软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(6):1516−1528 [doi: 10.13328/j.cnki.jos.005231]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

领域大数据应用开发与运行平台技术研究*

王建民

(清华大学 软件学院,北京 100084)

通讯作者: 王建民, E-mail: jimwang@tsinghua.edu.cn

摘 要: 随着大数据技术在不同领域的快速应用,构建大数据应用系统的开发与运行一体化平台,降低大数据技术在各行各业应用普及的门槛,为面向领域的大数据应用系统的快捷开发和高效运行提供方法、工具和平台支撑,成为大数据产业发展的迫切需求.由于大数据固有的复杂性、动态性、多样性及其价值独创性,目前尚未形成系统化的大数据软件开发方法,难以满足不同领域对大数据全生命周期处理的多样化需求.大数据时代的软件工程面临的挑战,体现在互为依赖的两方面:面向大数据全生命周期的集成设计开发环境和基于软件生命周期的系统运行分析工具.结合特定领域的实际需求,研究面向领域的大数据应用系统开发与运行一体化平台技术,覆盖大数据生命周期(获取、清洗、集成、分析、呈现)以及软件生命周期(设计、开发、运行、优化),形成自管理、自适应、自优化的平台化解决方案.在此基础上,开展面向装备物联网及气象民生服务的大数据示范应用,以验证平台的有效性.

关键词: 领域需求分析;一体化平台;运行时分析工具;大数据生命周期

中图法分类号: TP311

中文引用格式: 王建民.领域大数据应用开发与运行平台技术研究.软件学报,2017,28(6):1516−1528. http://www.jos.org.cn/1000-9825/5231.htm

英文引用格式: Wang JM. Key technologies in big data applications development and runtime support platform. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1516−1528 (in Chinese). http://www.jos.org.cn/1000-9825/5231.htm

Key Technologies in Big Data Applications Development and Runtime Support Platform

WANG Jian-Min

(School of Software, Tsinghua University, Beijing 100084, China)

Abstract: Big data technology is widely adopted across many disciplines. In order to build sustainable big data application systems and facilitate its rapid development and delivery of expected values with minimum efforts, innovative software engineering methodology and an integrated development and management platform for big data applications are in dire needs. Big data is complex, volatile, lack of correlation and value scarce by nature, which makes it difficult to form standardized and systematic technological solutions to meet the diversified requirements for life cycle management of big data in different application domain. Software engineering in big data era has to address two major challenges: data life cycle management with integrated development environment and software life cycle management using run-time behavior analysis tool. This paper proposes a domain requirements driven approach for big data application systems development and run-time support platform, covering the entire big data life-cycle, including dada collection, storage, computation, analysis, visualization, as well as the software systems life cycle. This platform forms a self-managing, self-adaptive, self-optimizing solution. The proposed techniques are applied in specific application domains such as industry 4.0 and meteorological engineering to provide an illustration and validation of the new platform.

* 基金项目: 国家自然科学基金(61325008); 国家科技支撑计划(2015BAH14F02); 清华信息科学与技术国家实验室(筹)大数据科学与技术专项

Foundation item: National Natural Science Foundation of China (61325008); National Science and Technology Support Program (2015BAH14F02); Data Science and Technology Project by Tsinghua National Laboratory for Information Science and Technology

收稿时间: 2016-10-17; 修改时间: 2016-10-26; 采用时间: 2016-12-22; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 15:06:11, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1506.028.html

Key words: domain requirements analysis; integrated platform; runtime requirements monitoring; big data life cycle

数据作为计算的处理对象,与软件是密不可分的.数据的获取与清洗、集成与分析、呈现与应用等环节都离不开软件的支持.在大数据时代,软件系统和工程面临的挑战体现在互为依赖的两方面:一方面,软件系统与工程应针对大数据处理的需求,研究如何开发支持大数据处理各个环节的软件技术与系统,形成面向大数据的软件工程——面向大数据生命周期的一体化集成设计开发环境;另一方面,软件系统与工程实施过程中会涉及大量具有大数据特征的系统运行过程数据,因此有必要对这些多维数据进行充分的关联挖掘和机器学习,发现数据驱动的软件开发和运行规律,形成基于大数据的软件工程方法学,指导大数据软件系统的开发——面向软件生命周期的大数据应用系统运行分析工具.

大数据应用系统覆盖数据的获取、清洗、集成、分析与可视化等大数据全生命周期的多个处理环节[1],而每个环节都存在着多款软件工具,它们以开源软件构件形式在大数据生态系统中“野蛮生长”,这给面向领域的大数据应用系统构建、运行与优化带来了挑战.本文针对现有大数据软件构件选型配置困难、维护管理代价昂贵等难题,研发大数据应用系统的开发运行一体化平台.在设计开发阶段,支持大数据应用系统的辅助选型、代码生成和自动部署;在运行分析阶段,支持大数据应用系统的自管理、自适应、自优化等特性,减少大数据应用系统的开发成本及运维代价.

面向大数据生命周期的软件工程方法学和面向软件生命周期的大数据软件系统,特别是两者之间的本质联系,是本文研究的核心问题.通过对大数据应用系统的开发与运行的共性模式进行抽象,从而形成一体化开发运行平台的架构体系.具体来说,本文的主要贡献包括:

1) 提出大数据应用开发与运行平台框架,创新面向大数据应用研发的软件工程方法:本文所开发的大数据系统开发与运行一体化平台覆盖大数据全生命周期以及软件全生命周期,形成自管理、自适应、自优化的平台化框架;
2) 提出面向领域的大数据构件辅助选型与自动部署方案:大数据应用需求可直观表现为加工数据的工作流程,通过复用已经积累的工作流程模板与实例数据,根据大数据应用的典型业务流程提炼出系统资源需求的计算模块,估算各个步骤所具有的数据处理特征及需要的处理资源,并利用构件知识库估算出相应的优化配置参数,选定合适的软件构件及相应配置[7],避免因系统知识缺乏而盲目指定参数导致低效的系统资源部署,从而解决系统资源依据大数据应用需求进行弹性配置的难点问题;
3) 提出基于系统日志分析的大数据系统优化方法:通过收集大数据应用系统运行日志,分析系统资源使用情况和计算能力瓶颈,对系统资源部署进行迭代改进,在优化应用计算效率的同时,避免静态资源配置造成的计算能力浪费.引入系统化的数据挖掘方法对大数据应用执行过程数据进行分析,更新大数据构件库中相应的模型知识,为类似负载的系统配置提供指导.

