计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(11):60−66 [doi: 10.15888/j.cnki.csa.006047]

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

银行卡跨行交易系统群的故障分析方法①

周继恩, 尹祥龙, 魏　丹

(中国银联股份有限公司 技术开发中心, 上海 201201)

摘　要: 当前银行卡跨行交易系统群复杂庞大, 如何在交易系统群产生的海量数据中快速定位交易日志和分析交易失败原因对于支付系统的运营维护而言显得尤为重要. 本文提出了一种交易日志切片的分析方法, 该方法将接入用户从交易开始至交易结束整个流程的日志进行切片, 主要利用 Hadoop 框架等相关技术进行大数据分析, 帮助定位交易的故障信息以及提供对应的解决方案. 采用上述方法实现了一个故障分析系统, 实验结果表明: 本方法可以明显提高交易日志查询和交易故障分析的效率, 降低银行卡跨行交易系统群运营成本.

关键词: 交易日志; 故障分析; 日志切片; 自助诊断

引用格式: 周继恩,尹祥龙,魏丹.银行卡跨行交易系统群的故障分析方法.计算机系统应用,2017,26(11):60–66. http://www.c-s-a.org.cn/1003-3254/6047.html

Fault Analysis Method of Interbank Transaction System

ZHOU Ji-En, YIN Xiang-Long, WEI Dan

(Technology Development Center, China UnionPay, Shanghai 201201, China)

Abstract: The current interbank transaction system is huge and complicated. How to quickly locate and analyze the cause of transaction failure from the massive amount of logs generated by the transaction system is very important. This paper proposes an analysis method based on the transaction log slice, which cuts the whole log from the start to the end of a transaction into slice. Then it leverages Hadoop framework and executes big data analysis with predefined scripts, which can locate the error information and find the solution more rapidly. A fault analysis system is implemented with this method. The experiments show that the method proposed in this paper can obviously improve the efficiency of transaction log query and fault analysis, which will reduce the operational cost of interbank transaction system.

Key words: transaction log; fault analysis; log slice; self diagnosis

随着互联网支付的推广与普及, 越来越多的人选择通过各种各样的支付工具进行支付. 但是由于支付途径的丰富性、支付网络的差异性、支付场景的多样性、支付操作的随机性, 支付过程会出现各种各样的问题或故障. 对于大型支付交易系统而言, 一笔交易流转往往要经过多个子系统, 比如子系统 A、子系统 B……子系统 N, 一笔交易在流转过程中的任何子系统都有可能出故障. 故障指的是在交易过程中由于报文格式错误、签名不匹配、无权限、页面过期等造成的非正常应答. 根据某银行卡组织 2015 年 1 月份的统计数据, 在该月产生的近 7000 个服务单中, 交易查询分析类的服务单以 1000 多的数量排名第三. 并且, 在所有超期未完成的服务单中, 交易查询分析类的超期服务单远超其他类别位列第一, 而交易查询分析的服务单往往是在交易中出现标识交易失败的应答, 即用户理解的故障. 以上表明, 为保证尽快解决用户的交易故障问题, 非常需要一个快速便捷的交易查询和生产事件分析模型对复杂系统进行故障分析.

① 收稿时间: 2017-02-16; 修改时间: 2017-03-02; 采用时间: 2017-03-16

在研究领域, 关于系统故障的动态检测诊断主要包含故障预测、故障检测、故障分析等方面. 现有的故障诊断方法主要有三种: 定性分析的方法[1-3], 定量分析的方法[1,4,5]以及基于历史分析的方法[6,7]. 定性的分析方法主要通过图形、仿真、知识库等的方式得到故障的通用分析方法, 有助于故障预测和分析故障出现趋势; 定量的分析则是通过数学模型计算、数据驱动等方式, 由具体的数据特征分析出故障的原因或发展趋势; 而基于历史分析则通过历史记录分析故障原因. 通常情况下在使用某种方法进行故障分析与诊断时也可能会运用到其他两类方法来进一步提升分析能力与准确性. 本文提出的故障分析方法主要结合了定性分析与基于历史分析两种方法, 通过分析模块进行定性分析, 并通过历史知识库模块进行基于历史的分析, 有效提高故障分析效率.

