统应用,2017,26(9):93–97. http://www.c-s-a.org.cn/1003-3254/5968.html

## Improvement of ETL System in NC Machine Sensor Data Analysis

LIU Feng[1], CAI Ming-Gao[1,2], YU Bo[1], YU Bi-Hui[1]

[1](Shenyang Institute of Computer Technology, Chinese Academy of Sciences, Shenyang 110168, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

**Abstract**: ETL(Extract-Transform-Load) is an indispensable component of the NC machine sensor data analysis. In view of the increasing amount of data in recent years, and the rising requirements for real-time data processing, this paper improves the traditional change data capture methods, and designs real-time data capture solutions based on the characteristics of those data. What's more, it details the entire ETL distributed architecture design. Finally, experiments show that the ETL system on NC machine sensor data real-time processing has higher efficiency.

**Key words**: NC machine; sensor; ETL; real-time

随着传感器技术的发展, 可以越来越准确的对工业器件的工作数据进行实时采集. 这些数据中蕴含着巨大的价值, 通过挖掘分析可以更好的改进生产流程、提高生产效率、减少产品缺陷、提高产品质量、节约资源成本. 如何通过各种技术手段, 把数据转化为信息、知识是数据分析中的首要一步.

数据仓库就是为了解决上述问题应运而生的产物. 数据仓库是一个面向主题的(Subject Oriented)、集成的(Integrate)、相对稳定的(Non-Volatile)、反映历史变化(Time Variant)的数据集合, 用于支持管理决策, 其目标是将分散异构的大量数据信息从多种数据源中分离, 转换为统一集成的信息并进行存储和分析. ETL(Extract-Transform-Load)是抽取、转换、加载的英文缩写, 是业务集成过程与数据仓库的核心. 根据国内外众多实践得到的共识, ETL 的工作量约占整个数据仓库工作量的 60%~80%, 因此是数据仓库中非常重要的且不可或缺的一环.

近年来, 数控机床生产过程中传感器数据量越来越大以及对实时性要求越来越高, 传统 ETL 对数据的处理已经力不从心. 基于此, 本文提出了一种改进的 ETL 系统. 其中主要结合数控机床传感器数据的特点设计了基于触发器的分表结构的变更数据捕获的解决

① 收稿时间: 2016-12-27; 采用时间: 2017-01-23

方案, 并结合该方案提出了一种高效的实时抽取方案.

## 1 基于 Hadoop 的分布式架构的设计

对数控机床数据进行数据挖掘、分析, 首先要做的就是对大量的历史数据(冷数据)进行数据建模, 然后将传感器实时传递的数据(热数据)导入到建好的模型中进行数据分析. 这两步对整个系统的实时性要求不同: 用于建模的历史数据对实时性要求较低, 注重的是结果; 而传感器传递的数据对实时性要求很高, 可能需要根据器件的实时数据来对其磨损程度以及寿命进行一个实时的预测. 基于以上特点, 本文设计了如图 1 所示的一种架构方案.

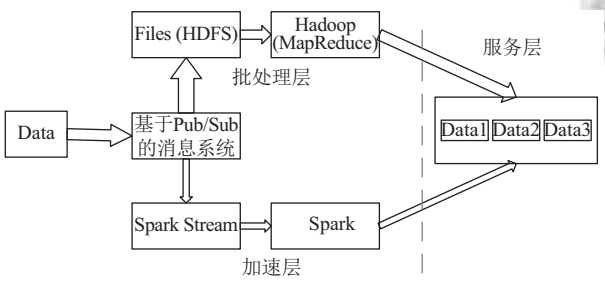


图 1 系统总体框架图

该架构主要分为三层(批处理层、加速层、服务层)和一个实时消息系统.

批处理层. 该层主要用来批量处理历史数据, 以便后续对历史数据进行建模. 由于数据量巨大, 且对实时性要求不高. 本层采用基于 HDFS 的分布式存储方式, 基于 MapReduce 大数据计算模型对抽取的数据进行批处理, 最后加载到业务分析系统中.

该层首先将所有数据全部加载到 HDFS 文件系统中, 在 Hadoop 之上搭建一层 Hive 数据仓库, 通过 Hive 的 HQL 语句来调用 MapReduce, 对数据进行分布式计算处理.

加速层. 该层是后续业务层模型已经建立, 主要是对一些实时的数据进行处理, 该层对数据的实时性要求比较高. 本层采用 Spark Stream 数据流的方式对数据进行抽取、转换、加载.

服务层. 该层主要是后续的业务层, ETL 将处理过的数据加载到该业务层, 进行后续的数据分析、数据挖掘.

实时消息系统. 该系统主要用于数据源和批处理层、加速层之间进行消息传递, 会在后面详细介绍.