本文第 1 节介绍一体化平台的研究动机.第 2 节详细阐述本文提出的一体化平台框架及相关技术.第 3 节为平台的应用情况分析.相关工作和总结展望分别在第 4 节和第 5 节给出.

1 研究动机

大数据技术正在从消费互联网向产业互联网渗透,开发人员从复合型极客转换为产业领域型人才,大数据应用推广面临新的挑战,具体表现为:领域大数据应用开发运行过程需要基于专家知识反复迭代试错,技术门槛与开发成本过高,系统选型、配置、部署困难.通过总结工业大数据、气象大数据、医疗健康大数据等多个应用领域研发经验,作者认为,造成上述问题的主要原因包括:大数据应用需求描述缺少框架指导,大数据领域尚缺乏有组织、可管理的大数据软件构件资源库,大数据软件构件集成部署比较困难,系统运行分析优化缺少自动化工具.归根结底,就是缺乏关于大数据应用系统开发的软件工程方法学.

本文研究聚焦两个方面.

- 第一,面向大数据生命周期,包括数据采集与清洗、集成与存储、分析与呈现在内的一体化集成设计开

发环境,支持高抽象层次的、面向领域的需求描述框架,具有用户友好的可编程性、并能描述复合业务需求的过程模型.开发环境支持从大数据构件资源库中根据用户需求辅助构建面向领域的大数据应用系统,具备系统设计、脚本生成、自动部署等主要功能.开发环境同时支持业务模型和操作序列优化,是大数据系统的软件生命周期的核心组成部分;

- 第二,面向软件生命周期,包括设计、开发、运行与优化的大数据应用系统运行分析工具,支持应用系统的自管理、自适应和自优化.运用机器学习、数据挖掘、关联分析、信息可视化等技术,有效处理、分析软件运行生命周期中生成的运行数据,如运行日志、系统配置、部署脚本、性能指标等,从中提取有用信息,做出优化决策,帮助软件开发者以数据驱动方式进行软件的运行优化;支持运行数据的分析方案设计、数据分析算法、系统运行优化、模型配置更新等主要分析任务,形成大数据系统运行时的知识库.

本文工作按图 1 所示的软件生命周期进行展开,包括大数据构件资源库、集成设计开发环境、系统运行分析工具、大数据应用系统这 4 个部分.大数据构件资源库作为整个平台的元数据存储引擎,负责存储用户状态、资源状态、集群状态等元信息,并为其他模块提供查询和更新接口,支持根据用户对大数据存储、计算和分析的需求,自动选择合适的大数据存储系统、计算系统和分析系统,并且能够根据用户需求和构件特点进行系统的软硬件参数自动配置和运行前优化.同时,还资源库模块处于系统消息的中心,连接需求模块、分析优化模块与自动部署模块,负责系统的状态存储与消息传递.主要内容包括:

- 大数据构件自动选型:根据用户提供的需求信息,自动选择最适合应用的大数据系统构件;
- 大数据系统硬件容量规划:根据用户提供的数据负载等需求信息,对大数据系统进行硬件容量规划,给用户提出硬件容量建议;
- 大数据系统软件参数配置与优化:根据用户提供的需求信息和硬件规划结果,对大数据系统软件参数进行初始配置与优化;
- 大数据构件资源库动态更新:通过配置文件更新大数据构建资源库的选型知识、参数配置知识,并且能够自动学习用户的反馈.

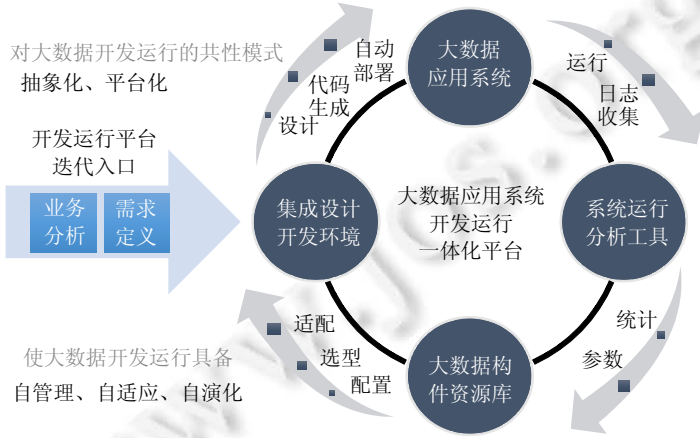

Fig.1 Big data application systems development and operation support platform

图 1 大数据应用系统的开发与运行支撑平台

大数据构件资源库的运行分为 3 个阶段:训练阶段、使用阶段和动态更新阶段.在使用前,首先要进行训练,训练选型决策树和性能模型,初始化软件参数配置规则库;训练完成后,可以用于解决大数据系统的构件选型和参数配置问题;在之后的使用过程中,选型决策树、性能模型和软件参数配置规则库可以根据用户的反馈进行更新迭代,不断地学习新的知识和方法.

在使用阶段:第 1 步是根据规范化的需求指标获取用户需求,通过用户友好的图形界面,用户回答大数据构件资源库提出的一系列问题,即可生成量化的需求指标;第 2 步是系统选型,用户可以自动选型,大数据构件资源库根据需求指标选择合适的大数据系统构件,用户也可以辅助选型,对大数据构件资源库推荐的构件进行修改或确认;第 3 步是硬件容量规划与资源分配,大数据构件资源库通过训练阶段得到的性能模型对用户提出的性能指标进行求解,给出最佳的硬件容量规划方案,用户可以修改或直接确认,完成硬件资源配置;第 4 步是大数据系统初始软件参数配置,大数据构件资源库通过规则系统计算最优的初始软件参数配置,由用户确认或修改;第 5 步是生成部署计划,部署计划是一个 XML 格式的详细配置清单,包括系统选型、硬件资源配置清单、每一个大数据系统构件的初始软件参数配置,然后提交到自动部署工具,生成大数据系统.一体化平台对自管理、自适应、自优化的支持分别体现在从业务需求导出指标化和参数化的需求定义,构件的自动选型基于大数据资源构件自动生成大数据应用系统代码,并自动部署,根据运行日志分析结果完成自适应调整和跨层优化的过程.

2 领域需求驱动的自动选型和部署

关于“领域”,本文具有两方面的含义:首先,将大数据本身作为“特定领域”进行建模与分析,是本文的核心研究内容;其次,将特定行业作为“应用领域”进行大数据平台技术应用示范.前者主要解决框架、方法问题,后者主要是对于上述框架和方法的实例化应用验证.

