基于 Spark 的交互式数据预处理系统①

张 磊[1,2], 朱 锋[2], 钟 华[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院软件研究所 软件工程技术研究开发中心, 北京 100190)

摘 要: 高质量的决策依赖于高质量的数据, 数据预处理是数据挖掘至关重要的环节. 传统的数据预处理系统并不能很好的适用于大数据环境, 企业现阶段主要使用 Hadoop/Hive 对海量数据进行预处理, 但普遍存在耗时长、效率低、无交互等问题. 提出了一种基于 Spark 的交互式数据预处理系统, 系统提供一套通用的数据预处理组件, 并支持组件的扩展, 数据以电子表格的形式展现, 系统记录用户的处理过程并支持撤销重做. 本文从数据模型、数据预处理操作、交互式执行引擎以及交互式前端四个方面描述了系统架构. 最后使用医疗脑卒中的真实数据对系统进行验证, 实验结果表明, 系统能够在大数据场景下满足交互式处理需求.

关键词: 数据预处理; Spark; 交互式; 大数据

Interactive Data Preprocessing System Based on Spark

ZHANG Lei[1,2], ZHU Feng[2], ZHONG Hua[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The high quality decision-making depends on high quality data, hence data preprocessing is an essential phase for data analytics applications. In the big data area, traditional data preprocessing systems cannot be directly applied. To handle the large-scale data, enterprises adopt Hadoop/Hive as a popular solution at the present stage. However, it brings many defects, such as poor performance, the lack of interaction and so on. To fill this gap, this paper proposes and implements an interactive data preprocessing system based on Spark. This system provides a series of common preprocessing logics as basic components and supports flexible user-defined extensions. To get an interactive interface, the system presents data to users in the form of spreadsheets, while it can automatically records users operations to provide undo and redo support. In this paper, we introduce the architecture of this system with four aspects: data model, data preprocessing operations, interactive execution engine and interactive GUI. In the end, we conduct experiments with real stroke data and the result shows that the system can meet interactive demands in most big data scenarios.

Key words: data preprocessing; Spark; interactive; big data

1 引言

随着信息化的发展, 企业内部积累了大量的数据, 这些数据对于企业来说是至关重要的. 然而, 现实世界中往往存在很多数据是脏的、不完整的、不一致的, 这些数据可能导致操作代价昂贵、决策制定失败甚至错误. 为了解决这个问题, 数据预处理受到国内外学者的广泛关注, 数据预处理[1]有多种方法: 数据清理、数据集成、数据转换和数据规约等. 这些数据预处理技术, 为后续数据的分析挖掘提供了重要的保障, 同时也提高了数据分析挖掘的性能.

传统的数据预处理技术并不能很好的适用于大数据环境. 在海量数据场景下, 一个数据预处理过程要对一张拥有千万行记录的数据表进行信息提取, 往往要花费数小时的时间, 从而整个系统的效率将变得极

① 基金项目:国家自然科学基金(U1435220)
收稿时间:2016-03-09;收到修改稿时间:2016-04-08 [doi:10.15888/j.cnki.csa.005453]

其低下. 面对大数据所带来的挑战, Hadoop 技术已然成为大数据处理的标准, 很多大型企业都采用 Hadoop 及其相关产品 Hive、Pig 等解决数据预处理问题. 虽然 Hadoop 能够有效的处理数据, 但缺点明显, Hadoop 基于 MapReduce[2]的批处理方式, 处理过程缓慢, 对于数据预处理这类随时需要反复尝试的操作会让分析人员难以忍受, 这也是一系列实时处理框架, 如 Storm、Spark 等兴起的原因.

Spark[3]是 UC Berkeley Amp Lab 开发的继 Hadoop 之后的新一代大数据分布式处理框架, Spark 基于内存计算, 通过将计算的中间结果数据持久地存储在内存中减少磁盘 I/O, 非常适合迭代式算法与交互式数据分析. Spark 提供了丰富的操作类型, 不像 Hadoop 仅仅只有 Map 与 Reduce 两种操作, 这使数据处理变得灵活简洁. Spark 一经推出便受到开源社区的广泛关注和好评, 并逐步形成完整的生态圈. 如 Spark Streaming 的实时处理、Spark SQL 的即席查询、BlinkDB 的权衡查询、MLlib 的机器学习库、GraphX 的图处理以及来自 SparkR 的统计分析等. 很多大数据应用纷纷转向 Spark, Spark 也成为了 Apache 的顶级开源项目.