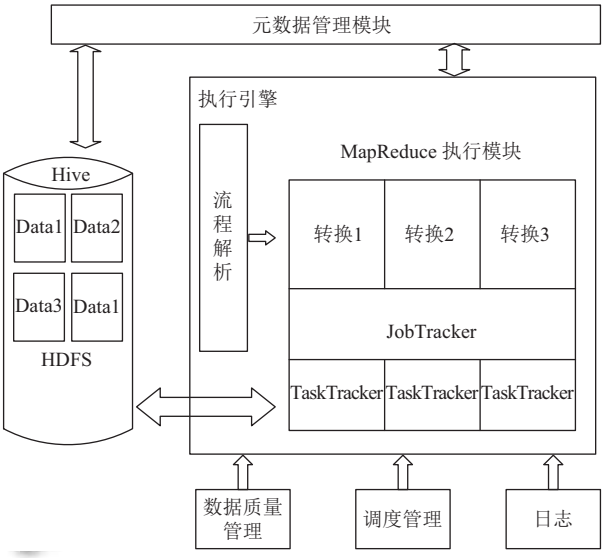


图 2 批处理层

## 2 变更数据捕获的解决方案

要加速数据抽取处理效率, 首先要解决的就是变更数据的捕获, 不仅要求能够准确快速的获得变化的数据, 而且不能对原有的系统造成太大的压力, 影响原有系统. 下面首先介绍几种传统的变更数据捕获的方法:

### 2.1 基于时间戳

时间戳方式是指 ETL 系统在抽取数据时, 通过比较数据源表中的时间戳字段, 来决定捕获哪些数据. 这种方式需要在数据源表中添加一个时间戳字段, 数据发生改变时, 相应的时间戳字段发生改变. 目前有的数据库支持时间戳字段的自动更新, 但是有的数据库就不支持. 对于不支持的情况就需要在数据发生改变时进行手动添加, 这就需要更改原始数据的采集程序, 对原系统影响较大.

### 2.2 基于日志分析

绝大部分的关系型数据系统都有一个记录数据操作的日志文件, ETL 系统通过分析该日志文件, 获取插入、删除、更新的数据, 从而达到变更数据捕获的目的. 但是这种方法对于不存在这种日志记录的数据库是不可用的.

### 2.3 基于全表对比

这种方式是在数据抽取过程中, 将原数据与目标数据库中的数据逐条对比, 将新增、更新、删除的记录写入目标数据库中. 这种方法适用于数据量小的情况下, 数据量大的时候, 这对系统的压力过大, 并且达不到实时性的要求.

### 2.4 基于全表删除插入

顾名思义, 这种方法就是在每次 ETL 数据抽取的时候, 都先将目标数据表清空, 然后将源数据表内所有数据加载到 ETL 系统中. 对于数据量不是很大, 执行代价小于增量抽取的情况下, 我们可以采用这种方式.

### 2.5 基于触发器方式

#### 2.5.1 传统触发器方式

传统的触发器方式是在数据源表上建立插入、修改、删除 3 个触发器, 当数据源表内的数据发生变化时, 触发器将变化的数据写入一个临时的变化表 Temp 表中, ETL 从该变化表中抽取相应的数据, 并从变化表中删除抽取过的数据.

通过测试, 这种方式对于数据量不大的情况下效率会比前面几种方式高. 但是, 数据量一旦过大, Temp 表的压力会非常大, 而且会出现数据库崩溃的现象.

#### 2.5.2 改进的触发器方式

由于传统触发器方式会存在 Temp 表的压力过大的情况, 因此, 我们对 Temp 表进行了改进, 做了分表处理. 根据数据量的大小建立 N 张 Temp 分表: Temp1、Temp2、……TempN. 为了保证负载均衡, 我们这里没有根据插入、修改、删除等操作信息来进行分表, 而是引入一个随机数, 通过 Hash 得到分表的分表号, 进而得到需要写入的 Temp 表.

下面是相关的操作代码:

(1) 创建临时变化表 TempN

```
create table TempN(
    //主键(自增)
    id int primary key not null auto_increment,
    //所在表的表名
    t_name varchar(255) not null,
    //增量类型, I 表示插入, D 表示删除, U 表示更新
    type varchar(1) not null,
//增量所在表的主键值
p_key int not null,
//操作时间
op_time date not null
);
```

(2) 对每一张需要监听的表创建对应的触发器

```
create or replace trigger T before insert or update or delete on T for each row
declare op_type varchar(1);
declare rand_num int;
declare N int;
declare temp_name varchar(30);
set N=10;
begin
    set ran_num=RAND()*N*1000%N;
    if inserting then set op_type='I';
    elseif updating then set op_type='U';
    elseif deleting then set op_type='D';
    endif;
    //Hash 之后得到相应的表名字
    set temp_name=concat('Temp', cast(ran_num as varchar(5)) );
    //记录相关数据到 TempN 表中
    insert into temp_name(t_name, type, p_key, op_time) values('Table 1', op_type, old.id, sysdate);
end;
```

## 3 实时抽取的解决方案

变更数据捕获完成后, 下一步要做的就是将变更数据实时的抽取到我们的 ETL 系统中去.