在实际应用领域, 当用户在交易出现问题时, 会通过交易平台在线提交问题, 然后由交易平台维护人员根据用户提供的交易信息, 比如订单号等信息, 首先定位出交易可能在哪个子系统出现问题, 然后从该子系统中获取用户的交易日志信息进行人工分析, 并将分析结果反馈给用户. 该方式主要通过维护人员进行人工查找大量日志进行交易故障分析, 不仅效率低下, 而且分析结果只涉及到部分交易子系统, 很难做到对交易在各个子系统中的日志进行全面分析, 而且分析的过程耗费较多时间、占用大量人力资源.

本文将接入用户从交易开始至交易结束整个流程的日志进行切片[8], 为支付系统中每一笔交易分配唯一标识 token(关联 key 值), 可更加全面地获取用户在支付系统中交易信息, 为高效对用户交易日志分析提供了便利, 更能方便用户快速查找交易的错误信息和解决方案.

1 系统概述

1.1 系统框架

本文提出的针对多系统支付交易的故障分析方法, 主要包含六个模块: 数据转移模块、大数据分析模块、二次分析模块、知识库模块、分析结果存储模块、交易日志查询模块, 系统总体框图如图 1 所示.

数据转移模块负责将分布在各个子系统中的日志文件收集整理, 转移给大数据分析模块, 供其进行后续分析. 大数据分析模块按交易唯一标识 Token 提取出日志, 按照一定的规则分析出不同子系统的日志, 并依据每一笔交易的唯一标识 Token, 确定不同子系统同一笔交易的传递方式, 然后将分析结果依据交易的唯一标识 Token 存入分析结果存储模块. 分析结果存储模块存放大数据分析结果并供交易日志查询模块进行查询分析. 用户可以输入交易相关的查询条件 (比如交易唯一标识 Token、订单号、商户代码等), 通过交易日志查询模块对分析结果存储模块存放的交易诊断信息进行查询, 获取该交易的信息和详细报文, 并可分析出错交易的故障信息以及对应的解决方案; 同时, 由于大数据分析模块对交易日志进行统一分析, 对于个别可以进行特殊分析的交易日志信息, 我们通过二次分析模块完成, 当用户进行查询的过程中, 二次分析模块对未进行二次分析的交易日志信息进行处理, 如果该交易日志信息可以进行特殊分析, 二次分析模块直接进行分析, 而对于不能进行特殊分析的交易日志信息查询知识库模块获取故障原因.