需求获取作为用户进行大数据应用系统开发的入口,涵盖了操作环境、操作对象、操作步骤和操作流程 4 个方面的功能性需求和非功能性需求.集成设计开发环境模块通过简单、直观、易懂的操作界面完成了用户的需求获取,为后续系统选型和参数配置做准备(如图 2 所示).需求分析利用大数据构件资源库对采集到的用户需求进行处理,确定合适的硬件配置和系统构件,从而做出系统选型以及参数配置的推荐.在这个过程中,有经验的用户可以对推荐的系统和参数进行干预,使得系统更加符合需求.对于熟悉各种大数据系统构件的高级用户,他还可以选择跳过需求获取和分析,直接进入到系统选型和参数配置.

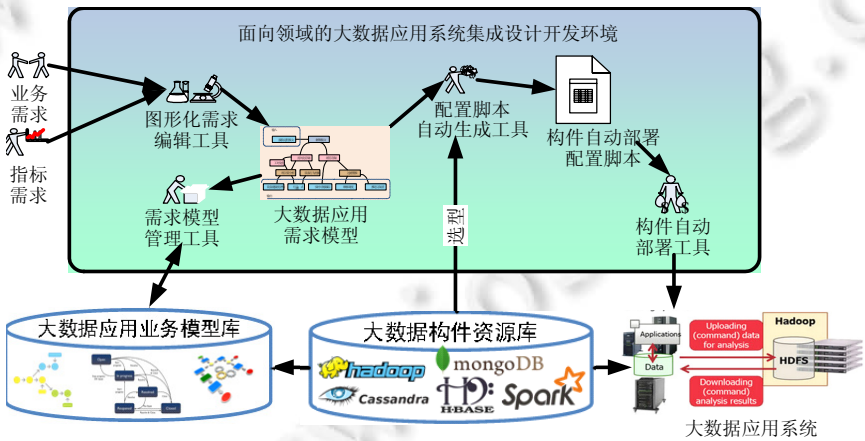

Fig.2 Domain requirements driven big data applications development environment

图 2 面向领域的大数据应用系统集成开发环境

大数据系统开发平台是面向用户的,需要区分不同用户之间的集群并提供方便的增、删、改、查功能.因此,用户模块是集成设计开发环境的重要组成部分.在用户模块控制下,新用户需要注册自己独有的帐号,系统通过帐号的唯一性,保证每个用户具有自己的系统空间,使部署的大数据系统之间相互独立、相互隔离、互不影响.用户需求描述需要表达大数据应用系统各种资源和业务,如硬件资源、软件资源、领域知识、配置参数、性能指标等需求的统一入口,其表达范围将覆盖典型大数据应用系统构建需要的功能性需求,如操作对象、操作步骤、操作流程和非功能性需求指标,如用户规模、数据规模、性能要求.从操作环境角度来看,既要支持对

单一类型资源的需求定义,也要支持多种类型资源配套需求的表达;从操作对象角度来看,既要支持输入数据、输出数据的明确定义,又要支持中间临时数据的表达;从操作步骤角度来看,既要支持开发者直接对所需各类硬件资源、软件资源、领域知识、配置参数和性能指标的精确表达,又要支持对上述所需资源的抽象/模糊表达;从操作流程角度来看,既要支持操作步骤间常见的顺序、同步等基本次序关系的表达,还要支持并行、循环等高级控制流结构的表达.

自动部署模块以 OpenStack 为支撑,通过 OpenStack 创建各种版本的 Linux 虚拟机,同时配置虚拟机的硬件和软件参数,自动启动虚拟机,进而在虚拟机上部署相应大数据构件,调整大数据构件参数,启动或停止大数据构件服务等.自动部署模块各接口可以接受前端调用,结合虚拟机集群和大数据构件的各种参数,自动生成虚拟机并同时部署大数据构件,并将生成集群的各种信息返回给前端.简言之,自动部署模块根据用户需求,自动部署用户所需要的大数据构件集群.不同的大数据构件所需要的部署环境和配置参数不尽相同,因此在执行自动部署时,要根据不同需求来设计特定大数据构件的部署逻辑.

在实际开发的过程中,以构件为单位,逐一对大数据构件的自动部署进行设计开发,并完成对 HDFS,Hbase,Cassandra,Hadoop,MapReduce,Spark,MLBase 和 SparkSQL 的自动部署.

系统管理涵盖不同大数据系统构件状态的查看和监控,同时还可以利用运行分析工具进行参数优化和日志诊断(如图 3 所示).参数优化工具根据运行监控工具中的数据为进行分析,得出集群的整体状况(如负载过高、网络流量过大等)并给出优化方案.同时,该工具还会对 Spark 系统运行的任务进行参数推荐,减少任务的运行时间与资源开销.日志诊断对 Spark 系统运行日志进行挖掘分析,可对系统常见异常进行诊断和定位,从而辅助修改运行任务的配置,提高任务运行的性能.

