软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(6):1285-1305 [doi: 10.13328/j.cnki.jos.004823]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件配置错误诊断与修复技术研究*

陈 伟[1], 黄 翔[2], 乔晓强[1], 魏 峻[1,3], 钟 华[1,3]

[1](中国科学院 软件研究所 软件工程技术研发中心,北京 100190)

[2](中山大学 信息科学与技术学院,广东 广州 510006)

[3](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

通讯作者: 陈伟, E-mail: wchen@otcaix.iscas.ac.cn

摘 要: 软件的多样性、复杂性、灵活性和高度可定制性对系统的正确配置提出了挑战,配置错误已经成为影响应用服务质量的关键问题之一.很多学者和研究机构致力于配置错误的检测、诊断和故障修复的相关技术和方法研究,以提高复杂应用系统的可用性和可靠性.为系统了解软件配置错误相关的研究现状和进展,建立了一种多方面、多角度的分析框架对该领域的主要研究工作进行分类总结和分析评价,该分析框架覆盖了方法类型、方式和适用范围这 3 个方面的多个角度.基于该分析框架的分析结果,总结了当前软件配置错误相关研究中存在的问题,并针对今后该领域的研究趋势进行了展望,对继续和深入研究具有一定的指导意义.

关键词: 软件配置;故障诊断;错误修复

中图法分类号: TP311

中文引用格式: 陈伟,黄翔,乔晓强,魏峻,钟华.软件配置错误诊断与修复技术研究.软件学报,2015,26(6):1285-1305. http://www.jos.org.cn/1000-9825/4823.htm

英文引用格式: Chen W, Huang X, Qiao XQ, Wei J, Zhong H. Research on software misconfiguration troubleshooting. Ruan Jian Xue Bao/Journal of Software, 2015,26(6):1285-1305 (in Chinese). http://www.jos.org.cn/1000-9825/4823.htm

Research on Software Misconfiguration Troubleshooting

CHEN Wei[1], HUANG Xiang[2], QIAO Xiao-Qiang[1], WEI Jun[1,3], ZHONG Hua[1,3]

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

[2](School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006, China)

[3](Key Laboratory of Computer Sciences, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Software configuration has been a big challenge due to the diversity, complexity, flexibility and customizability of a system. Configuration error has become a dominant cause leading to system failure and service outage. To improve software availability and reliability, many researchers and institutions have devoted their efforts on software misconfiguration troubleshooting. This paper first builds an analytical framework with establishment of 3 aspects and multiple perspectives, covering the method type, style and applicability. Based on this framework, the paper provides an overview of the state of art of misconfiguration troubleshooting along with analysis on the current leading methods of software misconfiguration troubleshooting. Finally, this paper summarizes the shortcomings in the current research and outlinesthe development prospects of future research. This paper aims to provide some available information and beneficial insight for future researches.

Key words: software configuration; failure diagnose; error fix

配置(configuration)是软件系统不可或缺的组成部分,广泛存在于软件部署、升级以及迁移等应用场景中.配置覆盖了软件系统及其环境的多个方面,包括配置文件、已安装的软件,甚至包括底层硬件环境.广义上来讲,

* 基金项目: 国家自然科学基金(61402453); 国家高技术研究发展计划(863)(2013AA041301); 国家科技支撑计划(2013BAH05F03, 2012BAH14B02)

收稿时间: 2014-05-13; 修改时间: 2015-01-05, 2015-01-21; 定稿时间: 2015-02-06

软件配置是指以用户需求和软件的功能、结构及主要特性等为依据,选择和确定相关硬件、软件和固件的型号、版本及数量,规划软件放置位置和关联关系,设置软件系统相关参数值等;狭义上的软件配置主要是指对软件系统配置参数取值的设置.

随着计算机技术和 Internet 的快速发展,软件系统在实现技术、应用模式和系统规模等方面发生了巨大变化,呈现网络化、协同化、普适化和服务化的趋势,软件应用的平台开放、动态、分布等特点也日益显著.复杂应用(尤其是复杂网络应用)往往通过大量配置参数提高系统的可定制性,通过不同的配置参数取值获得期望的系统功能和性能特性.软件配置往往决定了诸如数据存储位置、服务器端口绑定以及使用哪些模块和算法等多方面的问题,但是大量配置参数的存在及使用的灵活性为应用的正确配置带来极大困难[1],为系统后续运行的可靠性、可用性以及系统性能等带来极大挑战,具体表现如下:

(1) 配置错误已成为导致系统异常、服务失效甚至是系统崩溃的显著原因之一.基于对 3 个商业互联网服务的研究,Oppenheimer 等人[2]发现,50%以上的系统运维管理故障源于配置错误,并导致系统服务无法使用;Gray[3]的研究发现,42%的系统宕机与运维管理相关,其中,软件、硬件和环境错误分别占 25%,18%和 14%;还有相关研究发现,80%以上的网络故障也都是由于配置错误引起的[4].

(2) 配置错误会带来灾难性后果,严重影响应用的服务质量,带来经济和声誉等方面的负面影响.近年来,包括 Microsoft Azure[5],Amazon EC2[6]和 Facebook[7]在内的很多复杂网络服务或应用系统都经历过配置错误导致的运行中断,使数百万用户受到影响.2010 年,Facebook 的一个配置错误使多达 500 万用户在几个小时内无法访问其网站和服务[7];又如在 2009 年,由于 DNS 的配置错误,导致互联网的整个“.se”域在一个多小时内无法正常运行,使大约 100 万台主机受到影响[8].

本文首先给出软件配置错误定义,归纳总结了导致配置错误的主要原因,界定了软件配置错误诊断与修复的问题域,并对其相关研究领域进行了概述.其次,从方法类型、特征和适用范围这 3 个方面的多个角度建立了配置错误诊断与修复方法的分析框架.然后,简要介绍了该领域的研究现状和代表性工作,并基于分析框架对当前的研究工作进行分类总结和分析评价.最后,本文分析总结了当前软件配置错误的诊断研究中存在的问题,并对未来研究趋势进行展望.

1 概 述

软件配置错误(configuration error 或 misconfiguration)定义为由于软件配置项的值设置错误而导致系统运行产生确定性(deterministic)故障,并产生相应的错误信息[9].因此,配置错误可以认为是软件系统中存在的这样一类问题,即:软件代码本身正确,但是由于不正确的安装、配置或系统升级而导致软件系统无法按照预期正确运行[10].导致软件配置错误的主要原因可以归纳为以下几点:

(1) 应用规模庞大且配置参数众多,需要大量的时间和精力进行配置方法的学习和参数用途的理解.复杂应用常提供大量的配置参数项来满足用户对系统功能和性能相关特性的定制需求,增强系统的灵活性和可定制性.大规模复杂应用往往由上层应用服务和下层的系统支撑软件组成,包括 Hadoop[11],FreePastry[12],HBase[13]在内的系统软件均具有数量众多的配置项(例如,Hadoop 具有超过 400 个配置项)[1].随着云计算的发展和集群技术的运用,系统规模和参数配置项随着结点数量成倍增长.

(2) 应用组件存在交互和相互依赖,导致配置参数间存在关联,给应用配置及其错误诊断带来困难.复杂应用运行在多种/多层中间件平台上,分层架构使组件之间、容器之间以及组件与容器之间存在关联和依赖,也使应用及环境的配置参数间存在关联和约束,例如配置参数取值的一致性、依赖性、取值范围的相互制约等.如果忽略这些关联和约束,将极易发生配置约束违背,产生系统运行故障.

(3) 软件系统配置过程在一定程度上依赖于系统管理人员的知识和经验.首先,用户使用和配置软件需要花费大量的时间和精力进行学习,研究数据证明,大部分参数配置错误发生在系统的初次使用和配置中[14,15],并且文档与系统的不一致使用户可能获取错误的指导信息[16,17];其次,人为操作导致的参数设置错误难以避免,其中包括拼写错误、语义错误等,长期以来,人工操作一直是导致互联网服务

异常与不可用的显著因素之一[2],而其中一半以上的错误是由人为因素引起的[18,19].

(4) 系统配置错误的辅助诊断能力直接影响诊断效率与准确性,配置错误修复主要依赖于已有经验和外部技术支持.应用故障发生时经常抛出一个泛泛的错误提示,甚至没有任何提示,错误信息内容缺少可用的错误原因、系统状态和错误范围描述,无法从中获取有效的线索.已有研究表明,仅仅只有7.2%~15.5%的配置问题能够给出较为清晰准确的信息提示来支持错误诊断[14].在缺少有效提示信息的情况下,配置错误的定位和诊断花费的时间和精力随之成几倍甚至十几倍的增长.

软件配置错误诊断与修复(software misconfiguration troubleshooting,简称 SMT)的对象是配置错误导致的系统故障和异常,目标是确保应用系统正确运行、保障其可靠性和可用性,核心内容包括配置错误的发现(detect)、诊断(diagnose)和修复(fix).SMT 相关的研究领域主要有自动配置和软件故障诊断.

自动配置方法的主要研究目标是实现配置的自动化执行,减少人工操作,从而降低配置出错几率.当前,自动配置方法研究主要面向底层的网络设备,应对互联网大量的配置需求以及配置复杂度日益增长的趋势.自动配置是互联网配置管理研究领域的一部分,按照流程,可以将自动配置分为配置自动生成[20,21]、配置验证[22,23]和配置自动实现[24].清华大学的李福亮等人[25]对互联网自动配置领域的研究现状和代表工作进行了综述.

软件系统故障诊断方法的主要思想是:利用监测框架收集系统监测数据,在系统正常运行状态下建立状态模型,将监测到的系统状态与通过模型预测出的理想状态进行比较,把检测到的偏差作为可能的故障,据此定位问题出现的原因.软件系统故障检测与诊断的常用方法包括基于规则和信号的方法[26]、基于行为模型的方法[27]、基于度量分析的方法[28]、基于性能模型的方法[29]等.

软件配置错误诊断与修复作为一个细分的研究领域,与上述两个领域既有关联又有区别:首先,与网络设备不同,软件系统复杂多样且配置错误难以避免,因此,软件配置错误诊断主要从如何发现、诊断和修复配置错误的角度来提高系统的可靠性与可用性;另一方面,配置错误是引起系统故障的因素之一,软件配置错误诊断在方法上与系统故障诊断存在相似性,不同的是,前者完全面向配置错误导致的系统故障和异常,更加具有针对性.

软件配置管理(software configuration management,简称 SCM)与 SMT 名称相近,但分属于不同的概念范畴,其问题域和研究内容上有根本性区别.软件配置管理目的是协调软件开发使得混乱减到最小,是一种标识、组织和控制修改的技术.SCM 研究怎样在不同时刻标识软件系统的配置,以便系统化地控制配置的改变,以及在整个软件生命周期内维护配置的完整性和可追踪性,主要包括配置识别、变化控制、状态记录报告以及审计[30].