基于 Hadoop 实现的数据预处理过程依赖于作业的运行, 缺乏交互性, 这些操作是批处理方式进行的, 整个过程没有任何反馈, 任何的偏差都可能导致整个步骤的重做. 为了解决这个问题, 本文探讨基于 Spark 技术实现交互式数据预处理. 本文的系统充分利用了 Spark 的诸多优点, 建立了基于 Spark 的交互式执行引擎, 并提供了常用的数据预处理组件. 系统还提供了电子表格形式的交互前端, 用户的每个操作结果都能很直观的显示, 并可随时撤销重做, 从而增加了数据预处理的效率.

本文第 2 节介绍了相关工作, 描述了数据预处理的国内外现状; 第 3 节问题描述介绍了系统针对的问题以及解决的难点; 第 4 节则从四个关键点介绍了基于 Spark 的交互式数据预处理系统架构; 在第 4 节的基础上, 第 5 节描述了系统的具体实现; 第 6 节结合真实医疗数据对系统进行验证; 第 7 节则对本文的工作以及未来的工作方向进行了总结.

2 相关工作

数据预处理一直都是国内外学者关注的课题[4-7], 许多学者提出了一些数据清理框架. 文献[8]与文献[9]实现了一个可扩展的数据清洗工具, 工具划分为规范层与物理实现层, 用户在逻辑规范层设计处理流程后在物理层则实现并优化. 文献[10]提出了一个关于数据清理的交互式系统框架, 该框架将数据转换和差错检测紧密的结合在一起. 可以看出, 可扩展与交互式一直是学者在数据预处理方面致力研究的两个方向.

传统的数据预处理技术并不能很好的应用在大数据领域, 往往采取的解决方案有两种, 一是升级现有服务器, 但是这样往往不能根本解决问题, 因为升级服务器的花费巨大, 而且效果并不好; 二是是分布式系统的应用, 分布式系统的优点可以利用现有的服务器资源, 并且分布式系统可以动态的扩展. 很多的大型企业都选用分布式系统来解决这个问题, 比如 Google、Facebook 等公司的数据处理系统都是用这种方法. 企业实现现阶段实现数据预处理一般采用 Hadoop 以及基于 Hadoop 的数据仓库工具 Hive 和大规模数据分析平台 Pig. 这些工具能够很好的解决大数据预处理问题, 但缺点也很明显, 基于 MapReduce 架构的 Hadoop 性能低下, 无法达到交互式标准, 提交作业后需要耗费大量时间等待处理结果, 如果中间某个步骤发生错误, 则需要修改代码并重新运行作业, 这对于用户来说是无法忍受的.

值得一提的是 Datameer[11]这个工具. Datameer 是一个交互式数据可视化分析工具, 拥有用户友好的界面, 被称之为“Hadoop+电子表格”. Datameer 操作简洁、功能强大, 在同类产品中广受欢迎, 销量剧增, 可见交互式在大数据领域的价值.

3 问题描述

本节通过实际场景中的例子来阐述交互式大数据预处理的需求[12].表格 1 是一个记录了员工基本信息的关系数据表, 其中的数据存在很多质量问题, 下面列出这些常见问题, t_i 表示第 i 行数据.

表 1 员工信息表

No.	name	sex	zipcode	city	salary	rate
1	Annie	F	10001	NY	24000	15
2	Annie	F	10001	NY	24000	15
3	Laure		90210	LA	25000	10
4	Mark	M	90210	SF	88000	28
5	Rob	M	6082700	CH	15000	15
6	Mary	F	90210	LA	81000	28

3.1 数据质量

表格中存在几个以下几个质量问题: 重复值(t_1与t_2是重复的), 缺失值(t_3中 sex 字段缺失), 奇异点(t_5的 zipcode 非法)等.

3.2 逻辑错误

数据可能存在逻辑错误, 这些错误需要用户定义规则检测.比如$\Phi_F:D(zipcode \rightarrow city)$, 表示由 zipcode 可以推断出 city, 表中t_3,t_4,t_6存在冲突.又如$\Phi_D:\forall t_1,t_2 \in D,\neg(t_1.rate>t_2.rate \Lambda t_1.salary<t_2.salary)$表示数据应该满足等级高的员工工资应该更高, 则$t_1,t_3$存在错误.

3.3 数据格式

数据挖掘往往需要对数据进行一系列转换, 比如: 特征选择(仅考虑 salary 与 rate 字段依据员工等级预测其工资), 数据规范化(对 salary 进行归一化), 属性扩展(增加新的维度)等.