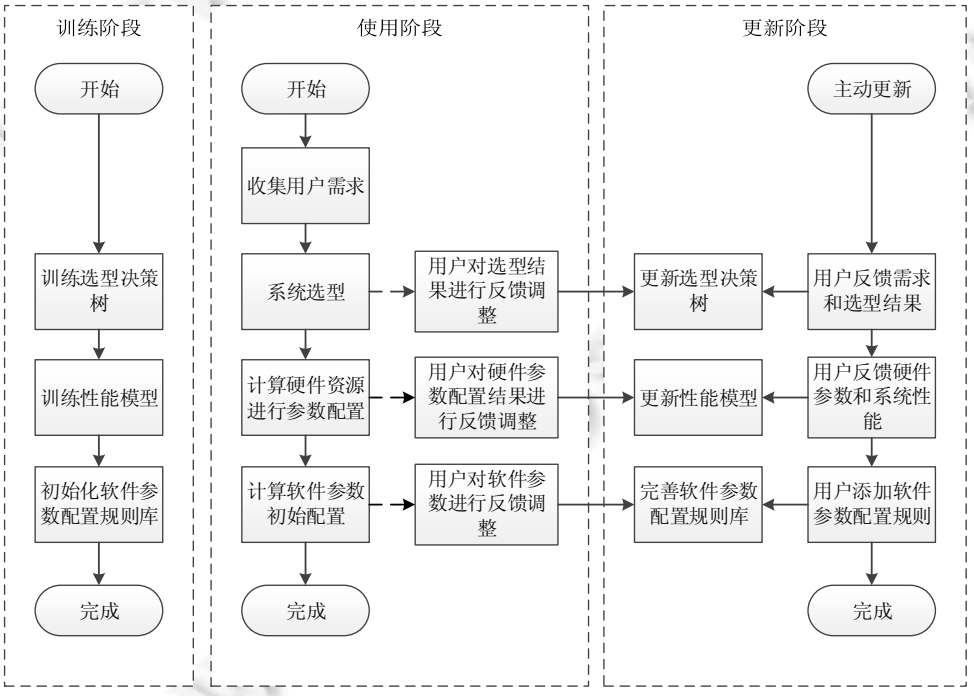

Fig.3 Big data components repository management process

图 3 大数据构件资源库管理流程图

2.1 大数据领域需求描述

通过总结多种应用场景的实践经验,作者将大数据应用的需求进行了详细的划分,从需求类型、数据类型、业务流程、性能指标等多个方面提取了 26 个需求指标,其中,与选型相关的有 14 个:存储类型、数据一致性、

数据类型、文件或单条数据规模、数据到达方式、数据模式特点、查询类型、计算类型、分析类型、查询响应时间、计算响应时间、分析响应时间、系统可用性、是否进行数据导入.基于上述需求指标,形成了基于决策树的大数据构件选型方法,对于存储、计算、分析等 3 种需求,分别构建决策树.实现时,采用 C5.0 算法构建决策树.构建决策树时,使用已知的一组数据,包括不同条件下的选型结果.系统内置了初始的知识,用以构建决策树;系统也支持用户增加新的知识,当用户对某一次的选型结果进行修改时,资源库会收到相应的反馈,一条新的知识(训练数据)将会添加到已有的知识中,并重构决策树.

2.2 多维度系统配置与参数优化

在大数据构件资源库模块中,实现了一种新的多维度的系统参数配置和优化解决方案,包括硬件参数、操作系统参数、大数据构件参数等,能够自动地进行系统参数配置和运行前优化.对每个大数据系统构件,通过网格搜索方法分析、测试不同的参数配置下的大数据系统的性能指标,结合不同参数配置的成本因素、硬件资源,通过多元回归、主成分分析等方法,建立起性能指标和系统硬件参数的相关性模型,并能通过该模型计算出用户需求情况下最优的硬件参数配置.对于大数据构件参数,通过构建基于规则系统的大数据构件参数配置知识库,实现软件参数的自动配置.软硬件参数配置都支持用户反馈和动态更新,用户可以对配置结果进行修正,大数据构件资源库能够学习用户反馈,以提高参数配置的准确度.

2.3 基于性能模型的硬件容量规划

在大数据应用系统中,硬件配置是决定大数据系统性能最根本、最主要的因素,除了要满足系统性能需求,还受限于用户的基础条件和投资成本,又称作硬件容量规划.硬件容量规划的目标是:针对不同的应用负载需求,如数据量、数据生成速度、数据分布情况、读写负载比例、计算速度要求等,计算并匹配合适的系统资源和系统配置.通过综合各种因素,包括用户需求、性能指标、系统架构和特点以及资源代价,构建一种较适合参数配置的性能模型.对照用户的需求,与性能相关的是读写吞吐量、读写延迟、客户端并发数,并以底层的大数据存储系统作为需求的基础进行参数初始配置,若需求存在计算和分析构件,则分析其性能是否满足要求,并调整配置.对于每一个大数据存储系统,对于上述 5 个性能指标,分别对 4 个参数共约 60 组不同组合进行实验,通过使用多元回归的方法建立起 5 个性能模型,刻画参数配置与系统性能的关系.

以 Cassandra 为例,写吞吐量的性能模型为

$$WriteThrought(byte/s)=(-536+2837\times C+1120\times\log_2 M-87\times C\times\log_2 M)\times(0.4+0.5\times N+0.1\times\sqrt{N}),$$

其中,N 为节点数,C 为 CPU 核心数,M 为内存大小.

通过求解性能模型,可以得到满足给定性能需求的最优参数配置.同时求解多个性能模型,求其中的最大值,即为满足所有性能需求的最优参数配置.但是,因为该模型可能有很多个解,因此需要同时求出取值范围内的若干个解,通过计算每一个解的价格,求出满足性能需求的最小价格的参数配置.

参考阿里云和亚马逊 AWS 等主流云计算平台,资源库使用如下的价格计算公式:

$$Price=N\times(45\times C+22\times M+4\times(\log_2 M-14)).$$

2.4 基于规则系统的软件参数配置

软件参数配置是指大数据构件运行时的参数配置,如队列大小、缓存大小、并发线程数等,通常通过系统的配置文件进行修改.软件参数配置的目的是提高系统的性能,通过构建了几种大数据系统的软件参数配置知识库,实现基于规则系统对软件参数进行配置.规则系统具有以下几个特点.

1) 规则优先级:在对某一个参数的配置存在多条规则时,取优先级最高的作为配置结果,若有多个优先级最高的规则,根据参数值的类型分两类情况.若为数值型参数,取所有规则配置结果的平均值;若为其他类型的参数,取所有规则配置结果的众数作为配置结果;
2) 输入参数:用户需求指标、配置完成的硬件参数和部分软件参数;
3) 每条规则都有一个条件,条件可以是既有参数的逻辑表达式,也可以是多个逻辑表达式的组合;
4) 每条规则都包括一个赋值表达式,表达式可以是具体的值,也可以是既有参数的算术表达式;

5) 规则系统支持用户动态添加新的规则,使得参数配置知识库得到更新.

面向领域的大数据应用系统作为一种大型复杂软件系统,在整个迭代式软件生命周期中会产生大量的、多样化的数据,例如开发过程中的源代码、需求文档、缺陷报告、测试用例,系统运行中的运行日志、性能度量、事件记录,用户交互操作行为序列、相关性反馈等.其中:运行日志数据覆盖了大数据系统运行时的事务操作、用户行为、网络动作、机器行为等丰富信息,是实现大数据系统智能化运行的关键数据资源.当前,面向领域的系统运行维护已经成为大数据技术的成本瓶颈,因此,如何对复杂的大数据系统进行智能化运行,提高系统自适应、自管理能力,是降低大数据系统运行成本的关键.基于系统运行数据,软件开发者可以分析关于软件质量和开发模式的重要信息.因此,研究如何运用关联分析、机器学习、数据挖掘、信息可视化等技术,有效处理、分析软件运行生命周期中生成的数据,从中挖掘有用的信息、做出优化决策,帮助软件开发者以数据驱动方式进行软件的开发、运行和维护,变得越来越重要.