2 配置错误诊断与修复技术分析框架

SMT 方法的目标系统、故障类型、采用的方法、具有的功能特征以及必须的前提条件和假设等方面各有特点,为了对 SMT 的研究现状进行分析、归纳与总结,本文从多个方面和角度建立了 SMT 的技术分析框架,如图 1 所示.该框架以配置错误诊断与修复的过程为依据,分别从功能、基本方法、前提条件、目标系统和故障类型多个角度进行分析总结,覆盖了方法类型(type)、方法特征(feature)和适用范围(applicability)这 3 个方面.

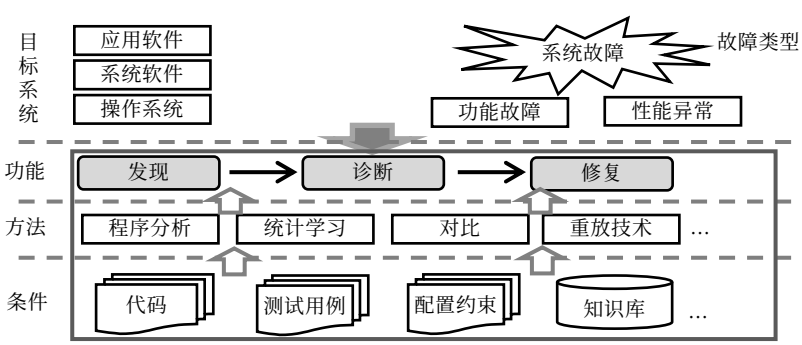


Fig.1 SMT analytical framework

图 1 SMT 分析框架

2.1 方法类型

SMT 从基本方法角度可以分为基于程序分析(program analysis based)的方法、基于统计学习(statistics based)的方法、基于重放技术(replay based)的方法和基于对比(comparison based)的方法以及基于上述两种或多种方法的混合方法,分类如图 2 所示.

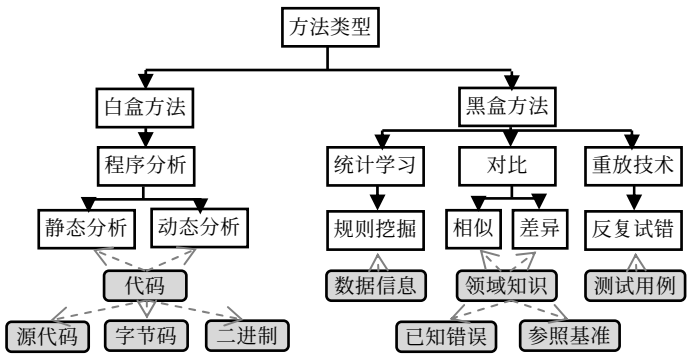


Fig.2 SMT classification

图 2 SMT 方法分类

2.1.1 基于程序分析的方法(program analysis based)

基于程序分析方法的核心思想是:将目标系统作为白盒(white box),采用与软件测试领域相似的技术手段,重点关注系统中与配置参数读写及操作相关的语句,通过数据流和控制流等信息流的分析,较为准确地获取配置参数影响的语句和执行路径,实现配置错误导致的系统异常检测与定位.

基于程序分析的 SMT 可以进一步分为基于静态程序分析(static analysis based)的方法和基于动态程序分析(dynamic analysis based)的方法,如图 2 所示.基于静态程序分析的方法主要对目标系统进行静态数据流分析,大多从配置参数读取位置为起始,获取并分析配置参数访问、使用和修改的相关语句,实现配置错误诊断.动态程序分析方法通过对目标系统代码的植入和运行时监控来获取系统运行时的动态数据流和控制流信息,重点关注和分析配置参数项影响的系统执行路径.

2.1.2 基于统计学习的方法(statistics based)

基于统计学习的 SMT 将目标系统作为黑盒(black box),观察、收集系统行为(behavior)、状态(state)及与事件(event)相关的历史数据,再基于概率统计和机器学习等方法分析挖掘出目标系统的配置参数访问模式及设置规则.当系统运行时发生故障,通过查看当前系统行为或状态违背的相关规则来实现配置错误的检测和诊断.

基于统计学习的方法无需目标系统的源码支持,也不需要理解系统的特定语义信息.但是为了实现配置访问规则挖掘与异常事件检测的准确性,需要大量历史信息作为分析和挖掘的数据基础,适用于有配置数据集中存储管理和大量访问的应用场景.对于使用频率低的配置参数及其配置行为,由于缺少数据的支持,难以挖掘出相应规则,导致此方法难以适用.

2.1.3 基于重放技术的方法(replay based)

基于重放技术的方法本质上是将原先人工执行的反复试错的方法通过计算机的方式加以实现,提高故障诊断和修复的效率.基于重放的配置错误诊断方法在沙箱环境(sandbox)[31]下不断尝试改变目标系统的配置参数设置,并观察系统行为的变化情况,最终实现错误修复.沙箱环境起到很好的系统隔离作用,避免了对于其他系统可能带来的影响;同时,通过支持目标系统状态的回滚,降低改变原有正确配置参数带来的风险.

基于重放技术的方法需要对底层操作系统内核进行修改来构造沙箱环境,因此,该方法对系统运行环境具有定制需求,带来较大的工作量和限制;其次,基于重放的方法还需要一组测试用例来检验修改配置参数前后目标系统的变化.

2.1.4 基于对比的方法(comparison based)

基于对比的方法通常以一组基准(baseline)、样本(sample)或已知故障(known problem)为依据,基准和样本记录了系统的状态[32-34]、签名[35]和事件[36]等,已知故障则记录了发生的系统故障特征及其修复方法[37].以基准和样本为依据时,方法将当前的系统状态、签名或行为与已有的基准进行对比,发现与其中相似基准或样本存在的差异,进而诊断出导致这些差异出现的配置错误;以已知故障为依据时,方法通过对比系统状态或行为找到与当前故障最为相似的已知故障,并将已知故障关联的配置错误作为可能导致当前系统故障的根本原因,同时,将已知故障对应的解决方法作为当前故障的推荐方法.

基于对比的方法同样采用的是黑盒策略,无需进行系统内部程序的分析和理解,但是需要为每个目标程序创建相应的样本库或已知故障库作为比较和分析的基准,在领域知识方面具有较高代价.由于不同目标系统的样本库或已知故障库各不相同,因此缺少广泛的通用性.该方法也主要适用于配置信息集中存储和管理的场景.

2.1.5 混合方法

程序分析、统计学习、重放技术和对比等方法在准确程度、执行效率以及额外开销等方面各有优劣,因此也有将上述多种方法结合在一起使用的 SMT,用不同的方法分别解决其中的部分问题.例如,基于程序分析技术进行可能的配置错误定位,基于统计分析来提高配置错误诊断的准确性.

混合方法融合了不同方法在错误定位和故障诊断方面的各自优势,但其局限性也来源于其中,需要充分考虑如何发挥方法的优势,并尽量限制或避免方法本身的局限性.

2.2 方法特征(feature)

如图 1 所示,SMT 包括了发现、诊断和修复等 3 个核心步骤,不同方法在侧重点和功能实现方面也有所不同.除此之外,方法在基于的前提条件等方面也存在差异,因此本节将从方法的功能覆盖(function)、诊断方式(style)和方法局限性(limitation)这几个角度分别对当前的研究现状和代表性方法进行分析.

2.2.1 功能覆盖

配置错误检测主要解决如何发现错误的问题,回答“是否发生错误以及发生了什么错误(what)”;配置错误诊断重点研究如何发现导致错误的根本原因(root cause),回答“为什么发生错误(why)”;配置错误修复技术解决如何修正配置错误的问题,回答“怎么修复错误(how)”,三者之间紧密相关又有所区别.当前,已有的 SMT 方法具有不同的侧重点,分别关注和解决上述 3 个方面中的某个或某些问题,具有不同的功能覆盖.

2.2.2 诊断方式

SMT 可以分为系统运行前的主动检测和运行时的被动诊断.

系统运行时的被动诊断以运行时的系统行为异常现象为依据,进行错误原因的查找和修复.该方式的优势在于具有针对性,能够准确、有效地排除系统故障;但其局限性在于对方法的效率具有较高要求,尤其是生产环境下,目标系统的故障诊断需要将对系统的影响降低到最小.

系统运行前的主动检测发生在系统运行前,通过约束发现和验证等方法主动进行系统配置检测,发现配置中存在的约束违背和错误.该方式的优势在于无需考虑方法的执行效率、降低系统运行时发生异常的风险,但是该方法需要有配置约束和规则的支持.

2.2.3 方法局限性

已有 SMT 方法或多或少都限定了一些前提条件,使方法必须在满足这些条件限定的前提下才能够得以应用.这些前提条件主要存在于以下几个方面:

1) 目标系统代码.根据需要的系统代码类型,可细分为源代码[38]、字节码[39-41]和二进制代码[10,42,43].这一前提条件主要出现在基于程序分析的方法中;
2) 定制化的目标系统和运行环境.为了能够准确获得目标系统运行时的一些细粒度的行为和状态信息,部分方法需要对目标系统进行植入[10,39,41,43],与之关联的是对目标系统代码(主要是二进制代码)的需求;另外,部分基于重放技术的方法需要对目标系统的运行环境进行定制[44],从而实现预测执行和系统回滚重放;

3) 其他领域知识和数据.在这一方面,主要包括了测试用例[32,44]、基准数据[33,39,42]和历史数据库[35-37]等.

2.3 适用范围(applicability)

应用系统在系统类型、程序语言以及运行环境等方面存在差异配置错误,类型也多种多样,导致的系统故障现象也不尽相同.因此,已有 SMT 存在对应的适用范围,本节从适用范围相关的目标系统类型、配置错误类型以及解决的系统故障类型等 3 个角度进行分析.

2.3.1 目标系统类型(target)

SMT 的目标系统均具有配置参数众多、可配置灵活性强的特征,从适用的系统类型方面,可以细分为操作系统级软件(OS-level)、系统支撑软件(system-level)以及面向最终用户的应用系统(application-level).

2.3.2 配置错误类型(error type)

软件应用的配置错误类型多种多样,包括配置参数取值错误、兼容性错误、组件放置位置错误等.在配置参数取值错误方面,已有工作通常将配置参数项建模为 key-value 形式的名值对,这些配置项通常存储于文本文件、XML 文件以及 Windows 的注册表中,配置参数取值错误还可以进一步细分为拼写错误、大小写错误、格式错误、取值范围错误、取值类型错误等.除此之外,还有部分工作能够诊断或修复包括配置参数取值错误在内的多种类型错误.