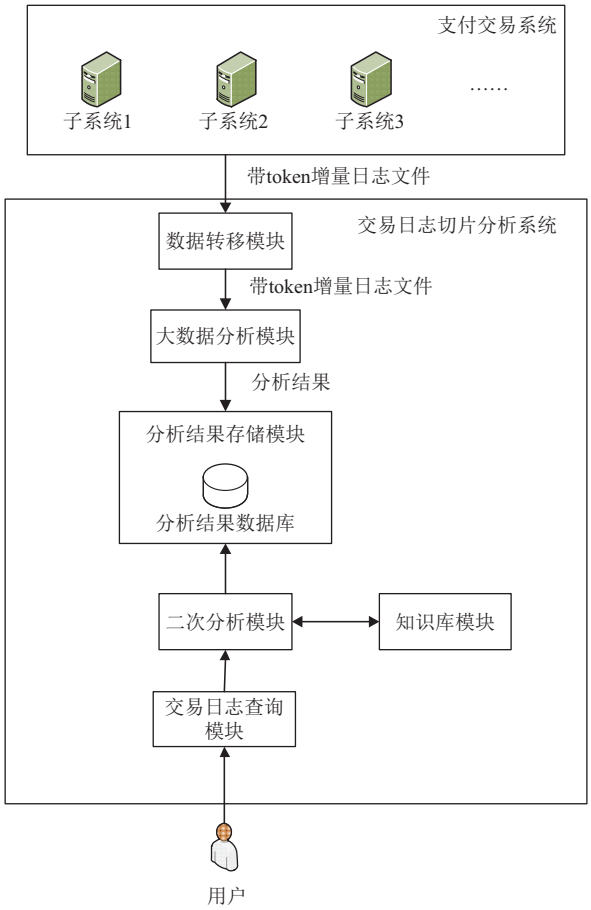

图 1 系统总体框图

1.2 实施前置条件

进行本文提出的方法验证前, 需要交易经过的各子系统对交易日志的输出进行改造, 为每一笔交易分配唯一标识 token, 放置在报文域中, 用于各子系统之间的交互. 该 token 的值生成规则如下所示:

[4 位子系统代码+14 位时间 (YYYYMMDD HHMISS)+12 位序列号]

子系统代码是指交易系统群为每个子系统分配的唯一标识. 序列号的组成规则不强制所有系统一致, 各系统保证同一秒内唯一即可. 参考构成: 1 位中心编号+2 位主机编号+9 位顺序号. 例如“102123456789”, 表示中心 1 的主机 02 产生的统一标识.

交易唯一标识 token 的产生、传递、删除、流转方式如下.

(1) 产生: 统一标识在入口子系统产生. 与外部系统、终端、网页等 (非本系统群) 相连, 交易进入本支付系统的第一个子系统为入口子系统.

(2) 传递: 入口子系统产生统一标识后, 透传给所有涉及的此交易联机处理接收子系统, 所有子系统都记录数据库交易流水表, 失败时在报错的应用日志中也记录.

(3) 删除: 为了不影响外部系统, 交易发送给非本系统 (外部系统) 时, 最后一个子系统将统一标识删除.

(4) 流转: 交易应答入口子系统从本子系统交易流水中将 token 取出, 返回给后续处理子系统.

2 各模块的设计与实现

2.1 数据转移模块

在交易量较高的实际生产环境, 每一秒都产生很高的交易量, 生成大量的交易日志, 数据转移模块用于将支付平台各子系统增量交易日志取出并用于数据处理模块进行分析诊断[9]. 该模块以一个可调时间 (比如 1 分钟) 为时间片, 轮询支付平台各子系统的交易日志文件, 并将增量日志数据转移到大数据分析模块. 为了提升数据转移的效率, 且尽可能减小对实际交易支付平台的影响, 数据转移模块仅将增量日志进行转移, 发送方和接收方数据转移伪代码如表 1 和表 2, 以上一时间片终点的时间游标作为起点.

如上述伪代码所示, 对于发送方而言, 我们以上一时间片读取结束位置开始, 获取所有以此刻开始的一个时间片内新增的交易日志, 并发送给接收方; 接收方接收发送过来的日志记录, 并存放到相应的日志文件中.

表 1 数据转移发送方伪代码

```
BEGIN
readStartTs /*本次读取时间片开始时间*/
readEndTs /*下次读取时间片开始时间*/
timeSlice /*时间片*/
readEndTs ← readStartTs + timeSlice
transDict ← 新建日志文件字典, key为日志文件名, value为该文件一个时间片内更新的日志记录list /*遍历文件目录*/
FOR var file IN fileList DO
  IF file.更新时间 >= readStartTs and file.更新时间 < readEndTs then
    /*读取该时间片内更新的记录内容*/
    transList.add (该时间片内更新的记录内容)
  ENDIF
END
send transDict to日志处理服务器
readStartTs ← readEndTs
END
```

表 2 数据转移接收方伪代码

```
BEGIN
var transDict ← 实际交易系统传送的日志记录字典
FOR var entry IN transDict DO
  File file ← new File(entry.key, “a”) /*以追加的方式打开文件*/
  /*将entry.value对应的内容追加到file中*/
  OutputStream os ← new BufferWriter(file)
  os.write(entry.value)
  os.close()
END
END
```