3 基于运行数据分析的系统性能优化方法

在大数据应用系统运行时,运行时数据管理系统会实时地收集各大数据构件所产生的数据,并同时进行数据存储和建立索引的工作,满足大规模运行数据的管理需求.除此之外,该框架中所存储的数据还将输入到参数推荐系统中,为其提供丰富的历史数据.

3.1 运行时数据收集

通过分析系统运行过程中所产生的日志类型,并从用户需求和系统实用性方面考虑,系统需要收集如下 4 种类型的运行数据.

1) 系统各集群的监控时序数据:
   - CPU:系统占用率、用户占用率;
   - 网络:输入字节数、输出字节数;
   - 硬盘:写入字节数、读取字节数;
2) 大数据构件模型数据;
3) Spark 任务调度图数据(有向无环图,DAG);
4) 大数据构件配置参数数据.

各大数据构件日志数据:Map/Reduce,Spark,Cassandra,HDFS 存储模块以分布式任务(Job)为最小划分单元,对所收集的数据进行预处理后存储到非结构化数据库 MongoDB 中,其中,每条任务数据的格式见下表.

字段	类型	说明
id	string	对应任务 id
status	int	任务完成状态: 0——正常完成 非零值——失败
config	Embedded document	提交任务时,用户所设置的参数
limited_resource	Embedded document	在执行任务时所能提供的资源限制
start_time	date	任务运行起始时间
end_time	date	任务运行结束时间
journal_data	Embedded document	抽取的日志数据特征
monitoring_data	Embedded document	每台机器的监控数据,包括 CPU、内存、网络 I/O 和硬盘 I/O 状况
rdds_data	Embedded document	该任务中的有向无环图数据

运行时数据管理系统为用户提供了类似于 Splunk 的系统运行日志数据检索功能.用户可编写一定查询语句对日志数据进行检索,目前可以执行以下查询请求:(1) 查询特定时间范围的日志;(2) 查询含有某些关键字内容的日志;(3) 查询某个大数据构件的日志;(4) 查询集群中特定节点的日志;(5) 以上 4 种查询类型的结合查询.

3.2 基于相似度搜索的参数优化工具

本文通过对多类型的运行数据(日志、监控、建模数据等)进行统计分析和机器学习,实现系统配置参数的自动优化.通过系统实验发现:任务执行时产生的运行数据可以反映出任务自身的特性,不同任务之间的相似性可用运行数据的相似性进行衡量.大量系统实验表明,相似任务的最优配置参数相近.因此,为当前任务找出运行情况较好且相似的历史任务,即可将相似的历史任务配置参数作为参数优化的结果;同时,通过引入用户反馈机制,形成带标记参数知识库,作为参数优化的指导信息.

以分布式计算框架 Spark 为例,以任务运行时长作为性能指标,以更优的配置参数作为输出目标.通过收集 Spark 运行时产生的反映系统性能或任务特征的 3 种运行数据(日志数据、监控数据和 Spark 任务有向无环图),并基于上述运行数据构建历史数据库.通过计算不同类型 Spark 任务之间的有向无环图相似度,并利用日志数据和监控数据对相似性搜索结果进行过滤,完成了对 Spark 特定任务的参数优化.

3.3 参数选择及基准测试(benchmark)

Spark 分布式系统配置参数很多,但能够显著影响任务性能的重点参数有限.通过对 Spark 进行源码研读、分析与逆向工程,确定了主从节点的内存大小、执行器(executor)数量、数据序列化方式、数据并行度等 5 个重点参数作为研究对象.这些参数都能够显著影响 Spark 特定任务的系统性能,使参数优化能够做到有的放矢.为了形成有价值的历史数据库,作者采用对 Spark 集群进行基准测试的方法构造历史数据库.通过在不同配置参数下运行多种机器学习任务,构造基准测试的任务集.

基于图相似性的参数优化工具首先获取 Spark 系统中历史任务的运行数据,构建历史数据库;进行任务参数优化时,根据约束条件对历史数据库中显著不相关的运行数据进行一次过滤;然后,对待优化任务对应的运行数据与一次过滤后的运行数据进行有向无环图的相似度计算,并对相似度低于一定阈值的运行数据进行二次过滤;最后,将两次过滤后的结果经过计算排序,并将排序后的运行数据所对应的任务参数作为任务参数优化结果.

3.4 相似度定义

系统实验表明,任务的有向无环图可以反映出任务自身的执行特性.因此,不同任务之间的相似性可定义为任务的有向无环图相似性.基于相似度搜索的参数优化工具特点和有益效果是:让计算机承担 Spark 系统中的任务参数优化的工作,减少了用户在使用 Spark 系统时的工作量.在用户不熟悉 Spark 系统的情况下,能给用户提供较为有效的任务参数,减轻了使用 Spark 系统的压力.工具结合了系统优化的经验规则和基于相似性搜索的优化方法,提高了任务参数优化的可靠性和可用性.工具会在 Spark 系统运行过程中收集 Spark 系统任务的运行数据,通过分析这些运行数据,工具能够适应系统的变化而进行改变,是一种自适应调优方法.工具具有即插即用型的特点,无需改动原有的 Spark 系统即可运行.

4 基于日志分析的 Spark 系统异常检测与诊断工具

当前,面向领域的系统运行维护已经成为大数据技术的成本瓶颈,因此,如何对复杂的大数据系统进行智能化的检测与诊断,提高系统自适应、自管理能力,是降低大数据系统运行成本的关键.本文使用机器学习、数据挖掘、信息可视化等技术,帮助软件开发者以数据驱动方式进行软件的开发、运行和维护,有效处理、分析软件运行生命周期中生成的数据,从中挖掘有用的信息,做出优化决策.

基于这一思路,Spark 异常检测与诊断工具通过对日志数据进行收集、处理和特征提取,进而使用异常检测与诊断模型对日志实现智能化的分析和优化,降低用户使用 Spark 系统的门槛.Spark 异常检测与诊断工具主要由 4 部分组成,包括日志预处理、日志特征构造、异常检测、可视化分析和异常诊断,分别详述如下.