2.3.3 系统故障类型(system fault)

配置错误导致的系统故障具有不同的特征与表现,从大的方面可以分为功能异常和性能异常两类.系统功能异常又可以根据故障的严重程度分为多种类型,包括:

1) 系统崩溃、抛出异常或错误信息.此类系统故障具有明显的错误现象,而且系统往往无法继续运行;
2) 系统行为或输出结果不正确.此类系统故障错误现象较为隐蔽,系统能够继续执行,但是其行为或输出错误;
3) 系统行为或输出与用户预期不符.严格意义上来讲,系统在功能方面是正确的,多数是由于配置参数项的变化导致行为或输出与用户本身的需求存在差异.

3 配置错误诊断与修复方法概述

为尽可能保证本文综述的全面性,采用以下方法进行相关文献的检索与选取:

1. 为保证能够反映最新的研究成果和研究动态,以 2000 年以后的研究成果为检索范围
2. 选取了与配置相关(configuration,misconfiguration)、错误相关(error,bug,failure,fault,anomaly)以及诊断修复相关(troubleshooting,detection,diagnose/diagnosis,fix)的多个关键词,采用关键词组合的方式构成文献检索的检索词组;
3. 基于学术搜索引擎(Google scholar)进行文献检索.对第 2 步中的关键词进行多种组合(如 misconfiguration troubleshooting),共使用 10 组检索词组.为保证检索结果的相关性,本文采用基于相关性排序的方式对每次检索结果的前 30 条(top-30)记录进行人工的分析过滤和去重;
4. 基于文献数据库的检索.本文从多个文献数据库中(ACM Digital Library,IEEE Xplore Digital Library)选择与 SMT 最为相关的领域会议/期刊作为检索范围,涉及软件工程(ICSE,FSE,ASE,ISSTA,ICSM,ISSRE,COMPSAC,TSE,TOSEM,etc.)、系统软件与程序设计语言(OSDI,SOSP,OOPSLA,etc.)和计算机系统(ASPLOS,USENIX ATC,LISA,etc.)等共计 20 多个会议/期刊.对检索结果同样采用多组关键词组进行检索、识别和过滤去重;
5. 为了尽可能找到检索词组以外的相关文献,本文对第 3 步、第 4 步检索得到的文献集合的相关工作、参考文献和研究团队逐一分析,找到与本文主题密切相关且在前两步中没有检索发现的相关成果;
6. 综合以上第 3 步~第 5 步的结果构成本文的综述范围,共计相关文献 34 篇,其中,会议/期刊 30 篇,博士论文 2 篇,技术报告 2 篇,详细信息见表 1.

Table 1 Literature search list

表 1 文献检索结果列表

会议/期刊名称	相关结果
IEEE Int'l Conf. on Software Engineering (ICSE)	5 (Ref.[1,39,41,45,46])
IEEE Int'l Conf. on Automated Software Engineering (ASE)	1 (Ref.[9])
Proc. of the ACM SIGOPS/EuroSys European Conf. on Computer Systems (EuroSys)	1 (Ref.[36])
The USENIX Symp. on Operating Systems Design and Implementation (OSDI)	4 (Ref.[10,32,34,43])
Proc. of the ACM SIGOPS Symp. on Operating Systems Principles (SOSP)	4 (Ref.[44,42,14,38])
Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)	2 (Ref.[47,48])
The USENIX Annual Technical Conf. (ATC)	2 (Ref.[37,49])
Proc. of the ACM SIGSOFT Int'l Symp. on Software Testing and Analysis (ISSTA)	1 (Ref.[50])
The USENIX Large Installation System Administration Conf. (LISA)	3 (Ref.[35,33,51])
IEEE Int'l Symp. on Software Reliability Engineering Workshops (ISSREW)	1 (Ref.[52])
IEEE Int'l Symp. on Dependable Systems and Networks With FTCS and DCC (DSN)	1 (Ref.[53])
IEEE Annual Int'l Computers, Software and Applications Conf. (COMPSAC)	1 (Ref.[54])
Companion Proce. of the Int'l Conf. on Software Engineering (ICSE Companion)	1 (Ref.[55])
Int'l Symp. on Software Reliability Engineering (ISSRE)	1 (Ref.[56])
IEEE Int'l Conf. on Cloud Computing (CLOUD)	1 (Ref.[57])
The USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques	1 (Ref.[58])
Ph.D. Thesis	2 (Ref.[40,59])
Technical report	2 (Ref.[60,61])

综上,本文能够较为全面地分析和综述当前 SMT 的主流、高水平研究成果,同时也较为充分地反映了本领域的最新研究现状、动态和进展,具有较强的时效性.

本文分析综述的研究成果来源于国内外多个研究团队与研究学者,其中包括了加州伯克利分校(UC Berkely)的 Katz 团队、密歇根大学(University of Michigan)的 Flinn 团队、华盛顿大学(University of Washington)的 Ernst 团队和圣地亚哥加州大学(UCSD)的 Zhou 团队等.本节将按研究团体和机构分别对具有代表性的研究工作和成果进行简要介绍.

3.1 加州伯克利分校的研究工作

加州大学伯克利分校 Katz 的团队[1,9,40,50]以开源的复杂系统软件(如 Apache Hadoop)为研究对象,提出了基于程序分析的方法来减少软件配置错误.Rabkin 等人的工作主要针对名值对(key-value)类型配置参数项展开 3 个方面问题的研究,包括配置信息管理(documenting configuration)、配置错误调试(debugging configuration)和改进系统日志配置(improving logger configuration),并设计实现了工具 Conf_Analyzer[40].

在配置信息管理方面,针对开源软件存在系统实际配置与文档记录不一致的情况,Rabkin 提出了一种基于静态程序分析的系统配置项识别方法[1],推断系统配置项的参数类型,提高系统文档关于配置信息的准确性.方法首先基于实证研究提出了应用配置项分类,将配置项类型划分为数值型(numeric)、标识(identifier)、使用模式(mode)和其他(other)这 4 类,每一类又根据使用语义分为多个子类;其次,方法基于 JChord[62]对 Java 程序字节码(byte code)进行字段敏感(field sensitive)的静态分析,在找到配置项读取方法后,通过分析方法参数获取关联的配置参数项及名称,通过分析参数读取方法的返回值类型、参数处理相关的系统函数类型和相关参数比较操作信息推断出系统配置参数的取值类型.该方法能够发现大部分的系统配置项,相比原有的系统文档,在配置参数项信息方面具有更好的准确性.

在配置错误调试和诊断方面,提出了一种基于静态数据流分析的检测方法[1,8].首先,方法通过目标系统源代码(Java byte code)的静态分析找到系统读写配置参数的关键语句;然后,方法基于数据流(data flow)分析建立不同程序点与相关配置参数间的映射表.当系统运行产生故障并抛出异常信息时,系统用户可以根据映射表以及抛出异常的代码行号实现关联配置项的快速定位.在此基础上,文章还提出了故障上下文敏感(failure-context-sensitive,简称 FCS)的分析方法,以系统日志信息和堆栈信息(stack trace)为补充来提高分析的准确性和效率.

3.2 密歇根大学的研究工作

密歇根大学 Flinn 研究团队基于动态分析发现输入配置参数与系统异常之间的因果关联关系(causality),通过提高配置错误诊断过程的自动化程度来实现配置错误诊断和修复[59].该团队分别对配置错误引起的系统功能异常和性能异常问题进行研究,并实现了相应的系统工具 SigConf[37],AutoBash[44],ConfAid[10]和 X-ray[43].

SigConf[37]以已知的配置错误信息库为依据,通过粗粒度的因果关联分析(causality analysis)进行配置错误诊断.配置错误信息库存储了已知的历史配置错误及其系统状态信息,对于当前配置错误导致的系统异常,SigConf 通过与已知配置错误的系统状态对比找到相似的已知配置错误.SigConf 运行一组测试用例(predicates)[61],基于每个测试用例的执行路径记录对应的因果依赖,生成相应的签名(signature),而系统状态则是基于测试用例的执行结果和签名生成的.SigConf 的粗粒度因果关联分析主要体现在它记录了测试用例执行过程中所依赖的所有对象,包括文件、目录、类库等,而不细化到对象内部,如配置文件中的具体配置项等.因此,SigConf 具有执行代价小和效率高的优点,能够诊断多种类型的配置错误(如配置取值错误、类库不兼容等),但也由于粗粒度的分析策略,导致在准确度方面存在不足.

AutoBash[44]基于操作系统内核级的预测执行(speculative execution)来测试不同的配置操作给系统带来的影响.AutoBash 同样基于进程和文件粒度的因果关联分析进行错误诊断.AutoBash 以一组测试用例作为辅助,在不影响和改变其他系统的同时,尝试可能的配置错误修复方法,通过反复的尝试和回滚,找到能够解决当前系统故障的配置修复方法.AutoBash 将进程作为黑盒,认为进程的输出完全依赖于其输入数据,在同样粗粒度的分析基础上,AutoBash 能够较为准确地定位存在故障的配置错误对象,但是难以进一步定位其内部的细粒度对象,例如 Apache Http Server 配置文件 httpd.conf 内的某个特定配置参数项.因此对于粗粒度对象中包括众多子项的场景,AutoBash 的诊断效果有限.

ConfAid[10]针对目标系统采用白盒分析策略,通过细粒度的因果关联分析实现配置错误的诊断.ConfAid 对目标系统的二进制代码进行植入,深入进程内部,通过分析系统运行时的动态信息流来获取由数据和控制流引起的因果依赖.ConfAid 将动态分析的范围限定于从配置文件中读取数据的相关语句、变量和取值,一旦发现改变某个配置参数的取值能够引起系统控制流的变化并避免当前系统错误的发生,那么 ConfAid 将这一配置参数作为可能的错误项.相比于前者,ConfAid 主要针对配置文件中配置项取值错误进行诊断,具有较高的准确度;但是细粒度的分析和探针植入将引入较高的额外开销,降低系统运行效率.

与 SigConf 和 ConfAid 不同,X-ray[43]主要基于细粒度的因果关联分析实现对配置错误导致的系统性能异常诊断.X-ray 提出了基于性能摘要(performance summarization)的配置错误诊断方法,首先为每个细粒度事件(如单个指令和系统调用等)确定对应的性能代价(performance cost),如响应时间、I/O 资源使用率等;然后,基于动态信息流分析的方法找到导致每个事件执行的关联配置项(root cause),并根据事件发生的概率将其对应的性能代价赋予对应的配置参数;对于每个配置项,通过汇总其关联的所有事件赋予的性能开销得到该配置项的总体性能代价,总体代价最高的配置项最有可能成为导致系统性能异常的根本原因.