2.2 大数据分析模块

大数据分析模块主要功能是将数据转移模块转移过来的实际交易日志, 借助 Hadoop 框架进行大数据分析. Hadoop[10]是一个分布式系统基础架构, 它实现了一个分布式文件系统 (Hadoop Distributed File System), 简称 HDFS[11]和 MapReduce[12]编程模型用于大数据量的计算[13]. 由于 HDFS 有着高容错性 (fault-tolerent) 的特点, 并且设计用来部署在低廉的 (low-cost) 硬件上, 而且它提供高传输率 (high throughput) 来访问应用程序的数据, 可以流的形式访问文件系统中的数据. MapReduce 是一种简化的并行计算的编程模型, 包含两个核心操作: Map 和 Reduce. Map 是把一组数据一对一的映射为另外的一组数据, 其映射的规则由一个函数来指定, 比如对[1, 2, 3, 4]进行乘 2 的映射就变成了[2, 4, 6, 8]. Reduce 是对一组数据进行归约, 这个归约的规则由一

个函数指定, 比如对[1, 2, 3, 4]进行求和的归约得到结果是 10, 而对它进行求积的归约结果是 24.

本文借助 HDFS 的分布式特性和 MapReduce 对大数据的并行计算, 对日志进行分析. 因为各个子系统的日志规则不完全一样, 我们为每一个子系统日志分析创建一个任务, 为每一个任务分配 Hadoop 执行任务的资源, 每个任务有其针对对应日志的分析规则, 所有的分析规则最终都将日志以纵向切片的方式切分出有效的数据项, 然后再将不同规则切出来的数据进行归约. 这就是 MapReduce 的过程, MapReduce 通过设定正则切片规则对日志文件进行 map 操作, 再通过 reduce 操作将拆分后的日志记录通过并集函数进行归约, 其中, HDFS 主要负责分布式访问文件系统中的日志文件. 如图 2 所示的流程图.

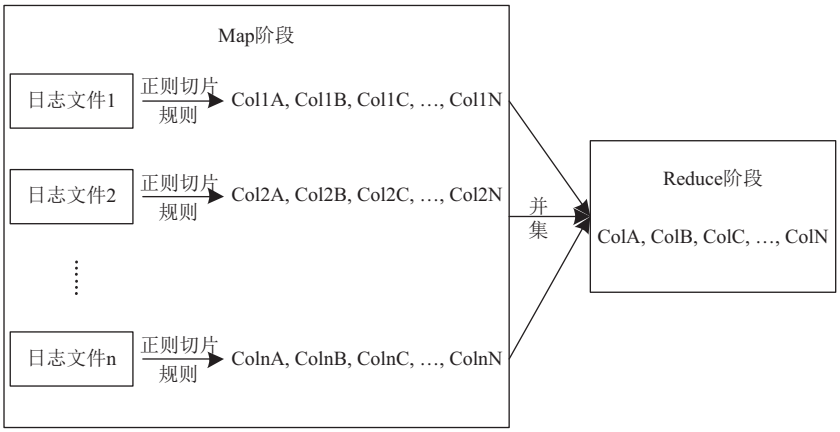

图 2 大数据分析模块流程图

2.3 二次分析模块

通过 2.2 节大数据分析模块的通用处理解析, 我们得到的是结构化的数据, 这些数据的分析模型和分析框架一致, 同一类别得到的结果也是具有相同的结构. 但是, 一些类别的日志数据还可进行更深层次的分析, 得出更详细的交易信息, 并能解读故障交易的原因. 二次分析流程伪代码如表 3 所示. 分析结果存储模块设有标识该交易信息是否已经进行二次分析的标识位, 对于二次分析完成的交易信息将跳过二次分析模块, 反之, 二次分析模块将会执行, 其主要功能就是将可进行特殊分析的交易信息再次分析, 剖析交易的详细流程, 并分析出有问题交易故障内容详细描述, 对于不可进行特殊分析的故障交易可查询知识库模块获取通用交易分析信息, 同时获取故障交易对应的解决方案.