数据分析需要解决这些数据问题, 比如对表 1 数据分析前进行以下数据预处理操作: ①重复值检测, 去除重复数据; ②缺失值检测, 除去 sex 为空的数据; ③数据筛选, 依据用户规则去除不符合的数据; ④特征选择, 选择所需字段; ⑤属性扩展, 增加新的维度.

现实中的数据往往非常复杂, 形式结构各异.很多数据预处理操作需要对全局数据进行分析, 如果数据量巨大, 传统单机的数据预处理系统并不能处理这些数据, 所以需要一个分布式的数据预处理平台, 提供一系列常用的数据预处理操作, 并支持用户扩展.

使用 Hadoop 技术可以实现这个目标, 但存在几个问题: ①代码偏向底层且难以复用, 分析人员需要手动编写复杂的 Map 和 Reduce 逻辑来完成预处理, 因为未提取共有的操作, 彼此很难复用, 开发维护代价大; ②批处理, Hadoop 的批处理方式导致运行过程没有反馈, 某个步骤出错无法撤销并需要重新运行; ③运行时间, Hadoop 作业需要启动、中间数据序列化到磁盘等原因, 将花费大量时间, 用户只能耐心等待并无法观测数据状态, 如果出错还需更多时间.

通过以上分析, 一个通用的数据预处理平台很有必要, 这个平台需要支持多种数据预处理操作并支持扩展.同时, 交互式也具有举足轻重的意义, 只有每个操作的结果实时展现给用户, 用户才能够了解数据的状态, 高效率的实现数据预处理.

4 基于Spark的数据预处理系统

一个通用的数据预处理系统架构如图 1 所示, 可以看出一个完整的数据预处理系统应满足以下需求:

支持多种数据源. 数据源的多异性一直都是数据处理系统需要解决的问题, 系统需要对这些数据源进行统一处理, 并提供接口标准一遍扩展新的数据源.

数据预处理操作. 数据预处理包含了很多具体操作, 这些操作可能是一些简单的行列变化、数据筛选, 也可能是复杂的数据清理算法. 系统需要提供一套标准数据预处理组件模型, 并支持扩展.

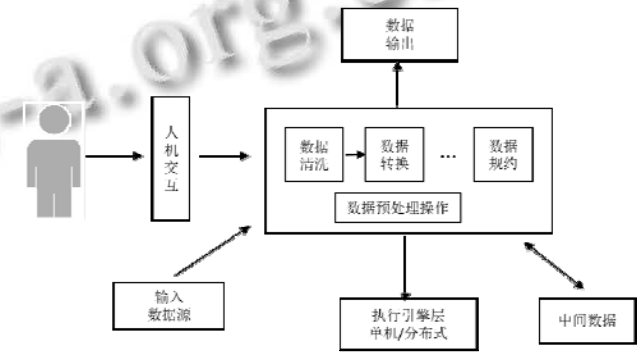

图 1　通用数据预处理系统架构图

通用执行引擎. 数据预处理是很多个操作的集合, 这些操作之间是相互独立的, 并且数据是共享传递的. 系统需要提供这些操作依赖的执行环境, 常见的有单机与分布式执行引擎.

交互式接口. 系统应该提供一个人机交互界面, 用户能够方便的进行数据预处理操作.

本节接下来将从这四个方面介绍如何基于 Spark 实现一个数据预处理系统.

4.1 数据模型

现实中数据源是复杂多变的, 系统需要对多种数据源提供支持, 比如关系型数据(Oracle、MySQL 等), 二进制, 文本(CSV、JSON 等), XML, 各种大数据平台的数据模型(Hive、HBase 等). 系统基于 Spark RDD/DataFrame 建立数据模型, 各种数据只需要通过相关数据源输入组件转化为 RDD/DataFrame 后就可使用内置或者扩展 API 实现各类数据操作.

RDD(Resilient Distributed Datasets, 弹性分布式数据集)是 Spark 的基本数据模型, 是一个容错的、只读的、并行的数据结构, 可以让用户显式地将数据存储到磁盘和内存, 并能控制数据分区. DataFrame 是

Spark 推出的一个用于大规模数据科学的 API, DataFrame 使得 Spark 具备了处理大规模结构化数据的能力, 在比原有的 RDD 转化方式易用的前提下, 计算性能更还快了两倍. 基于 DataFrame 可以联结所有主流数据源并自动转化为可并行处理格式. DataFrame 是 Spark SQL、Streaming、MLlib 的基本数据模型.