3.3 圣地亚哥加州大学的研究工作

圣地亚哥加州大学(UCSD)Zhou 研究团队在应用配置错误的调研分析和错误诊断方面开展研究.

Yin[14]对多个商业和开源应用系统的配置错误进行了实证研究,是第一个针对配置错误相关特征研究和分析的工作.该工作研究了 546 个真实的配置错误,其中,309 个来源于一个商业化存储系统,另外 237 个错误来源于 4 个广泛使用的开源系统(CentOS,MySQL,Apache Http server,OpenLDAP).Yin 主要从配置错误的类型、模式、原因、系统表现以及造成的影响等几个方面进行了实证研究,将配置错误的类型分为参数错误、兼容性错误、组件错误几大类.该实证研究工作对配置参数错误导致的系统故障诊断与修复研究提供了丰富的论据和事实依据.

Yuan[49]以 Windows 环境下的应用为目标系统,研究 Windows 注册表中应用配置项的错误发现和检测方法.Windows 注册表中存在大量的应用配置信息,并且在运行时有频繁的注册表信息访问事件发生,配置数据的访

问事件序列在一定程度上反映了应用系统的控制流信息,并能够提供程序运行时行为的上下文.考虑到系统正常运行时对于注册表数据的访问行为通常是确定的、有规律的,Yuan 提出了在线的配置错误检测方法,通过监控和记录注册表访事件序列分析和挖掘出注册表配置信息的访问规则,例如:在事件 *a*,*b*,*c* 之后应该确定性地发生事件 *d*,即规则为 *abc*→*d*.当系统运行时监控到违反配置访问规则的事件发生时(例如,在事件 *a*, *b*, *c* 后发生了事件 *e* 而不是 *d*),则认为当前事件可能导致配置错误产生并加以警告.Yuan 基于该方法设计实现了配置错误检测工具 CODE,包括配置数据访问事件监控收集和访问规则分析两部分.CODE 以字典树(trie)表示事件序列,并基于 Sequitur 算法[63]进行规则的分析挖掘.CODE 基于系统外部事件分析,无需目标程序源码以及系统相关语义信息的支持,但其局限性在于必须能够提供大量的、频繁发生的配置访问事件来支持规则的分析挖掘,因此适用于具有配置数据集中存储管理和大量访问的应用场景,如 Windows 环境.另一方面,CODE 基于规则偏离分析解决配置错误的检测和诊断问题,即使在系统没有表现出明显的系统异常时,也能够在线发现潜在的配置错误,通过与已知规则的对比来提供错误解决依据.

Xu[38]从软件系统开发和设计的视角来看待配置错误问题,认为配置错误有时是由系统及配置本身相关的开发设计问题导致的,不应该完全归咎于用户的错误行为.基于这一观点,Xu 针对名值对类型的配置参数错误,基于静态程序数据流分析从配置参数的读取和使用模式中推断出参数配置约束,主要包括单个配置参数的类型约束(type constraint)、取值范围约束(value range)、涉及多配置参数的控制依赖(control dependency)和取值关联(value correlation),以此为依据来辅助程序开发人员改进系统配置的设计,提高系统应对配置错误的能力.SPEX 是基于上述方法实现的配置约束推理工具,首先从目标系统源代码中识别出配置变量;然后,基于数据流分析获取与配置参数对应的每个程序变量,记录沿着数据流路径发现的所有属性约束.在第 2 遍源码分析中,SPEX 基于得到的每个配置参数的代码切片进行配置参数的约束推理.以 SPEX 得到的目标系统参数约束集合为基础,Xu 还实现了配置错误生成与测试工具 SPEX-INJ.SPEX-INJ 生成违背约束的配置设置并注入目标系统,通过运行测试用例观察系统在异常发生时的反馈,以此来评价系统应对配置错误的能力,指导系统配置设置的改进.SPEX 将配置错误的诊断和修复工作提前到开发和设计阶段,更加具有主动性.但是,基于静态分析的方法使得必须有目标系统源代码的支持;同时,SPEX 目前主要应用于单个程序和系统的名值对类型配置参数,对于跨系统的配置间关联导致的约束无法适用.

Zhang[48]研究发现,软件配置项之间以及配置项与运行环境之间存在大量关联(correlation).例如,配置项 *A* 指定某文件路径,而系统环境下该文件路径是否存在影响 *A* 取值的正确与否;又如,配置项 *A* 为某文件,而配置项 *B* 的取值表示文件的拥有者,则 *A* 与 *B* 间存在相互关联.关联是导致配置错误的原因之一,由此,Zhang 实现了基于配置关联的错误检测工具 Encore[48].Encore 基于训练数据集(已有的系统配置)分析挖掘配置关联约束,将获取到的关联约束应用到待检测系统,实现配置错误的主动检测,其主要步骤包括:

1) 训练集数据处理(data assembling).根据配置项语义和取值特征进行配置项类型推断,根据配置项类型为其附加相应的环境信息.例如,当前配置项为文件类型,则对应的环境属性可以包括用户(owner)、用户组(group)、权限(permission)等;

2) 预先定义一组约束模板(template)来描述配置项及环境信息间可能存在的二元关联关系.例如,图 3 中的模板表达了用户名称类型的配置项 *B* 应该是文件类型配置项 *A* 的所有者,其中,*A* 和 *B* 为模板中的占位符(placeholder);

Template: [*A*⟨*FilePath*⟩]⇒[*B*⟨*UserName*⟩]
Rule: *DataDir*⇒*user*

Fig.3 Example of constraint template
图 3 关联约束模板示例

3) 基于模板集合,从训练集中选取满足模板的配置项并逐一进行实例化,产生针对目标系统的一组具体备选约束规则集合;

4) 基于一组启发性规则过滤备选约束集合,提高约束规则的正确性;

5) 基于最终得到的约束规则对目标系统进行检测,对于出现约束违背的配置项产生警告.

除此之外,Encore 还能够对单个配置项的拼写错误、取值类型错误以及取值错误进行检测.Encore 是基于统计分析和数据挖掘的配置错误主动检测方法,它只关注软件系统的配置项,但需要以训练数据集为基础,还需要提供预定义的约束模板;其次,Encore 主要用于检测单个系统内部由于存在配置关联而导致的错误,没有考虑多个系统间存在关联的场景.

3.4 华盛顿大学的研究工作

Zhang 针对单个配置项取值错误导致的系统功能异常设计实现了工具 ConfDiagnoser[39,46].ConfDiagnoser 将静态分析、动态分析和统计学习结合在一起,进行配置错误自动诊断.在静态分析过程中,ConfDiagnoser 采用基于程序薄片撷取(thin slicing)[64]的轻量级依赖关系分析方法识别每个配置项能够影响到的控制流分支判断语句.然后,ConfDiagnoser 有选择性地对目标系统源代码进行植入,获取被配置参数影响的判断语句运行时的行为作为执行概况(execution profile).ConfDiagnoser 的故障诊断以预先建立的正确执行概况数据库为支撑,当系统运行时遇到可疑的配置错误时,ConfDiagnoser 通过运行植入代码的系统进行错误重现,并从数据库中找到与当前错误相似的正确执行概况.最后,基于统计分析的方法,ConfDiagnoser 找到动态行为在正确和错误执行概况之间差别最大的那些判断语句,而与这些判定语句相关的配置则有可能是导致异常的根本原因.

在进一步的工作中,Zhang[41]发现:软件系统演化过程中,新版本软件的配置参数项会发生变化,并导致系统的某些行为或特征发生变化;用户升级新版本后,由于配置的使用和取值的变化,新版本系统的运行与用户的原有预期存在偏差.尽管新版本系统本身正确,但从用户角度来说,与预期的偏差则难以接受.为了保证系统演化后满足用户的预期目标,Zhang 基于运行时植入监控、静态程序分析和运行轨迹对比的方法设计实现了配置诊断工具 ConfSuggester,诊断并发现可能导致系统新老版本行为偏差的配置参数项,用户可以改变这些配置项的取值使系统行为符合其预期.ConfSuggester 的基本思想是:对于软件演化前后的新老版本系统,通过控制流的差异发现程序行为的差异;然后,基于程序行为差异的分析推理找到可能导致差异存在的配置项.ConfSuggester 通过 3 个步骤找到引起新旧版本系统行为偏差的配置项:首先,在两个版本的字节码中植入探针,监控目标是影响控制流变化的判定语句;其次,基于相同的输入和配置运行两个版本的系统,发现两个执行轨迹中的控制流差异;然后,基于一种轻量级的依赖关系静态分析技术,找到造成当前控制流差异的可能配置项集合;最后,通过基于权重的排序算法将可疑配置项的分级排序列表返回给用户,排序越靠前,可疑性越大.与 ConfDiagnoser 不同的是,ConfSuggester 诊断的配置问题更加宽泛,在诊断配置错误的同时,还能够找到导致系统行为违背用户预期的配置项.

3.5 微软研究机构的工作

微软是较早进行 SMT 技术研究的机构之一,主要针对 Windows 环境下基于注册表的配置错误诊断问题展开研究,研究成果有 Strider[33,51]、PeerPressure[34,60]、Snitch[58]和基于事件轨迹(event trace based)的诊断方法[36].

Strider[33]采用黑盒的、基于状态的配置错误诊断方法,将注册表的当前内容作为系统状态[51],采用状态差分(state diffing)的方法逐步缩小可能存在配置错误的搜索范围,最终实现可能的配置错误定位.首先,Strider 将异常时系统状态与本机的正确系统状态(或其他机器的正确系统状态)对比,得到状态差分集合;其次,Strider 重新执行异常程序获取异常发生时涉及的注册表配置项集合;然后,Strider 通过对两个集合执行交集操作进一步缩小可能导致系统异常的注册表配置项范围;最后,针对交集内的可疑配置参数项,Strider 采用一系列启发性规则对其中的配置项进行排序,排序越靠前的配置项成为系统异常原因的可能性越大.

PeerPressure[34,60]相比于 Strider 有了进一步改进:基于统计学习的方法,通过对比运行同样应用的样本系统和当前故障系统的注册表项找到可疑的错误配置项集合;然后,基于贝叶斯统计方法对可疑的错误配置项进行概率计算及排序;最后,依据少数几个候选配置在正确样本系统中的取值,采用试错的方式进行修复.

文献[36]采用事件轨迹(event trace)表示系统行为,将当前发生故障的系统事件轨迹与已知的系统故障及

解决方法的事件轨迹集合进行对比,找到与之相似的事件轨迹,从而进一步找到对应的已知系统故障及其修复方法.该方法以 Windows XP 环境下系统调用(system call)作为系统事件,通过监控和拦截系统调用获取事件信息.方法的系统实现包括 4 个组成部分,其中,troubleshooter 负责故障重现;tracer 同时进行事件轨迹的记录并发送给 classifier;classifier 基于 *n* 元语法模型(*n*-gram model)表示事件序列,并基于支持向量机(support vector machine,简称 SVM)算法[65]进行事件序列中系统行为模式的识别,计算与已知问题库中(known problem DB)事件轨迹的相似度来找到可能相关的已知系统故障及修复方法.