4.1 日志预处理

在这个模块,完成对原始日志的所有预处理工作,包括日志收集、模板提取和日志匹配这 3 部分.本文首先

使用通用的日志框架 Log4j 收集用户 Spark 集群产生的日志,不需要对用户集群配置额外日志收集工具,只需要对 Log4j 的配置文件进行修改即可.对于部署在 Yarn 上的 Spark 集群,也可以使用 Yarn 的日志聚合工具进行日志收集.获取日志后,需要将日志转化为结构化可被机器识别的数据形式,本文使用模板库完成这一工作.模板库是基于对系统源代码中日志输出片段的提取得到的,可以与实际的日志进行匹配,可以理解为匹配日志的正则表达式仓库.

4.2 日志特征构造

本文基于对 Spark 系统内部机理的理解,结合源码解析,提取出针对于 Spark 系统的日志特征:数据分块 RDD 特征和任务单元 Task 特征,特征提取结合在日志匹配的过程中.RDD 特征用来描述作业执行流程中数据块的特征,从数据流的角度发现系统的异常表现;而 Task 特征用来描述作业执行流程中子任务的特征,从任务流的角度发现系统的异常表现.同时,为了提高性能,满足大数据规模下日志处理和特征提取的需求,本文可实现对两类特征的分布式提取.

4.3 异常检测模型

本模块将 PCA 和 *K*-means 两种异常检测的算法进行融合,进而共同完成对两类特征单元的检测.基于主成分分析的异常检测算法是利用 PCA 过程中寻找主方向的思路构造异常子空间,这样,通过度量数据点到正常子空间的距离来判断其是否为异常点,在确定判断阀值时,使用统计检验中常用的 Q 统计量来自动决定阈值;基于聚类分析的异常检测算法是使用 *K*-means 算法首先对特征单元进行聚类,然后结合专家知识分辨出正常类别以及各个异常类别.本文将这两种异常检测算法进行融合,首先使用 PCA 将数据分为正常点和异常点,然后对异常点再进一步进行聚类分析.同时,为了提高性能,满足大数据规模下日志分析的需求,本文实现了异常检测算法的分布式.

4.4 可视化分析和异常诊断

本模块通过构造决策树对检测结果进行可视化展示,决策树帮助用户找到哪些日志最能导致异常的发生.同时,本模块将特征单元与 Spark 的监控信息结合在一起,将异常单元映射到任务代码,辅助用户检查可能导致异常的代码,这将帮助用户理解和诊断异常,进而优化代码或任务配置(如图 4 所示).

基于对 Spark 运行机制的理解,提出了两类针对于 Spark 系统的日志统计特征,并可有效检测出常见的 Spark 异常.在不需要修改系统源代码的情况下,将日志特征与 Spark 的分阶段信息相结合,实现了对 Spark 任务的代码诊断,大大提高工具的实用性.

5 领域应用

大数据技术正在从消费互联网向产业互联网渗透,本文选择装备物联网大数据和面向天气预报的气象大数据两个领域对上述平台关键技术进行应用验证:前者主要关注领域大数据应用功能实现,后者主要关注跨分析与存储层的大数据系统性能优化.

5.1 装备物联网大数据应用实例

X 集团有限公司在企业信息化的发展过程中,通过工程机械的传感器数据获得了大量数据.这些数据会实时地通过移动网络发送给其 M2M 服务平台,并存储在中心数据库中.M2M 平台日均收到监测数据量为 20G,现有起重机数据 9 亿条,每半年产生的主机工况数据 114 亿条,电子工况数据 185 亿条.之前使用关系数据库进行监测数据的存储,当数据到达百亿级别后,查询性能快速下降,无法满足实际工程需要.因此,X 集团急需采用大数据系统进行集群式部署,以解决性能瓶颈问题.作者基于一体化平台,首先分析了 X 集团的业务需求,以确定合适的大数据系统.其应用需求如下.

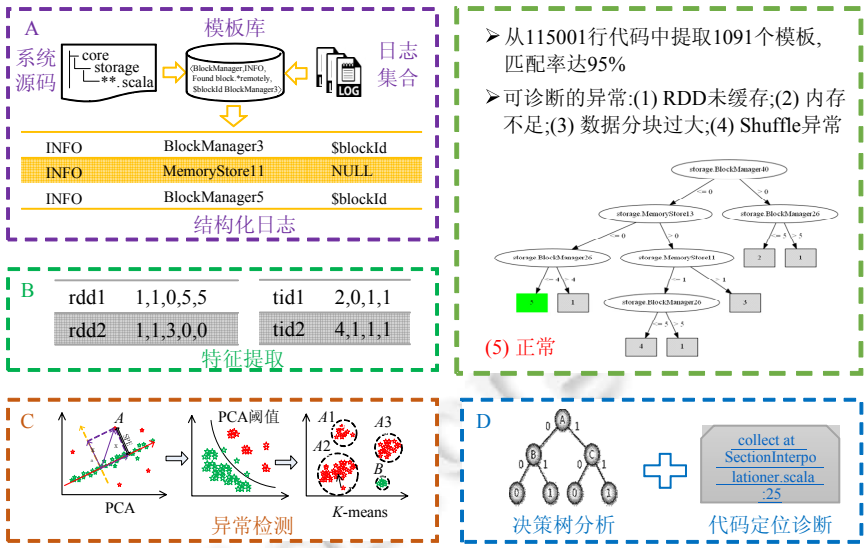

Fig.4 Spark exception detection and diagnosis tool

图 4 Spark 异常检测与诊断工具原理图

- R1:大表查询.X 集团的用户经常需要在大量工况数据在线的情况下查询某个特定编号的主机在某段时间内的某种工况值,从而为用户对主机的管理与检测提供参考依据.该需求要求在百亿数量级下,能在几十毫秒内返回结果;
- R2:监测数据写入.X 集团已经累计售出超过 20 余万台设备,每台设备在开工运作期间均会向 M2M 平台发送监测数据,这些监测数据需要被实时地写入到系统中.该需求要求系统支持每秒 10 万条数据的写入;
- R3:历史工况纠错.X 集团生产的主机在使用过程中传回来的值,由于网络原因会出现一些异常情况,纠正历史工况错误的工作是典型需求.该需求要求系统能在百亿条数据内对错误数据的修改在秒级完成;
- R4:起重机防锈保养.机器如果长时间不用会导致生锈,从而对机器的性能或者正常使用会产生一定的影响.为了对机器进行防锈保养,需要对一段时间内某一种或几种工况的工况值进行统计分析,向用户给出需要进行防锈保养的机器编号.该需求要求系统能在百亿条数据内进行分钟级的数据查找;
- R5:故障预警.X 集团的重型机械在正常情况下,每隔一段时间会发回一次工况数据.基于这些工况数据,用户希望通过故障预警的功能,定时分析这些数据,预测出发生故障可能性较大的机器,并提前告知用户.该需求要求系统具有海量数据的分析能力.