4.2 数据预处理操作

本文第 3 节提到了对表 1 数据的预处理过程, 列举了 5 个预处理操作. 本系统提供了一系列常用的数据预处理操作组件, 在系统前端以菜单栏的方式分类, 用户只需要点击这些组件并填写相关参数即可实现相应的数据预处理过程.

组件通过基于 RDD/DataFrame 的数据模型实现数据的输入输出, 组件本身大都采用 Scala、SQL 等脚本语言编写, 实现从源 DataFrame 生成新的 DataFrame 过程. 系统为组件制定了一套接口, 运行用户按照标准扩展新的数据预处理组件.

在组件实现方面, Spark 为 DataFrame 带了一套通用的 API 接口, 常用的统计操作(min、max、std 等), 数学算法(频繁集、交叉验证等), 数据操作(去重、映射等)都提供了相关函数, 很多数据预处理操作都可以基于现有 API 实现. 当然, Spark 还支持用户使用 UDF(User Defined Functions)扩展新 API, 系统为常用的数据处理操作封装成 UDF, 简化了数据预处理组件的实现.

表 2 是一些常用的数据预处理操作, 涵盖数据清理、数据集成、数据转换和数据归约等多个方面, 系统将对这些操作逐一实现.

表 2 数据预处理操

类型	数据操作	操作方法
数据清理	缺失值清理	①忽略元组或属性 ②使用默认值 ③预测缺失值
	噪音清理	①分箱 ②聚类 ③回归
	重复值清理	寻找重复值并去除
	数据不一致	基于规则转换数据
数据集成	数据集成	多个数据源整合起来
	模式集成	按照相同元数据模式整合
数据转换	聚集	对数据进行汇总
	归一化	min-max, z-score, 小数定标
	属性构造	依据现有属性构造新的属性
	数据泛化	使用高层概念替换原始数据
数据规约	数据规约	数据立方体聚集、维规约、数据压缩、数值规约、离散化等

4.3 交互式执行引擎

数据预处理操作组件提交后, 系统需提供一个组件的运行时环境, 本文考虑了以下几个问题, 并基于 Spark 实现了一个交互式的执行引擎: ①连续执行, 支持多个操作的连续执行; ②数据传递, 每个操作都是通过数据传递的, Hadoop 技术将数据缓存在在磁盘, 无法满足交互性. 本文系统对 Spark 进行改造, 数据缓存在内存以及分布式内存文件系统 Tachyon[13]中, 达到交互式标准; ③多语言支持, 组件通过 Scala、SQL、Python 等语言实现, 系统为这些组件都提供了支持. 接下来介绍交互式执行引擎的实现方式.

通常使用 Spark 时通过 spark-submit 的方式提交作业, 用户编写的 Spark 应用程序称之为 Application, 其中包含了一个 Driver 功能的代码和分布在集群中多个节点运行的代码, 图 2 显示了 Spark 作业运行架构. Driver 负责创建 SparkContext, 目的是准备 Spark 应用程序的运行环境. SparkContext 负责和 Cluster Manage 通信, 进行资源的申请、任务的分配和监控等. Executor 负责将集群节点的 Task 包装成 TaskRunner, 并从线程池中取出一个空闲的线程运行 Task.

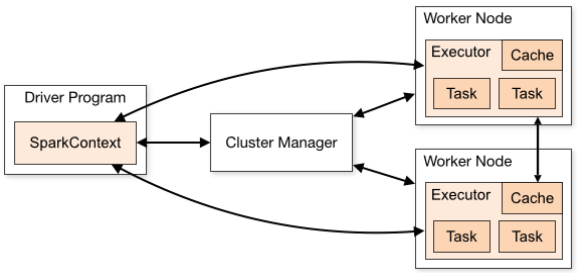

图 2 Spark 应用运行架构图

Spark 应用运行启动 SparkContext, SparkContext 负责向集群注册并申请资源(CPU、内存等), 用户在编写 Spark 程序时可以指定配置信息. 同时, 不同的 Spark 应用之间数据是独立的, 彼此之间不能访问. 现在考虑数据预处理过程, 用户在执行处理数据时时往往需要进行多个操作, 这些操作都是独立的个体, 并且通过 RDD/DataFrame 进行数据传递. 如果每个操作都提交一个作业, 那么这些作业之间的数据无法共享, 并且每个作业都需要占用集群资源, 而集群的资源是有限的, 无法为多用户多应用同时提供资源.