Snitch[58]监控获取大量的应用配置参数及文件的读/写事件信息,然后,基于机器学习的方法建立应用错误与配置状态之间的二元决策树(binary decision tree),反映出错误信息与应用配置之间的关联.应用运行再次发生错误,能够基于该决策树快速定位与错误相关的配置信息.Snitch 主要应用于 Windows 环境下存在频繁、大量配置读/写事件的场景,需要预先获取大量训练数据(应用配置读/写事件)建立二元决策树.对于不同的应用系统,必须建立各自对应的决策树,同时,决策树的建立采用一种交互式的、需要人工指导的方式.

3.6 其他研究工作

Choronus[32]在系统出现异常时首先定位发生异常的时间点,然后,通过对比该时间点前后的系统状态和配置参数的变化来找到可能引起系统故障的配置错误.为了达到这一目标,Chronus 基于虚拟化技术和一个定制的文件系统来找到引起系统无法正常运行的配置参数设置的改变.

ConfDebugger[52]是一个完全基于程序静态分析的轻量级配置错误诊断工具,仅仅依靠配置项信息、系统错误堆栈信息(failure stack trace)以及目标系统的 Java 字节码来定位可能导致当前错误的配置项.ConfDebugger 首先基于正向静态分析和薄片撷取(thin slicing)技术找到每个配置项的程序读取语句以及该语句影响的语句集合;其次,基于错误堆栈信息找到出错的语句,并基于反向静态分析和薄片撷取技术找到当前错误语句关联的语句集合;最后,在两个语句集合的交集中查找相关联的配置项,那么该配置项或配置项集合则作为可能导致错误的所在.

Triage[42]支持在线的故障诊断,采用轻量级的重执行(replay)技术来支持程序运行时异常的重放,获取当时的环境与上下文信息.Triage 采用一种模仿人工调试过程的故障诊断协议进行故障原因分析,通过一组相似的输入和执行环境得到正确和错误的执行结果,并基于动态程序分析进行相似输入且执行结果不同的系统执行路径和数据流的分析,最终诊断出导致系统故障发生的根本原因.Triage 能够诊断多种类型的系统错误,包括配置错误导致的系统故障,并能够提供详细的诊断报告,包括故障信息、触发条件、相关代码和变量、错误扩散链以及可能的修复方法等.

文献[35]提出了一种基于程序行为的配置错误诊断方法,该方法通过监控系统运行轨迹来获得运行时系统调用(system call)、信号量、环境变量等行为信息,并将系统行为与状态合二为一建立系统签名(signature)作为正常运行的基准.当系统运行出现异常时,通过对比异常行为与已有的系统签名来找到当前系统行为存在的偏差,从而分析可能导致系统异常的错误.

ECC Fixer[45]是一个针对 eCOS(embedded configurable operating system)的配置错误修复工具,针对出现约束违背的系统配置项,自动产生满足需求的取值范围(range fix).ECC Fixer 基于 Reiter[66]诊断理论,将配置参数取值问题映射为可满足问题(SAT)求解,从约束和配置集合中得到需要重新设置和修改的最小变量集合;然后,基于谓词逻辑将最小变量集合涉及的约束转化为等价的合取范式,并将每个子句作为一个最小的修复单元(fix unit);最后,在产生全局的修复取值范围时考虑了多个约束间的相互作用,为涉及的多个配置参数产生无冲突的取值范围集合.

ConfErr[53]是第一个模拟配置错误并进行目标系统注入和评估的工具.ConfErr 模拟用户在系统配置时人工操作可能造成的拼写错误、结构错误以及语义错误,将上述类型错误随机注入到目标系统对应的配置文件中,然后计算和评估系统配置错误的应对能力和表现,生成相应的分析和评估概况.ConfErr 同样可以用来作为目标系统应对配置错误能力的测试和评估基准,其生成的分析和评估概况可以作为系统改进的依据,也可以用于在相似的系统间进行容错能力的对比.

Arshad[56]对 Java EE 应用服务器的配置问题进行了特征研究,对开源应用服务器 GlassFish 和 JBoss 的 281 个配置问题,从 4 个维度进行分析,并基于分析结果设计实现了一个面向 Java EE 的应用服务器配置错误注入及评估工具 ConfInject.ConfInject 向应用服务器的 XML 配置文件进行基于字符串的配置项错误注入,并从 4 个方面对该错误导致应用服务器出现异常的情况进行分析,用以评价目标应用服务器应对配置参数错误的能力.

Meng[57]针对大规模分布式的、基于多层体系架构的网络应用系统提出了一种基于分析和比较的配置文件自动发现方法.该方法把待分析的目标机器与一组基准镜像虚拟机进行比较,通过分析对比目标机器与镜像之间、镜像与镜像之间在文件方面发生的变化,总结提出了 6 种文件变化模式(change patter,简称 CP),并将属于 6 种 CP 的文件作为候选配置文件.最后,该方法采用基于领域知识库(存储已有的配置文件发现语义规则)或人机交互的方式对选配置文件进行最终判断和认定.

文献[54]从系统错误报告(bug report)的角度来判断当前系统错误是否由配置错误导致(configuration bug report prediction).该工作基于特征选择技术(feature selection technique),如信息增益技术(information gain)和卡法检验(Chi-square),从大量错误报告文本中选择紧密相关的特征术语(significant term);然后,使用多种文本挖掘方法(如朴素贝叶斯、支持向量机、朴素贝叶斯多项式等)建立错误报告的分类统计模型(classifier);最后,给定一个错误报告,该模型能够判断出当前错误是否是配置错误相关的.

4 软件配置错误诊断方法分析

本节依据第 2 节提出的配置错误诊断方法分析框架对第 3 节概述的代表性工作进行分析,并对每个分析方面和角度中的各类工作进行统计.

4.1 方法类型分析

所有代表性方法中,基于程序分析的方法、基于统计学习的方法和混合方法分别为 5/18,4/18,5/18;此外,基于对比的方法和基于重放的方法分别占 3/18 和 1/18(由于 ECC Fixer[45]仅针对配置修复问题,因此不在方法类型分析的统计范围内,详见表 2).

4.1.1 基于程序分析的方法

基于程序分析的配置错误诊断方法可以进一步分为基于静态程序分析的方法和基于动态程序分析的方法.基于静态程序分析的代表有 Conf_Analyzer[40],ConfDebugger[52]和 SPEX[38],而基于动态程序分析方法的工具有 ConfAid[10]和 X-ray[43].

Conf_Analyzer,ConfDebugger 和 SPEX 分别对目标系统的字节码(byte code)和源代码进行静态分析,而 ConfAid 和 X-ray 则基于系统的二进制代码(binary code)进行植入、监控和分析.程序分析的配置错误检测与诊断方法具有较高的准确率,但同时也带来了较高的额外开销.为了提高执行效率,该类型的方法需要考虑路径覆盖和分析规模的问题,尽可能通过启发式规则降低分析复杂度.基于程序分析方法的局限性还包括:1) 程序语言相关性;2) 编程风格和编程实践影响方法的有效性.如果程序捕获所有异常,并抛出一个没有实质内容的泛泛的出错信息,那么该方法将难以进行相关配置参数项的定位.

4.1.2 基于统计学习的方法

CODE[49]从大量的、频繁的配置数据访问事件中分析挖掘出事件序列表示的配置访问模式和规则,通过发现违背规则的事件来检测配置错误.文献[36]采用事件轨迹(event trace)表示系统行为,基于 SVM 识别事件序列中的系统行为模式.Encore[48]从大量的训练数据集中分析挖掘出配置项间的关联约束.Sintch[58]基于训练数据集建立二元决策树来辅助配置错误诊断.

基于统计学习的方法需要大量历史信息作为分析和挖掘的数据基础,例如,CODE 和文献[36]均以 Windows 环境下的注册表信息作为配置参数实例,而 Encore 则从 Amazon EC2 中的公共镜像文件中获取训练数据.

4.1.3 基于重放技术的方法

代表性工作有 AutoBash[44]和 Triage[42]:AutoBash 基于操作系统内核级的预测执行(speculative execution)来测试不同的配置操作变化给系统带来的影响;Triage 采用轻量级的重执行技术来支持程序运行异常时的重

放,获取当时的环境与上下文信息.

AutoBash 对底层操作系统内核进行了定制构造沙箱环境,在 Linux 内核之上,实现了系统状态的回滚和与其他应用的隔离;其次,AutoBash 和 Triage 还需要一组测试用例来检验修改配置参数前后目标系统的变化.

4.1.4 基于对比的方法

Strider[33]和 PeerPresure[34]以本机正确系统状态以及样本机的系统状态为基准,通过错误系统状态与基准的状态差异对比进行诊断;Choronus[32]通过对比系统故障发生前后系统状态和配置参数的变化找到可能引起系统故障的配置错误;signature-based[35]方法对比异常行为与已有的系统签名来找到当前系统行为存在的偏差,分析可能导致系统异常的错误;SigConf[37]以已知的配置错误信息库为依据,通过与已知配置错误对应的系统状态对比找到相似的已知配置错误,实现配置错误的诊断.

4.1.5 混合方法

采用混合方法进行配置错误诊断的有 ConfDiagnoser[39],ConfSuggester[41]和 Triage[42]:ConfDiagnoser[39]综合采用了静态程序分析、动态程序分析和对比技术,通过静态分析识别每个配置项能够影响的控制分支判断语句,通过动态分析获取系统的执行概况,最后对比找到正确和错误执行概况之间差别最大的判断语句集合;ConfSuggester[41]同样基于静态、动态程序分析和运行轨迹对比发现系统演化过程中新老版本间配置的差异;Triage[42]基于重放技术获取系统故障时的环境与上下文信息,对输入相同但执行结果不同的多次系统执行进行动态程序分析,最终诊断出导致系统故障发生的根本原因.

如前所述,混合方法充分融合了不同方法的优势,对其局限性也兼而有之:ConfDiagnoser 和 ConfSuggester 需要程序分析必须的源码支持,具有编程语言的针对性;其次,ConfDiagnoser 还需要以正确执行概况的数据信息作为与当前系统故障行为进行对比的基准.

4.2 方法特征分析

4.2.1 功能覆盖分析

现有方法的研究重点是配置错误诊断(12/19),而错误发现和修复各占 2/19,还有部分工作覆盖了诊断与修复两个方面(3/19).