二次分析模块采用脚本执行的方式对交易日志进行深度分析, 脚本执行过程如图 3 所示, 分析伪代码如表 4 和表 5 所示. 当二次分析的流程启动时, 主应用会调用脚本执行. 首先进入统一分发的 dispatch 脚本, 该脚本负责识别交易信息, 并将交易信息依次分发下去, 遇到匹配交易类型的脚本后, 该脚本将会对交易信息详细分析, 分析结果传送给 dispatch 脚本. 我们会按照交易日志类型进行分类, 对每一种类型的交易日志提供一种分析脚本, 随着交易日志类型的增多, 脚本的数量也是可以改变的.

值得一提的是, 二次分析模块的触发采用惰性计算[14]的思想, 仅在必要时, 即用户有需求时, 才会触发二次分析的流程. 通过这种方式, 减少了冗余计算的浪费, 提升了框架的有效性.

表 3 二次分析流程伪代码

```
BEGIN
analyseRslt /*存放分析后的结果*/
DO获取交易记录信息
IF未二次分析THEN
  IF可进行二次分析THEN
    DO二次分析
    analyseRslt ← 二次分析结果
  ELSE
    DO查找知识库, 获取通用分析
    analyseRslt ← 知识库通用分析结果
  ENDIF
  DO结果analyseRslt写入用于存储分析结果的数据库
  DO更新交易信息状态为已进行二次分析
ENDIF
END
```

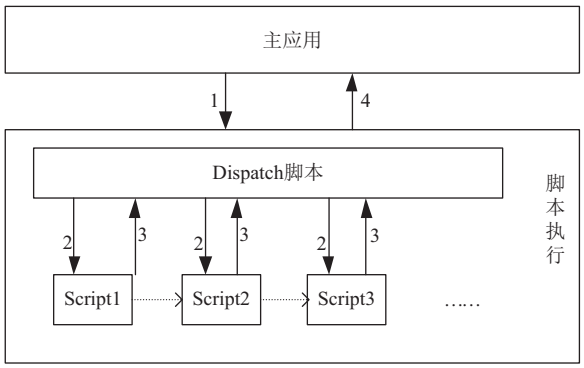

图 3 二次分析之脚本分析

表 4 Dispatch 脚本伪代码

```
BEGIN
logText /*必输值: 日志文本*/
fileName /*选输值: 文件来源名称*/
scriptList /*脚本list, 默认顺序*/
IF filename为空THEN
  DO按照scritList中列举的脚本顺序执行, 匹配日志规则成功则退出, 返回分析结果; 匹配失败, 返回null, 直接结束.
ELSE
  DO根据fileName匹配脚本名称, 找到匹配脚本, 用此脚本进行解析, 返回分析结果; 找不到匹配脚本, 返回null, 直接结束.
ENDIF
End
```

表 5 脚本执行伪代码

```
BEGIN
SCRIPT_KEYWORD_1 /*脚本关键字1*/
SCRIPT_KEYWORD_2 /*脚本关键字2*/
……
SCRIPT_KEYWORD_3 /*脚本关键字3*/
IF日志logText满足脚本关键字THEN
  DO用该脚本的规则继续解析该日志logText
  RETURN解析结果
ELSE
  该脚本无法解析该日志logText
  RETURN null
ENDIF
END
```

2.4 知识库模块

知识库模块存储了各类交易类型的通用分析方法, 错误应答码及对应的解决方案. 当交易类型不在进行特殊分析的范围内时, 知识库模块为故障类型提供以上故障的通用分析信息, 比如交易的订单号, 交易时间, 应答信息, 错误分析与解决方案等.