为了解决资源与数据的共享问题, 系统实现了一个共享 SparkContext 并能对 RDD/DataFrame 进行管理

的运行时环境. 如图 3 所示, 首先在 Spark 集群建立一个常驻后台的 SparkContext, 由此生成 SQLContext 以及 HiveContext, 然后合并成 SharedContext, 在这个共享的环境之上可以建立 Scala、Python、SQL、Hive 等多语言的运行环境, 这些环境共享 RDD/DataFrame, 系统针对共享的数据进行管理.

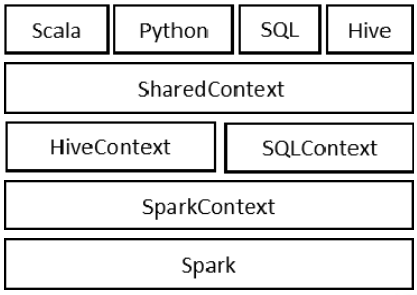

图 3　共享的 SparkContext 架构图

4.4 交互式前端

输入的数据源以及每个操作产生的数据都以电子表格的形式展现在 WEB 界面, 用户可以很直观的查看每个数据预处理操作后的结果. 同时, 用户可以将当前数据状态进行保存, 系统还会自动记录每个步骤的状态. 有了这些状态系统可以提供撤销重做功能, 支持回退到任意指定的数据状态中.

5　系统实现

本文基于 Java Spring MVC 实现了一个交互式数据预处理系统, 并使用 Scala 与 SQL 实现了一套常用的数据预处理组件, 系统整体架构图如图 4 所示. 系统总体分为 4 个模块: WebUI、组件库、执行引擎以及 Spark 集群.

Spark 集群: 提供 Spark 集群环境, 使用 Tachyon 管理 HDFS 以及本地文件.

执行引擎: 在 Spark 集群环境下建立共享的 Spark Context, 并编写 interpreter 层, 提供 Scala、Hive、SQL、Python 等多语言运行环境. Engine 层使用 Apache Thrift 协议与 interpreter 层通信, 实现 Scala、SQL、Python 等实现的数据预处理组件能动态选择相应 interpreter 执行, Engine 层对共享的 RDD/DataFrame 进行管理.

组件接口层: 提供各类数据预处理组件, 并提供可扩展的组件接口. 组件接口层主要依赖 Java 反射与注解实现, ModuleBase 是所有组件类的基类, 使用注解形成组件元数据信息; ModuleRepo 负责所有组件实例的加载, ModuleProxy 提供组件服务接口.

WebUI: 提供交互式的数据预处理界面, 数据以电子表格的形式展现, 操作通过菜单栏方式分类. 用户的每个操作都实时刷新表格, 并记录每个步骤的数据状态以便撤销. WebUI 通过 Rest 接口与后台交互.

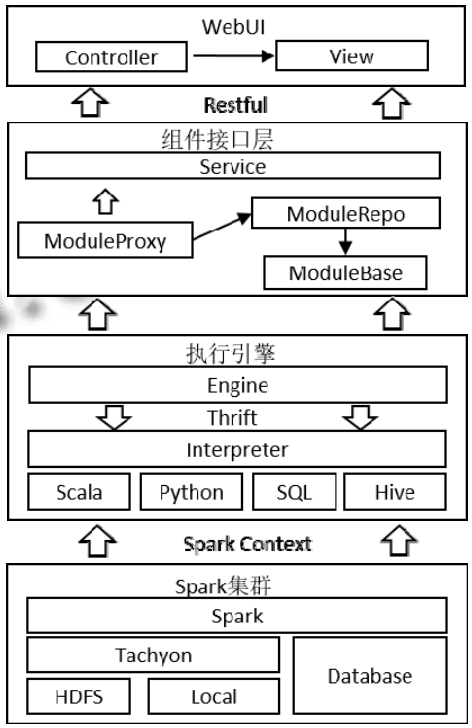

图 4　系统整体架构图

6　实验验证

6.1 实验环境

本文使用 10 个节点的集群作为测试环境, 包含 1 个主节点与 9 个子节点, 每个节点配置都是 4 核、8g 内存, 并搭载 CentOS 6.5 操作系统. 大数据相关软件版本如下: Hadoop 2.5.2, Spark 1.4.0, Tomcat 7.0.55, Scala 2.10.4, Tachyon 0.6.4 等.