常用的实时抽取方案是对 ETL 系统设定一定的轮询时间, 每隔特定的时间就对源数据库进行一次抽取. 这样做法的好处是系统设计简单, 但是也存在很明显的缺点. 对于数据生成速度不定的情况下: 如果数据生成速度很快, 那么本次抽取、处理的压力就会很大, 很有可能下次轮询时本次处理还没有结束; 如果数据生成速度很慢, 那么两次抽取之间的空窗期就会很大, 造成不必要的性能浪费.

另外数控机床传感器数据分析中对数据的真实性要求很高, 所以我们一般抽取来自传感器的真实数据, 因此原数据库中很少存在 update、delete 等操作, 而大量存在的是 insert.

基于以上两点分析考虑, 本论文基于 Redis 的 Pub/Sub(消息发布/订阅)模式, 提出了一种直观且高效的实时抽取解决方案.

如图 3 所示, 首先, 我们开启数据库独立的表空间, 然后, 在消息系统中发布一个话题“有新数据到达”, 分布式 ETL 系统去订阅这个话题. 在业务系统中实时监控增量日志表的大小, 一但其大小改变达到设定的阈

值, 则向话题中发布通知, 通过消息推送, 分布式 ETL 系统接收到该消息, 故根据增量日志表中的记录在业务数据库中开始数据抽取.

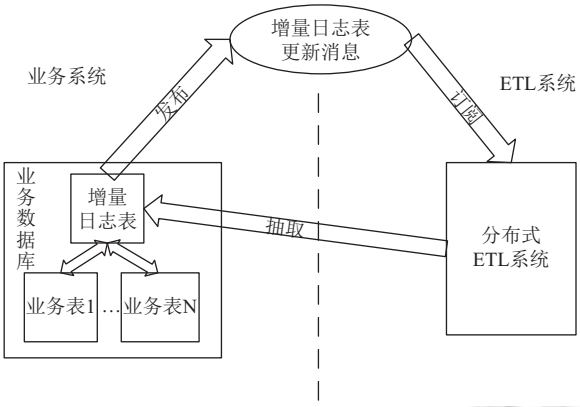


图 3 实时抽取解决方案

## 4 实验分析与总结

### 4.1 测试方案

基于以上的设计思想, 我们对该 ETL 做了实例验证.

数控机床某部件轴承传感器实时将轴承数据写入业务系统数据库, 监控组件对 Temp 表进行实时监控, 并通过消息系统实时通知 ETL, ETL 实时抽取数据进行清洗、转换等操作, 最后将处理完成的数据写入服务层数据库中.

### 4.2 测试环境

源数据库: SQL Server2008;

目标数据库: Hbase1.2.1;

测试数据: 数控机床轴承数据;

网络: 100 Mbps 局域网;

ETL 系统集群: Ubuntu 14.04.

### 4.3 测试数据结果

数据延迟的影响因素主要有以下几个方面: 其一, 对监控数据表大小设定的阈值, 临时表到达阈值之后触发 ETL 系统进行抽取, 因而不同阈值会对数据的延迟产生影响. 其二, 是临时表的个数, 不同个数的临时表对源数据库的压力也不同, 数据拥堵情况不同, 进而造成的延迟也不同. 其三, ETL 流程对数据的处理时间.

1) 阈值

设定临时表个数为 1, 控制新数据的插入速度为匀速, 需要处理的数据条数为 0 条(排除干扰), 记录系统吞吐量随新插入的数据条数的变化.

处理数据数量小于 10000 条时, 系统吞吐量逐渐增大; 10000 条左右时系统吞吐量达到最大; 而数据量大于 10000 条之后, 系统吞吐量逐渐降低.

数据量达到 10000 条之后, 系统吞吐量下降, 分析原因有如下几个方面: 其一, ETL 组件对数据的处理速度有限, 数据量增大之后, 组件间交互的开销增大, 导致数据延迟增大; 其二, 实验中只用到了一张 Temp 表, 数据量增大之后, 该表的压力增大, 数据库性能下降, 数据延迟增大.

2) 临时表个数

设定源数据库新插入的数据为 50000 条(即阈值设定为 50000), 并且控制新数据的插入速度匀速, 需要处理的数据条数为 0(排除干扰), 记录系统吞吐量随 Temp 表的个数的变化.

从图 5 中可以看出, 在 Temp 表个数为 3 时系统吞吐量达到最大, 但是仍小于图 4 中 10000 条时的吞吐量, 这时由于 Temp 表数量增多, ETL 系统时数据交互增多导致的; 另外, Temp 表个数大于 3 个时, 随着表个数增多系统吞吐量逐渐减小, 且有加速的趋势.