除分析业务需求外,我们也对 X 集团的数据形式进行了分析.X 集团的工况数据可看做四元组(E,S,T,V),其中,E 为机械设备编号,S 为工况参数标识,T 为接收时间,V 为工况参数值.即:一个机械设备 E 的工况参数标识为 S 的传感器在时间 T 产生一个工况值.由于传感器数据实时地通过 M2M 平台发送给服务器,因此,该企业用接收时间来代替工况数据的产生时间.

该企业对工况数据有以下需求:可以按照接收时间范围对某个机械设备编号的某种工况进行查询,可以快速地对工况数据的最新值进行查询,可以基于不同的接收时间对工况值进行带条件的聚集操作等等.

通过对上述需求的收集和开发运行一体化平台系统选型工具的评估,我们采用 Cassandra 和 Hadoop Map/Reduce 大数据系统对 X 集团的平台系统进行了升级.该大数据存储平台采用多服务器组网方案,可根据业务规划、容量大小调整客户机,存储服务器节点数量.在多服务器组网架构下,能突破单节点处理性能上限,满足大容量、高性能要求.同时,在容灾、负载均衡上有更高的稳定性.该平台与 M2M 智能服务平台进行整合,使用

该平台架构可以在使用 M2M 真实监测数据、不干扰在线系统正常运行的同时,对数据库中大数据进行任意操作,既可以作为主要的生产系统,亦可作为 X 集团大数据的备用平台,便捷安全.

在升级中,针对 X 集团单条数据量小、数据总条数多的特点,作者将传统关系型数据库中的长表存储改为宽表存储,节约存储空间,提高查询速度.M2M 平台利用大数据存储系统实现了海量级别的监测数据存储、查询和分析.经过改造的平台能够支持对 10 万台以上设备的数据进行管理,支持 300 亿以上监测数据的快速存储,支持每秒写入监测数据达到 10 万条以上.在多种工作场景下,表现出优于传统关系数据库的性能.

经测试,在 1 100 亿数据量级下,“大表查询”操作能在毫秒内返回结果.而在原有关系数据库中,需要耗时数秒.“监测数据写入”操作可以达到每秒十数万条监测数据的写入速度,满足当前以及未来长时间内的应用需求.“历史工况纠错”操作在存储百亿条记录的前提下,对错误数据的查询能在分钟级别内返回,对错误数据的修改能在毫秒内完成,而现有关系数据库则无法支持海量数据下的在线查询[14,16].“起重机防锈保养”操作在存储百亿条记录的前提下,能在分钟级别内找出需要防锈保养的车辆信息,而现有关系数据库则无法支持海量数据下的在线查询.对于“故障预警”操作,M2M 平台利用 Map/Reduce 分析系统实现了在海量数据基础上的数据分析挖掘功能,对观察机械设备运转情况、及时发现故障具有重大意义.我们通过对工程机械行业特点的研究,推出了从时间、主机和工况的多维度分割合并方法,并给出了用于评价工况数据的关键指标体系,并完成了在大数据基础上的快速计算过程.在此基础上,设计了敏感工况分析工具、开机偏差热度图以及主机工况时序图等一系列可视化方法,为大数据技术在工业界应用提供了一套可操作的方法.平台通过对 590 亿条电子工况数据、380 亿条报警数据、120 亿条油耗数据和 10 亿多条故障记录进行决策分析,发现了 9 000 多个工程机械运行和操作异常现象,识别出 100 余种异常操作行为和安全隐患;通过基于智能传输协议的企业控制中心,减少 7 500 多名服务保障工程师的出勤次数达 60%以上.

用户表明:通过对工程机械易损件消耗特征进行统计,并利用远程监测技术结合定期维修计划预测关键易损件(例如泵管、切割环、眼镜板、输送缸等)的消耗和备件需求,减少呆滞库存 10%以上,每年可以有效节省呆滞库存约 9 300 万元.通过对 X 产品的历史工况数据进行分析,可帮助各个专业研究院寻找品质问题的产生原因,减少破坏性寿命实验所需投入的成本 20%.

5.2 面向天气预报的气象大数据应用示范

通用大数据平台技术是支持专业应用的基础平台,是支撑领域核心业务、优化服务质量的基础.为解决气象领域数据类型多、规模大、访问延迟低、业务逻辑复杂等数据管理难题,我们基于本文的一体化平台开发了专业化、定制化、一体化的气象大数据管理系统,针对气象大数据的特点提供了优化的数据解析、导入、存储和计算内核(如图 5 所示).

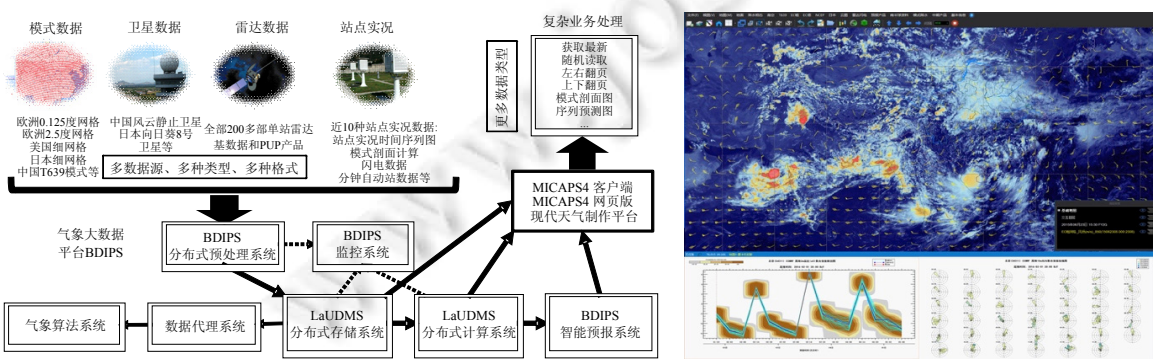

Fig.5 Meteorological big data synthesis processing framework

图 5 气象大数据综合分析处理系统架构

为解决气象预报中的大数据问题,采用跨层优化的先进软件工程理念,研制了气象大数据的近实时数据分

析与处理平台.平台对气象数据的存储组织进行了优化,提升了数据检索效率,同时能够兼容原有存储系统;通过一体化数据处理流程,提升了数据使用效率;通过对标准格式的支持,简化了数据处理流程;通过分布式的数据处理,提升了数据分析速度;通过按地域对数据切片,提升了数据的传输效率;通过分布式存储,提升了数据加载及传输的速度;针对气象数据多维度的特性,进一步演化出多维数据空间模型,进一步提高了系统性能.