配置错误诊断方法含 Conf_Analyzer[40],ConfAid[10],X-ray[43],SPEX[38],ConfDiagnoser[39],ConfSuggester[41],Strider[33],PeerPressure[34],Snitch[58],ConfDebugger[52]和 Choronus[32]等.上述研究成果的前提通常假设配置错误导致的系统异常是能够明显被观察到和发现的,例如抛出异常信息或崩溃[10,40]、系统执行错误[39]、输出信息与预期不符[41]和系统性能显著下降[43]等,因此,诊断出导致系统异常的根本原因成为修复系统故障的必要前提和核心问题,也使得配置错误诊断方法的研究成为最受关注的研究问题.

在明显的、可被观察的系统故障和异常之外还存在无法及时发现的系统错误,使得系统配置错误检测也成为研究问题之一.CODE[49]是系统配置错误检测的代表性工作,能够通过在线的配置访问事件序列对比发现违背配置访问模式和规则事件的发生,从而在系统进行错误传播并表现出某些故障现象之前能够及时检测到错误的发生.

配置错误修复的研究目标是实现错误修复方法的自动生成,甚至是自动修复,进一步提高方法整体的自动化程度,降低对用户知识和经验的要求.基于对比的方法通常能够在实现错误诊断的同时给出问题的解决方案,这是由于有对比基准或已知错误的解决方法作为参考,如 SigConf[37],Signature based[35],Event trace based[36].基于重放技术的方法则通过重放技术将系统错误调试和修复的过程自动化,达到错误修复的目标,代表性工作有 AutoBash[44]和 Triage[42].ECC Fixer[45]是一个专门以配置项取值错误修复为目标的工具,采用逻辑推理和约束求解的方法,为存在约束违背的系统配置项自动生成正确的取值范围.

4.2.2 诊断方式分析

从已有研究方法和成果来看,系统运行时的被动诊断为主要方式,在本文分析的 19 个代表性方法中,有 16 个采用的是故障发生后的诊断方式.这些方法大多基于系统故障的现象、特征或者通过再次的错误执行重放(replay)或重执行(re-execution)来获取细粒度的、深层次的错误相关信息,从而有针对性地进行诊断和修复.

相比于运行时的被动诊断,已有工作中采取主动的运行前诊断的方法主要有 ECC Fixer[45],SPEX[38]和 Encore[48].这些方法基于目标系统已有的配置约束[38]、程序分析推断出的配置约束[45]和基于统计学习得到的约束[48]为依据,采用约束验证的方法来主动检测和发现当前系统配置参数存在的约束违背.ECC Fixer 还能够通过逻辑推理和约束求解的方法为存错误的配置参数项自动生成正确的取值范围.

4.2.3 方法局限性分析

本文分析的现有方法多数具有多个前提或局限性(8/19).其中,需要系统代码并植入探针的方法占 4/19,包括 ConfAid[10],X-ray[43],ConfDiagnoser[39]和 ConfSuggester[41],这些方法均通过对目标系统代码进行探针植入来获取系统运行时的动态信息.需要系统代码并人工标记分析起始位置的方法占 2/19,包括 Conf_Analyzer[40]和 SPEX[38],这些方法则需要通过人工标注的方法来限定程序静态分析的范围和起始位置,提高分析效率.需要运行环境定制并提供测试用例的方法[44]和需要代码、监控植入以及领域知识支持的方法[39]各占 1/19.存在单个前提条件的方法中,需要领域知识和数据支持的方法占了 8/19,包括 SigConf[37],CODE[49],Strider[33],PeerPresure[34],Encore[48],Snitch[58]等,均是以领域知识和历史数据作为分析诊断的依据.而仅需要系统代码[52]或测试用例支持[32]或已知配置约束[45]的方法则各占 1/19.

4.3 适用范围分析

4.3.1 目标系统类型分析

代表性方法主要以系统支撑软件为研究目标(12/19);其次是应用软件(6/19);以操作系统为目标系统的方法所占比例最小(1/19),主要以嵌入式可配置系统和 Linux 系统为对象.

ECC Fixer[45]实现 eCos 配置错误的修复,eCos 是一种嵌入式可配置实时操作系统,因此,其目标系统属于操作系统级软件.当前,系统支撑软件越来越具有规模大和配置复杂的特征,例如 Apache http server,mysql,Hadoop 等.互联网应用、大数据和云计算等新技术的发展与应用,也使得这些系统支撑软件成为当前已有工作的主要研究对象,面向系统软件的配置诊断方法和工具有 Conf_Analyzer[40],ConfAid[10],ConfDiagnoser[39]等.CODE[49],Strider[33],PeerPresure[34],Event trace based[36],Snitch[58]以 Windows 环境下的应用软件作为目标系统,如 Internet Explorer 和 Outlook 等,这些应用以 Windows 的注册表作为配置设置和管理的基础,使得上述工作采取的方法能够获取丰富的配置相关数据作为配置错误检测、诊断和修复的依据.

4.3.2 配置错误类型分析

诊断名值对类型的配置参数取值错误方法占 13/19,其他方法(6/19)则在配置错误类型方面没有明确限制.

基于程序分析的诊断方法主要围绕配置参数取值错误展开,该方法能够获得较为详细的配置项读取和使用信息等,代表性工作有 Conf_Analyzer[40],ConfAid[10],X-ray[43],CODE[49],SPEX[38],Encore[48],ConfDiagnoser[39],ConfSuggester[41],Strider[33]和 PeerPresure[34].除此之外,还有部分工作能够诊断或修复多种类型的配置,这些工作主要采取的是基于对比[10,21]和基于重放[13,14]的方法,由于有已知故障及修复信息作为基准,使得基于对比方法的诊断和修复技术能够应付多种配置错误类型,而不是仅仅局限于配置参数的取值错误;基于重放技术的修复方法由于提高了调试过程的自动化程度,也能够修复多种类型的配置错误.

4.3.3 系统故障类型分析

当前已有方法主要是面向功能故障相关的配置错误诊断,占 18/19,仅有少数工作诊断系统性能异常相关的配置错误(1/19).

代表性工作多数面向系统崩溃、抛出异常或错误信息的功能异常,如 Conf_Analyzer[40],ConfAid[10],SPEX[38]等;ConfDiagnoser[39]以正确的执行概况为基准,使得系统行为即使没有明显的异常表现,在发生偏差时就能够得到诊断;ConfSuggester[41]以系统演化过程中配置参数变化导致的新老版本的行为应用场景,采用新老版本系统行为差异的对比方法,使导致系统功能与用户需求不符的这一类配置问题得以诊断.当前,对于配置错误相关的系统性能异常诊断和修复的研究工作较少,代表性的工作主要有 X-ray,它基于动态程序分析的方法获得程序执行语句的细粒度性能代价,然后,根据异常发生时汇总的所有语句的总体性能代价以及与配置项的关联度进行诊断.

4.4 小 结

本节对国内外具有代表性 SMT 研究成果进行归纳,见表 2.从本文提出的分析框架的方法类型、方法特征以及适用范围这 3 个方面和多个角度对研究工作进行了总结,当前软件配置错误诊断与修复相关的研究成果呈现出以下特点:

(1) 从方法类型来看,较早的研究工作[33,34,36]主要通过建立系统表现出来的外部行为特征以及系统配置的关联来诊断和修复配置错误,但越来越多的工作[10,38,40]采用程序分析方法从系统内部获取细粒度的数据依赖和控制流信息,使配置错误的诊断和修复更为准确.最新的研究工作[39,41]更趋向于采用多种方法相结合的混合方法,能够借助各种方法的优势,需要注意的是如何避免相应方法本身的一些局限性;

(2) 从方法特征来看,首先在功能覆盖方面,当前代表性的研究工作大多覆盖了配置错误诊断阶段,并将该问题作为研究的重点,同时,越来越多的工作[38,45]将关注点投向配置错误的自动修复方面,使错误修复更加自动化,降低对用户的知识与经验的依赖度;其次,在诊断方式上,当前研究成果主要采用的是一种运行时的被动诊断和修复方法,在错误发生时和发生后采用在线或者离线的方法完成配置错误的诊断和修复;在方法局限性方面,由前述分析可以看到,每种方法或多或少在系统代码、运行环境和其他数据知识方面存在一定的前提和假设,因此很多研究工作的重点之一也是如何消除或放宽这些前提条件,使方法在使用范围以及自动化程度等方面有所提高;

(3) 从适用范围来看,当前已有的研究工作主要面向大规模的网络应用支撑系统软件和特定的 Windows 环境下的客户应用,而解决的配置错误类型主要是具有 key-value 形式的配置项取值错误,而这类配置错误引起的系统异常和系统故障也主要是功能性异常.

Table 2 Analysis of SMT methods

表 2 软件配置错误诊断与修复技术分析

	方法类型	方法特征			适用范围		
		功能覆盖	诊断方式	局限性	目标系统	错误类型	故障类型
Conf_Analyzer[40]	A	D	R	C+A	S	K	F
SigConf[37,61]	C	D+F	R	K	S	A	F
AutoBash[44]	R	F	R	E+T	S	A	F
ConfAid[10]	A	D	R	C+I	S	K	F
X-ray[43]	A	D	R	C+I	S	K	P
CODE[49]	S	E	R	K	A	K	F
SPEX[38]	A	D	P	C+A	S	K	F
ConfDiagnoser[39,46]	A+C	D	R	C+I+K	S	K	F
ConfSuggester[41]	A+C	D	R	C+I	S	K	F
Strider[33,51]	C	D	R	K	A	K	F
PeerPressure[34,60]	C+S	D	R	K	A	K	F
Snitch[58]	S	D	R	K	A	K	F
Event trace based[36]	S	D+F	R	K	A	A	F
Choronus[32]	R+C	D	R	T	S	A	F
ConfDebugger[52]	A	D	R	C	A	K	F
Triage[42]	R+A	D+F	R	C+I	S	A	F
Signature based[35]	C	D	R	K	S	A	F
ECC Fixer[45]	-	F	p	R	O	K	F
Encore[48]	S	E	P	K	S	K	F

方法类型:A—Program analysis based, S—Statistics based, C—Comparison based, R—Replay based;
功能覆盖:E—Exploring, D—Diagnosing, F—Fixing;
诊断方式:R—Reactive, P—Proactive;
局限性:C—Code, T—Test case, E—Environment customization, K—Knowledge DB,
I—Instrumentation, A—Annotation, R—Constraint;
目标系统:O—Operating system; S—System software; A—Application;
错误类型:K—Key value, A—Any;
故障类型:F—Functional fault, P—Performance fault.

总结当前的研究工作,主要存在以下不足:

(1) 目标系统主要是单个软件系统,忽略了构成应用的运行支撑环境以及构成复杂网络应用的不同组件和系统之间在配置参数上存在的隐式关联.