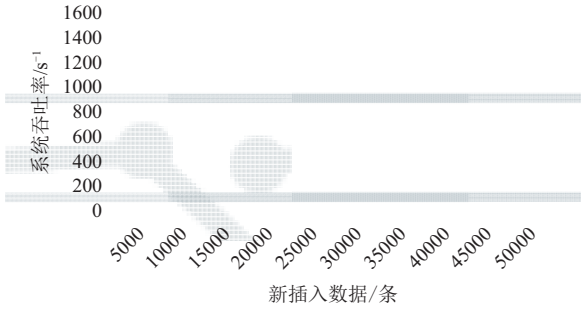


图 4 系统吞吐量与阈值关系图

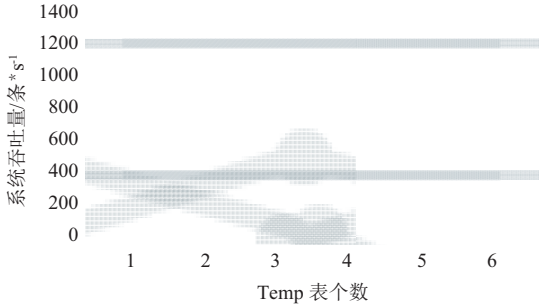


图 5 系统吞吐量与 Temp 表个数关系图

3) 待处理数据数量

设定临时表个数为 1, 控制新数据的插入速度为匀

速, 源数据库新插入的数据为 10000 条, 记录系统吞吐量和从第一条数据进入到源数据库到最后一条数据插入到目标数据库所用的时间.

由图 6 可以看出, 待处理数据条数对系统吞吐量的影响不大. 且由图 7 可以看出, 当阈值设为 1000 条, Temp 表个数设为 1 时, 延迟时间不超过 8 s.

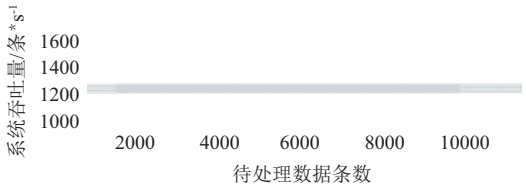


图 6 系统吞吐量与待处理数据条数关系图


图 7 系统处理时间与待处理数据条数关系图

实验表明, 该 ETL 系统在数控机床传感器数据分析中有较高的效率, 可以做到很好的实时性.

## 参考文献

1 何刚. 基于 Hadoop 平台的分布式 ETL 研究与实现[硕士学位论文]. 上海: 东华大学, 2014.
2 宋杰, 郝文宁, 陈刚, 等. 基于 MapReduce 的分布式 ETL 体系结构研究. 计算机科学, 2013, 40(6): 152–154.
3 张亮. 分布式数据仓库中 ETL 技术的研究[硕士学位论文]. 沈阳: 沈阳航空工业学院, 2009.
4 刘胜. 基于增量 ETL 的分布式数据交换平台的研究与实现[硕士学位论文]. 北京: 国防科学技术大学, 2011.
5 林建昌. 电力行业分布式 ETL 数据集成系统研究与实现[硕士学位论文]. 成都: 电子科技大学, 2015.
6 马瑞新, 许力. 基于 SOA 的实时 ETL 的研究与实现. 计算机工程与科学, 2007, 29(8): 115–117.
7 段成, 王增平, 吴克河. 一种轻量级电网实时数据 ETL 系统的设计与实现. 电力系统保护与控制, 2010, 38(18): 174–177, 182. [doi: 10.7667/j.issn.1674-3415.2010.18.034]
8 Kumar Hota CPP, Ramu Y, Subba Rao DBV. A relative study on traditional ETL and ETL with apache hadoop. NCRTIT2K16. Andhra Pradesh, India. 2016. 74–78.
9 Awad MMI, Abdullah MS. A framework for interoperable distributed ETL components based on SOA. Proc. of the 2nd International Conference on Software Technology and Engineering. San Juan, PR, USA. 2010. V1-67–V1-70.
10 Kakish K, Kraft TA. ETL evolution for real-time data warehousing. Proc. of the Conference on Information Systems Applied Research. New Orleans Louisiana. 2012.
11 Liu XF, Thomsen C, Pedersen TB. ETLMR: A highly scalable dimensional ETL framework based on mapReduce. Proc. 2011 International Conference on Data Warehousing and Knowledge Discovery. Berlin Heidelberg. 2011. 196–3111.
12 章水鑫, 徐宏炳, 于立. 增量式 ETL 工具的研究与实现. 现代计算机, 2005, (3): 6–10.
13 廖飒. 数据仓库增量 ETL 的设计及优化. 贵阳学院学报(自然科学版), 2008, 3(3): 5–9.