2.5 分析结果存储模块

分析结果存储模块用于将交易日志分析结果信息存储在数据库中. 我们的数据库采用关系型数据库, 双机主从备份, 按照日期分表存储, 数据库中存储了最近 31 天的交易日志结果并持续更新, 每天的交易信息会替换上一月同一天的交易信息. 交易日志的更新频率可根据用户在数据转移模块设定的数据转移间隔时间片而定. 采用这种存储模式, 有效的保证用户的请求得到快速查询响应, 提高了热点数据的命中率, 便于不间断保持数据库的热度.

2.6 交易日志查询模块

交易日志查询模块为用户提供远程交易日志的 Web 查询页面, 用户整个支付过程在交易系统发生的交易信息, 可以通过该模块查询出来. 该模块采用 Spring MVC 编程框架[15]搭建一个 web 平台, 由于分析结果存储模块以交易的关联 key 值为标识进行存储的, 所以交易日志查询模块以交易关联 key 值的关键信息 (如 logId、商户号、订单号、卡号等) 为查询条件. 用户输入交易的关键信息, 查询出该条件相关的结果列表, 获取的分析信息来源于分析结果存储模块, 结果列表中展示了交易的基本信息 (如订单号, 交易时间, 卡号, 商户号, 应答信息等). 对于发生故障的交易信息, 用户点击诊断, 可以触发二次分析流程, 具体操作流程参考 2.3 节, 分析之后得到故障分析结果信息, 包含故障原因码、故障原因描述、故障解决方案等.

3 效果分析与展示

用户输入交易的关联要素, 查询模块从数据存储中查找到相应的结果, 交易日志信息查询效果如图 4 所示. 对于交易的故障分析, 我们点击图 4 中所示的诊断按钮, 调用 2.3 节中的二次分析模块进行分析, 结果如图 5 所示.

通过实验验证, 本文所设计的框架, 大大地提高了相关人员交易日志查询和交易故障分析的效率, 节省了处理用户查询分析请求的人力, 但是由于机器分析日志相对于人工分析, 范围和方法有限, 部分日志的分析方法有待提升, 因此准确度有所下降, 如表 6 所示.

通过以上分析和数据展示, 本方案具有以下几个优势: 准确找出热点数据; 提前缓存, 快速响应用户请求; 适用于大规模复杂系统群的自助分析.

4 结语

实际投产环境中, 支付系统中的交易往往具有随机性、突发性、高复杂性, 本文从多方位分析, 借助大

数据分析技术进行日志通用切片, 再对日志类型进行分类, 有针对性的进行特殊分析, 同时采用知识库的方式对日志进行通用分析. 综合运用大数据处理相关技术、惰性计算、web 框架等技术, 通过自动化的方式对日志进行分析为相关运维人员提供了极大的便利, 大大缩短了交易故障的排查时间, 节省了相关人力.