已有工作主要针对单个目标系统的配置问题进行研究[1],例如系统软件 Hadoop、HBase 和 Java EE 应用服务器等.实际应用中,复杂应用(尤其是复杂网络应用)是由操作系统、中间件、数据库以及异构组件构成的上层应用组成的,不同层次之间以及异构组件之间存在着依赖和交互,导致复杂网络应用的不同组成部分之间在配置参数方面同样会存在关联.例如,复杂网络应用中,数据库访问组件中的数据源相关配置与底层数据库的相关配置之间就存在着关联关系.在这种情况下,如果无法获取构成应用的组件及系统间配置参数的关联,那么在进行系统的配置或迁移时会导致配置参数设置和修改的遗漏,使得存在关联关系的配置参数发生相关约束和规则违背的情况,并最终导致系统故障的发生.

(2) 已有研究成果主要采取的是一种系统异常发生时的被动诊断与修复策略,主动检测与修复相对薄弱.

被动的错误诊断与修复策略具有针对性,但是为了减少系统失效时间,对方法的效率和代价有较高要求.加强系统运行前的、主动的配置错误检测力度,能够避免部分配置错误在运行时发生,降低系统由于配置错误导致运行异常和失效的几率.因此,深入研究运行前的主动检测与修复机制是运行时诊断与修复的有效补充,通过覆盖运行前与运行时两个阶段来降低出错几率,保障系统可应用和提高服务质量.

(3) 已有研究成果通常假设当前系统异常由且仅由一个配置参数项的错误导致,对于多个配置项错误导致的系统异常情况难以适用.

当前,已有的系统配置错误诊断方法通常假设导致当前系统故障的配置错误有且仅有一个,然而实际情况下,很多应用错误都是由少数几个参数的相互作用导致的.例如,Kuhn 和 Reilly 分析了 Mozilla 浏览器的错误报告记录,发现超过 70%的错误是由某两个参数的相互作用触发的,超过 90%的错误是由 3 个以内的参数互相作用而引发的[67].因此,已有方法不适用于多配置错误的诊断与修复.

对于存在多个配置项错误的情况,基于程序分析的方法需要对配置参数项进行逐个分析,存在效率低下的问题;而基于对比的方法对于状态和现象高度相似的系统异常存在错误检测和诊断的困难.同时,多配置参数错误导致的系统异常不会因为部分参数的修正而得到解决,而且有可能因为部分参数的修正导致系统异常现象发生变化,这都为基于程序分析的多配置项错误诊断带来了挑战.

(4) 在方法适用范围方面,已有方法的目标配置错误类型和系统故障类型较为单一.

当前方法主要针对键值对(key-value)形式的配置参数取值错误,对于其他类型的配置错误(如兼容性错误、组件放置位置错误等)的研究涉及较少.因此,对于其他类型配置错误的检测、诊断和故障修复技术研究,是对当前已有工作的有益补充,能够有效提高 SMT 的覆盖面和完整性.

在系统故障方面,当前方法主要解决配置错误导致的系统功能异常问题.相比于系统性能异常,功能异常更加明确,且可能关联的参数配置项也较为固定.而系统性能异常则往往是由于系统资源相关的参数配置项设置不合理导致的,异常现象本身具有隐蔽性和滞后性(通常是在一定的场景下才发生,例如系统的并发压力达到一定程度).进一步的,导致系统性能异常的资源相关配置参数之间存在着相互的关联和制约,往往一个配置项的变化也会影响其他配置项的最佳设置值.因此,对于系统性能异常相关的配置错误诊断与修复具有更大的难度.

也有相关工作[55]对当前 SMT 的实用性进行讨论.通过对高可配置(highly configurable)的一个商业化系统和两个开源系统进行定量分析,该工作认为,SMT 的挑战存在于:1) 复杂系统可以基于多种开发语言,因此,基于程序分析的 SMT 必须能够跨程序语言进行分析;2) 系统存在很多修改配置的语句和方法需要从多个源头和位置进行配置追踪(tracing);3) 仅仅基于系统出错时的持久化数据进行配置状态分析是不够的,需要结合运行时的内存数据与其他启发式规则来提高分析的准确性.

5 研究展望

通过当前软件配置错误检测、诊断与修复相关的研究热点和主要工作分析,本文认为,该领域的研究趋势和研究内容包括以下几个方面:

(1) 考虑参数关联和组件依赖的配置错误主动检测

构成复杂网络应用的组件存在频繁交互和相互依赖,导致配置参数间产生隐式的复杂关联,给应用的配置及其错误诊断带来困难.基于异构组件的复杂网络应用运行在多种/多层中间件平台上,分层架构使组件之间、容器之间以及组件与容器之间存在关联和依赖,也使应用及环境的配置参数之间存在隐式的复杂关联和约束,例如配置参数取值的一致性、依赖性、取值范围的相互制约等.主动检测以配置中存在的不变性规则和约束为基础,配置约束具有隐式的特点,隐藏于配置参数的使用方式及其语义中.因此,实现配置主动检测的难点和关键点在于发现应用配置参数关联及其蕴含的约束,并通过形式化的方法进行显示描述,实施自动化的配置约束验证.

(2) 系统性能异常及可靠性、可用性等问题相关的配置错误诊断与修复

在配置参数错误导致的系统性能异常诊断方面,配置参数与性能异常现象间的关联更为复杂,同时,很多情况下系统的运行时故障不是由单个配置参数引起的,而是由几个参数的相互作用导致[67],某一配置参数的取值变化可能导致其他参数的最佳取值范围也随之发生改变,很难通过程序分析进行诊断.而基于历史数据的统计分析方法能够发现异常现象与配置参数之间以及两两配置参数之间潜在的关联性,是诊断性能异常的有效方法.除配置错误引起的功能和性能异常外,还能导致系统可靠性及可用性等问题.相比于功能和性能异常,此类问题对系统的影响等级较轻,且具有一定的主观性,因此,发现、诊断和修复配置错误引起的此类问题也具有更大的难度.

(3) 配置错误修复技术研究

很多情况下,一个系统错误的修复恰恰是新错误产生的直接原因[5],因此,配置错误修复必须确保不引入新的系统错误,其中的关键问题在于如何推断出错误配置项的正确取值或取值范围,以及如何确认当前修复方法的有效性,避免引入新的错误.当前,已有工作采用程序分析[38]和约束求解[45]的方法在这一方面做出了探索和尝试,在基于程序分析的配置取值自动推断技术上,除了根据配置项的取值和使用信息来推断可能取值外,还应该分析该配置项与程序其他变量和执行路径的关联,并充分分析在该取值范围内程序的行为是否会发生改变,从而避免新错误的发生.

(4) 应用配置错误辅助诊断能力提升的相关研究

应用的错误提示信息缺少可能的错误原因、系统状态和错误范围等描述,使配置错误难以诊断,因此,提高系统的错误应对能力和辅助诊断能力十分关键.当前,已有相关工作通过完善日志信息和异常信息提示的方法来提高系统对于错误信息的描述能力,从而辅助诊断[47].在这一研究问题上,可以采取回溯(back tracking)的方法,从异常信息抛出位置逆向分析与之关联度高的配置参数集合以及相关的系统状态,并作为系统提示信息的输出内容加以完善.

(5) 软件配置方法研究

当前研究成果主要集中在如何被动的进行配置错误的检测、诊断和修复方面,忽视了如何从软件工程领域的视角出发,研究如何主动减少或避免配置错误的发生.今后的研究工作可以从方法论以及建立规范机制的角度出发,研究软件本身的配置方法,如:针对不同的软件类型归纳总结出相应的软件配置方法模式以及从配置的功能和语义角度出发,对配置命名及取值类型等建立相应的规范机制.

6 总 结

软件应用的多样性、复杂性、灵活性和高度可定制性对系统的正确配置提出了挑战,配置错误已经成为影响应用服务质量的关键问题之一.很多学者和研究机构致力于配置错误的检测、诊断和故障修复相关的技术和方法的研究,以提高复杂应用系统的可用性和可靠性.本文对软件配置错误诊断修复问题进行深入探讨.文章分析了导致系统配置错误的多个因素;然后,从方法类型、方法特征和适用范围这 3 个方面和多个角度建立了一个 SMT 分析框架;接着,对研究现状进行概述,并基于该分析框架对代表性工作进行分类总结和分析评价;最后,本文总结了当前相关研究存在的不足,并对今后的研究趋势进行展望,对于今后该领域的继续和深入研究具有

指导意义.

References:

[1] Rabkin A, Katz R. Static extraction of program configuration options. In: Proc. of the 33rd Int'l Conf. on Software Engineering (ICSE). 2011. [doi: 10.1145/1985793.1985812]

[2] Oppenheimer D, Ganapathi A, Patterson DA. Why do Internet services fail, and what can be done about it? In: Proc. of the 4th USENIX Symp. on Internet Technologies and Systems (USITS). 2003.

[3] Gray J. Why do computer stop and what can be done about it? Technical Report, 85.7, Tandem Corp., 1985.

[4] Hale B. Why every IT practitioner should care about network change and configuration management. 2012. http://web.swcdn.net/creative/pdf/Wh-itepapers/Why_Every_IT_Practitioner_Should_Care_About_NCCM.pdf

[5] Sverdlik Y. Microsoft: Misconfigured network device led to azure outage. 2012. http://www.datacenterdynamics.com/focus/archive/2012/07/microsoft-misconfigured-network-device-led-azure-outage

[6] Amazon Web Services Team. Summary of the Amazon EC2 and Amazon RDS service disruption in the US east region. 2011. http://aws.amazon.com/message/65648

[7] Johnson R. More details on today's outage. http://www.facebook.com/notes/facebook-engineering/more-details-on-todays-outage/431441338919

[8] CircleID. Misconfiguration brings down entire .SE domain in Sweden. www.circleid.com/posts/misconfiguration_brings_down_entire_se_domain_in_sweden/

[9] Rabkin A, Katz R. Precomputing possible configuration error diagnoses. In: Proc. of the 26th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). 2011. [doi: 10.1109/ASE.2011.6100053]

[10] Attariyan M, Flinn J. Automating configuration troubleshooting with dynamic information flow analysis. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation (OSDI). 2010.

[11] Hadoop. http://hadoop.apache.org/

[12] Rowstron A, Druschel P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Proc. of the IFIP/ACM Int'l Conf. on Distributed Systems Platforms (Middleware), 2001.