针对气象实时数据多类型、高维度、弱模式的特性,我们利用多源异构数据的自由表模型,实现全部海量气象数据从文件系统至非结构化分布式数据库的迁移,采用弹性可扩展服务端架构,应对海量高并发行业及公众用户对于气象实时数据的访问.采用气象大数据分布式实时流式解析技术,实现数据产生即可见的高速加工流水线,同时给出了气象大数据分布式存储解决方案,确保全部数据毫秒级写入与查询.系统服务器端强大的容灾备份能力,高度可靠性保障,完备的系统监控,自动化、智能化的数据处理流程.该系统目前已应用到国家、省、市三级,支撑每日天气预报业务.新系统具有高稳定性、高容错性和良好的可扩展性,支持海量、多类型气象数据的快速解析、导入、存储和访问,数据规模峰值达 16TB/天,数据检索速度达到毫秒级.

6 总 结

在大数据时代,软件系统和工程面临的机遇挑战体现在互为依赖的两方面:一方面,软件系统与工程应针对大数据处理的需求,研究如何开发支持大数据处理各个环节的软件技术与系统,形成面向大数据的软件工程——面向大数据生命周期的一体化集成设计开发环境;另一方面,软件系统与工程实施过程中会涉及大量具有大数据特征的系统运行过程数据,因此有必要对这些多维数据进行充分的关联挖掘和机器学习,发现数据驱动的开发和运行规律,形成大数据的软件工程方法学,指导大数据软件系统的开发——面向软件生命周期的大数据应用系统运行分析工具.

大数据应用系统是个万花筒,覆盖数据的采集提取、存储、计算、分析、可视化等大数据全生命周期的多个技术环节,而各个环节都涉及多种解决方案,涉及到的各类系统有几百种之多,这给面向领域的大数据应用系统构建带来了极大的挑战.图灵奖得主 Stonebraker 提出了“one size does not fit all”的理念[15],认为大数据就应该面向特定领域和问题进行定制和优化.然而,这必然导致大数据生态圈的碎片化,特别是在大数据技术从消费互联网向产业互联网渗透过程中,开发人员从复合型极客转换为产业领域型人才,凸显出大数据应用系统选型困难、系统配置难以预设、维护管理代价大等难题.

本文研究了面向大数据生命周期的软件工程方法学和面向软件生命周期的大数据软件系统,特别是两者之间的本质依赖关系,通过对大数据系统的开发与运行的共性模式进行抽象,提出了一体化开发运行平台框架,具体包括两个方面:第一,研制一体化集成设计开发环境,支持高抽象层次的、面向领域的需求描述框架,具有用户友好的可编程性、并能描述复合业务需求的过程模型;第二,研制了大数据应用系统运行分析工具,通过有效处理、分析软件运行生命周期中生成的运行数据,如运行日志、系统配置、部署脚本、性能指标等,从中提取有用信息、做出优化决策,帮助软件开发者以数据驱动方式进行大数据应用软件的运行优化.

References:

[1] Jagadish HV, Gehrke J, Labrinidis A, Papakonstantinou Y, Patel JM, Ramakrishnan R, Shahabi C. Big data and its technical challenges. Communications of the ACM, 2014,57(7):86−94.

[2] Beatty J, Wiegers K. Forward thinking for tomorrow’s projects requirements for business analytics. Seilevel Whitepaper, 2015.

[3] Chen H, Chiang RHL, Storey VC. Business intelligence and analytics: from big data to big impact. MIS Quarterly, 2012,36(4): 1165−1188.

[4] Chung L, Nixon BA, Yu E, Mylopoulos J. On non-functional requirements in software engineering. LNCS 5600, Springer-Verlag, 2012. 363−379.

[5] Computing Community Consortium, Computing Research Association. Challenges and Opportunities with Big Data: A Community White Paper Developed by Leading Researchers Across the United States. White Paper, 2012.

[6] Hu H, Wen Y, Chua TS, Li X. Toward scalable systems for big data analytics: A technology tutorial. Access, IEEE, 2014,2:652−687.

[7] Huang XD, Wang JM, Zhong Y, Song SX, Yu PS. Optimizing data partition for scaling out NoSQL cluster. Concurrency and Computation: Practice and Experience, 2015,27(18):5793−5809.

[8] Kambatla K, Kollias G, Kumar V, Grama A. Trends in big data analytics. Journal of Parallel and Distributed Computing, 2014,74(7): 2561−2573.

[9] Khouri S, Bellatreche L, Jean S, Ait-Ameur Y. Requirements driven data warehouse design: We can go further. In: Proc. of the Int'l Symp. on Leveraging Applications of Formal Methods, Verification and Validation. Berlin, Heidelberg: Springer-Verlag, 2014. 588−603.

[10] Long M, Wang J, Sun J, Yu PS. Domain invariant transfer kernel learning. IEEE Trans. on Knowledge and Data Engineering, 2015, 27(6):1519−1532.

[11] Long M, Wang J, Cao Y, Sun J, Yu PS. Deep learning of transferable representation for scalable domain adaptation. IEEE Trans. on Knowledge and Data Engineering, 2016,28(8):2027−2040.

[12] Manyika J, Chui M, Brown B, Byers AH. Big data: The next frontier for innovation, competition, and productivity. Report, McKinsey Global Institute. 2011.

[13] Russom P. Data analytics. TDWI Best Practices Reports, Fourth Quarter. 2011. 1−35.

[14] Song S, Zhang A, Wang J, Yu PS. Screen: Stream data cleaning under speed constraints. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2015. 827−841.

[15] Stonebraker M, Cetintemel U, Zdonik S. The 8 requirements of real-time stream processing. ACM SIGMOD Record, 2005,34(4): 42−47.

[16] Wang J, Song S, Lin X, Zhu X, Pei J. Cleaning structured event logs: A graph repair approach. In: Proc. of the 31st Int'l Conf. on Data Engineering (ICDE). IEEE, 2015. 30−41.

[17] Zhang QL, Li SL, Li ZH, Xing YJ, Yan Z, Dai YF. CHARM: A cost-efficient multi-cloud data hosting scheme with high availability. IEEE Trans. on Cloud Computing, 2015,3(3):372−386.

王建民(1968－),男,吉林磐石人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据与知识工程,软件工程.