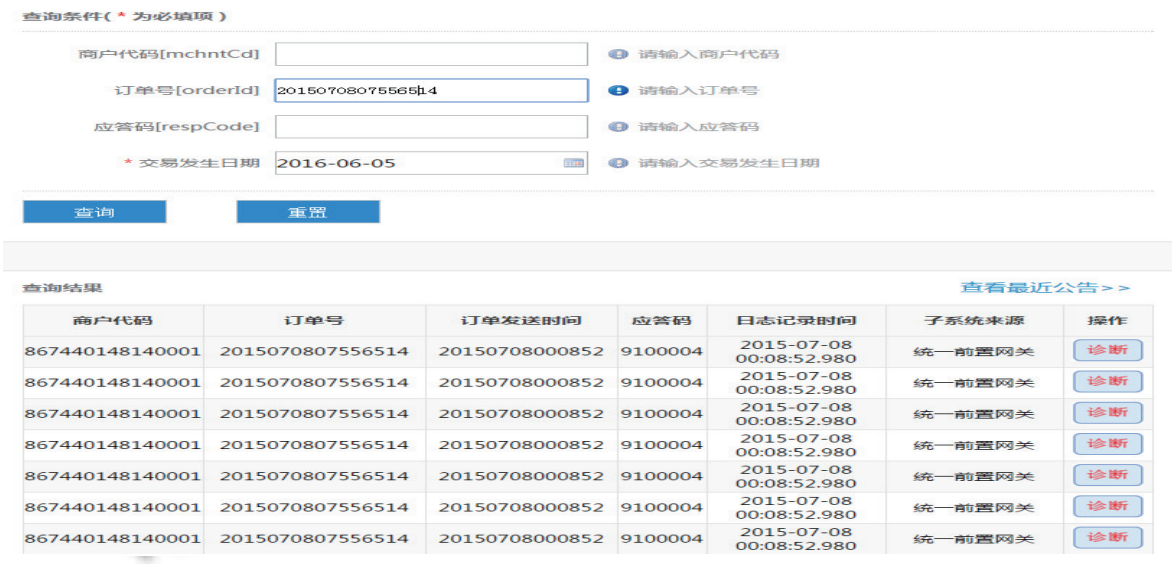

图 4 交易日志信息查询效果图

图 5 交易故障分析图

表 6 设计方案使用前后效果对比

指标	使用前	使用后
交易日志平均查询时间(分钟)	30	1
交易故障平均分析时间(分钟)	120	5
人力需求(人)	5	1
准确性(%)	95	90

参考文献

1 周东华, 胡艳艳. 动态系统的故障诊断技术. 自动化学报, 2009, 35(6): 748–758.

2 Li KT, Reichenbach C, Csallner C, *et al*. Residual investigation: Predictive and precise bug detection. Proc. of the 2012 International Symposium on Software Testing and Analysis. Minneapolis, MN, USA. 2014. 298–308.

3 Liu C, Yan XF, Fei L, *et al*. SOBER: Statistical model-based bug localization. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 286–295. [doi: 10.1145/1095430]

4 Xu W, Huang L, Fox A, *et al*. Detecting large-scale system problems by mining console logs. Proc. of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. Big Sky, Montana, USA. 2009. 117–132.

5 Wang DW, Arogeti S, Zhang JB. Causality assignment and model approximation for quantitative hybrid bond graph-based fault diagnosis. IFAC Proc. Volumes, 2008, 41(2): 10522–10527. [doi: 10.3182/20080706-5-KR-1001.01782]

6 Zhou J, Zhang HY, Lo D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. Proc. of the 2012 34th International Conference on Software Engineering (ICSE). Zurich, Switzerland. 2012. 14–24.

7 Lo D, Cheng H, Han JW, *et al*. Classification of software behaviors for failure detection: A discriminative pattern mining approach. Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Paris, France. 2009. 557–566.

8 章程. 基于机器学习和程序分析相结合的程序调试技术研究[博士学位论文]. 上海: 上海交通大学, 2013.

9 姚恒, 薛质. 基于增量日志的数据复制. 信息安全与通信保密, 2007, (6): 176–178.

10 Shvachko K, Kuang H, Radia S, *et al*. The hadoop distributed file system. Proc. of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). Incline Village, NV, USA. 2010: 1–10.

11 Borthakur D. HDFS architecture guide. Hadoop Apache Project. http://hadoop.apache.org/common/docs/current/hdfsdesign.pdf. 2008: 39.

12 Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters. Proc. of the 6th Conference on Symposium on Opearting Systems Design & Implementation. San Francisco, CA, USA. 2004. 10.

13 Chu CT, Kim SK, Lin YA, *et al*. Map-reduce for machine learning on multicore. Proc. of the 19th International Conference on Neural Information Processing Systems. Cambridge, MA, USA. 2006. 281–288.

14 Launchbury J. A natural semantics for lazy evaluation. Proc. of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. Charleston, South Carolina, USA. 1993. 144–154.

15 Fu PJ, Du ZJ. Application of spring in realizing MVC framework. Computer Technology and Development, 2006, 16(6): 236–238, 241.