实验背景来源于国家卫计委的“中国卒中数据中心”, 数据中心已建立了社区、乡镇、医院门诊和住院患者脑卒中高危人群 4 个数据子库, 覆盖全国 30 个省市 308 家三甲医院、2700 余个社区和乡镇医院, 收集 350 多万人的各类筛查及随访信息. 本节依据数据挖掘的需求, 对数据中心抽取的 17 张表格数据进行预处理模拟操作.

6.2 实验结果与分析

数据挖掘需要对数据进行预处理, 这里模拟了 5 种常见的数据预处理操作: ①缺失值检测: acage(患者

年龄)字段不能为空, 对缺失的数据行删除; ②数据筛选: 根据用户规则筛选数据, 比如 acage 需要满足, 不满足的数据行删除; ③特征选择与维度扩展: 筛选指定数据列并构造新的数据列; ④归一化: 对指定列进行 z-score 归一化; ⑤多表级联: 由多张表的数据依据主键进行级联, 生成新的特征表. 本节使用 Hadoop/Hive, Spark-submit 以及本系统分别实现了以上几种数据预处理操作, 并记录了运行时间.

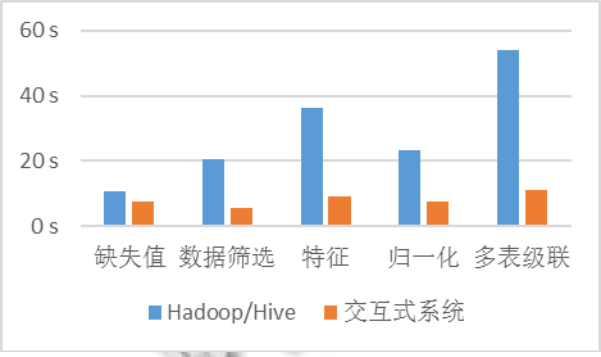

图 6 几种数据预处理操作的耗时对比

从实验结果可以看出以下几点: ①基于 Spark 的交互式系统执行速度优于 Hadoop/Hive, 对于复杂查询以及多表级联查询, 速度具有显著提升; ②系统由于共享数据机制, 对已查询过的数据缓存, 所以下次查询速度明显提升, 比如缺失值检查中的执行速度接近, 但后续操作都有显著提升; ③系统大部分操作都能保持在 10s 以内, 满足交互式需求. 实验结果表明, 基于 Spark 技术的交互式数据预处理系统相比 Hadoop 速度有了很大的提升, 基本达到交互式标准, 能够在大数据场景下满足交互式处理需求.

7 总结

基于 Hadoop 技术的数据预处理存在很多缺点, Hadoop 的批处理方式使得数据处理过程缓慢、效率低下、无法实现交互式. 本文从数据模型、数据预处理操作、交互式执行引擎以及交互式前端四个方面介绍了如何使用 Spark 技术实现一个交互式数据预处理系统, 最后通过医疗脑卒中数据进行验证, 结果表明系统能够很好的实现大数据场景下的交互式数据预处理. 未来将进一步实现更多的数据预处理操作, 同时还将对交互式执行引擎层进行优化, 进一步缩短数据预处理所需时间.

参考文献

1 Han J, Kamber M, Pei J. Data mining: Concepts and techniques. Elsevier, 2011.

2 Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107–113.

3 Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets. HotCloud, 2010, 10: 10–10.

4 陈伟,丁秋林.可扩展数据清理软件平台的研究.电子科技大学学报,2006,35(1):100–103.

5 郭志懋,周傲英.数据质量和数据清洗研究综述.软件学报, 2002,13(11):2076–2082.

6 陈伟.数据清理关键技术及其软件平台的研究与应用[博士学位论文].南京:南京航空航天大学,2004.

7 Raman V, Hellerstein J M. An interactive framework for data cleaning. Computer Science Division, University of California, 2000.

8 Galhardas H, Florescu D, Shasha D, et al. Declarative data cleaning: Language, model, and algorithms. ResearchGate. 2001.

9 Galhardas H, Florescu D, Shasha D, et al. AJAX: an extensible data cleaning tool. ACM Sigmod Record. ACM, 2000, 29(2): 590.

10 Raman V, Hellerstein J M. Potter's wheel: An interactive data cleaning system. VLDB. 2001, 1: 381–390.

11 Datameer. http://www.datameer.com/.

12 Hellerstein JM. Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE), 2008.

13 Li HY, Ghodsi A, Zaharia M, Shenker S, Stoica I. Tachyon: Reliable, memory speed storage for cluster computing frameworks. Proc. of the ACM Symposium on Cloud Computing. ACM. 2014. 1–15.