[13] HBase. http://hbase.apache.org/

[14] Yin Z, Ma X, Zheng J, Zhou Y, Bairavasundaram L, Pasupath S. An empirical study on configuration errors in commercial and open source systems. In: Proc. of the 23rd ACM Symp. on Operating Systems Principles (SOSP). 2011. [doi: 10.1145/2043556.2043572]

[15] Eddy N. Human error caused Google glitch. 2009. http://www.eweek.com/c/a/Enterprise-Applications/Human-Error-Caused-Google-Glitch/

[16] Levesque M. Fundamental issues with open source software development. 2005. http://dlc.dlib.indiana.edu/dlc/bitstream/handle/10535/3215/Fundamental_issues_with_open_source_software_development.pdf? sequence=1&isAllowed=y

[17] Schreck D, Dallmeier V, Zimmermann T. How documentation evolves over time. In: Proc. of the 9th Int'l Workshop on Principles of Software Evolution (IWPSE). 2007. [doi: 10.1145/1294948.1294952]

[18] Nagaraja K, Oliveira F, Bianchini R, Martin RP, Nguyen TD. Understanding and dealing with operator mistakes in Internet services. In: Proc. of the 6th Symp. on Operating Systems Design & Implementation (OSDI). 2004.

[19] Oliveira F, Nagaraja K, Bachwani R, Bianchini R, Martin RP, Nguyen TD. Understanding and validating database system administration. In: Proc. of the USENIX Annual Technical Conf. (ATC). 2006.

[20] Böehm H, Feldmann A, Maennel O, Reiser C, Volk R. Network wide inter-domain routing policies: Design and realization. In: Proc. of the 34th Conf. on North American Network Operators' Group Meeting. 2005.

[21] Chen X, Mao Y, Mao ZM, van der Merwe J. Declarative configuration management for complex and dynamic networks. In: Proc. of the 6th Int'l Conf. on Emerging Networking Experiments and Technologies (CoNext). 2010. 61-72. [doi: 10.1145/ 1921168.1921176]

[22] Huan L, Dan O. Remote network labs: An on-demand network cloud for configuration testing. SIGCOMM Computer Communication Review, 2010,40(1):83-91. [doi: 10.1145/1672308.1672324]

[23] Al-Shaer ES, Hamed HH. Discovery of policy anomalies in distributed firewalls. In: Proc. of the 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM). 2004. 2605-2616. [doi: 10.1109/INFCOM.2004. 1354680]

[24] Enck W, Moyer T, McDaniel P, Sen S, Sebos P, Spoerel S, Greenberg A, Sung YW, Rao S, Aiello W. Configuration management at massive scale: System design and experience. IEEE Journal on Selected Areas in Communications, 2009,27(3):323-335. [doi: 10. 1109/JSAC.2009.090408]

[25] Li FL, Yang JH, Wu JP, An CQ, Jiang N. Research on Internet automatic configuration. Ruan Jian Xue Bao/Journal of Software, 2014,25(1):118-134 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4458.htm

[26] Chen HF, Jiang GF, Yoshihira K, Saxena A. Invariants based failure diagnosis in distributed computing systems. In: Proc. of the 29th IEEE Symp. on Reliable Distributed Systems. India, 2010. 160-166. [doi: 10.1109/SRDS.2010.26]

[27] Chen HF, Jiang GF, Ungureanu C, Yoshihira K. Failure detection and localization in component based systems by online tracking. In: Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Data Mining. Chicago, 2005. 750-755. [doi: 10.1145/ 1081870.1081968]

[28] Bodic P, Friedman G, Biewald L, Levine H, Candea G, Patel K, Tolle G, Hui J, Fox A, Jordan MI, Patterson D. Combining visualization and statistical analysis to improve operator confidence and efficiency for failure detection and localization. In: Proc. of the 2nd Int'l Conf. on Autonomic Computing. 2005. 89-100. [doi: 10.1109/ICAC.2005.18]

[29] Cherkasova L, Ozonat K, Mi N, Symons J, Smirni E. Automated anomaly detection and performance modeling of enterprise applications. ACM Trans. on Computer Systems, 2009,27(3):1-32. [doi: 10.1145/1629087.1629089]

[30] Zhang L, Xie B, Mei H, Shao WZ, Yang FQ. Study of component-based software configuration management technologies. Acta Electronica Sinica, 2001,29(2):1-3 (in Chinese with English abstract).

[31] Sandbox. http://en.wikipedia.org/wiki/Sandbox_(computer_security)

[32] Whitaker A, Cox RS, Gribble SD. Configuration debugging as search: Finding the needle in the haystack. In: Proc. of the 6th USENIX Conf. on Operating Systems Design and Implementation (OSDI). 2004.

[33] Wang YM, Verbowski C, Dunagan J, Chen Y, Wang HJ, Yuan C, Zhang Z. STRIDER: A black-box, state-based approach to change and configuration management and support. In: Proc. of the 17th Large Installation Systems Administration Conf. (LISA). 2003. [doi: 10.1016/j.scico.2003.12.009]

[34] Wang HJ, Platt JC, Chen Y, Zhang R, Wang YM. Automatic misconfiguration troubleshooting with PeerPressure. In: Proc. of the 6th USENIX Conf. on Operating Systems Design and Implementation (OSDI). 2004.

[35] Ding XN, Huang H, Ruan YP, Shaikh A, Zhang XD. Automatic software fault diagnosis by exploiting application signatures. In: Proc. of the 22nd Large Installation Systems Administration Conf. (LISA). 2008.

[36] Yuan C, Lao N, Wen JR, Li J, Zhang Z, Wang YM, Ma WY. Automated known problem diagnosis with event traces. In: Proc. of the 1st ACM SIGOPS/EuroSys European Conf. on Computer Systems (EuroSys). 2006. [doi: 10.1145/1217935.1217972]

[37] Attariyan M, Flinn J. Using causality to diagnose configuration bugs. In: Proc. of the 2008 USENIX Annual Technical Conf. (ATC). Boston, 2008.

[38] Xu TY, Zhang JQ, Huang P, Zheng J, Sheng TW, Yuan D, Zhou YY, Pasupathy S. Do not blame users for misconfigurations. In: Proc. of the 24th ACM Symp. on Operating Systems Principles (SOSP). 2013. [doi: 10.1145/2517349.2522727]

[39] Zhang S, Ernst MD. Automated diagnosis of software configuration errors. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). 2013. [doi: 10.1109/ICSE.2013.6606577]

[40] Rabkin A. Using program analysis to reduce misconfiguration in open source systems software [Ph.D. Thesis]. Berkeley: University of California, 2012.

[41] Zhang S, Ernst MD. Which configuration option should I change? In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE). 2014. [doi: 10.1145/2568225.2568251]

[42] Tucek J, Lu S, Huang C, Xanthos S, Zhou YY. Triage: Diagnosing production run failures at the user's site. In: Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP). 2007. [doi: 10.1145/1294261.1294275]

[43] Attariyan M, Chow M, Flinn J. X-Ray: Automating root-cause diagnosis of performance anomalies in production software. In: Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation (OSDI). Hollywood, 2012.

[44] Su YY, Attariyan M, Flinn J. AutoBash: Improving configuration management with operating system causality analysis. In: Proc. of the 21st ACM Symp. on Operating Systems Principles (SOSP). 2007. [doi: 10.1145/1294261.1294284]

[45] Xiong Y, Hubaux A, She S, Czarnecki K. Generating range fixes for software configuration. In: Proc. of the 34th Int'l Conf. on Software Engineering (ICSE). 2012. [doi: 10.1109/ICSE.2012.6227206]

[46] Zhang S. ConfDiagnoser: An automated configuration error diagnosis tool for Java software. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). 2013. [doi: 10.1109/ICSE.2013.6606737]

[47] Yuan D, Zheng J, Park S, Zhou YY, Savage S. Improving software diagnosability via log enhancement. In: Proc. of the 16th Int'l Conf. on Architecture Support for Programming Language and Operating Systems (ASPLOS). 2011. [doi: 10.1145/1950365.1950369]

[48] Zhang JQ, Renganarayana L, Zhang XL, Ge NY, Bala V, Xu TY, Zhou YY. EnCore: Exploiting system environment and correlation information for misconfiguration detection. In: Proc. of the 19th Int'l Conf. on Architecture Support for Programming Language and Operating Systems (ASPLOS). 2014. [doi: 10.1145/2541940.2541983]

[49] Yuan D, Xie XL, Panigrahy R, Yang JF, Verbowski C, Kumar A. Context-Based online configuration-error detection. In: Proc. of the 2011 USENIX Annual Technical Conf. (ATC). Portland, 2011. 313-326.

[50] Rabkin A, Katz R. Using static analysis to diagnose configuration errors. In: Proc. of the 11th Int'l Symp. on Software Testing and Analysis (ISSTA). 2011.

[51] Lao N, Wen JR, Ma WY, Wang YM. Combining high level symptom descriptions and low level state information for configuration fault diagnosis. In: Proc. of the 18th Large Installation Systems Administration Conf. (LISA). 2004.

[52] Dong Z, Ghanavati M, Andrzejak A. Automated diagnosis of software misconfigurations based on static analysis. In: Proc. of the 2013 IEEE Int'l Symp. on Software Reliability Engineering Workshops (ISSREW). 2013. [doi: 10.1109/ISSREW.2013.6688897]

[53] Keller L, Upadhyaya P, Candea G. ConfErr: A tool for assessing resilience to human configuration errors. In: Proc. of the IEEE Int'l Conf. on Dependable Systems and Networks with FTCS and DCC (DSN). 2008. [doi: 10.1109/DSN.2008.4630084]

[54] Xia X, Lo D, Qiu W, Wang X. Automated configuration bug report prediction using text mining. In: Proc. of the IEEE 38th Annual Computer Software and Applications Conf. (COMPSAC). 2014. [doi: 10.1109/COMPSAC.2014.17]

[55] Jin D, Qu X, Cohen MB, Robinson B. Configurations everywhere: implications for testing and debugging in practice. In: Companion Proc. of the 36th Int'l Conf. on Software Engineering (ICSE Companion). 2014. [doi: 10.1145/2591062.2591191]

[56] Arshad FA, Krause RJ, Bagchi S. Characterizing configuration problems in Java EE application servers: An empirical study with GlassFish and JBoss. In: Proc. of the IEEE 24th Int'l Symp. on Software Reliability Engineering (ISSRE). 2013. [doi: 10.1109/ISSRE.2013.6698919]

[57] Meng FJ, Zhuo XJ. A generic framework for application configuration discovery with pluggable knowledge. In: Proc. of the IEEE 6th Int'l Conf. on Cloud Computing (CLOUD). 2013. [doi: 10.1109/CLOUD.2013.16]

[58] Mickens J, Szummer, Narayanan D. Snitch: Interactive decision trees for troubleshooting misconfigurations. In: Proc. of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SYSML). 2007.

[59] Attariyan M. Improving software configuration troubleshooting with causality analysis [Ph.D. Thesis]. University of Michigan, 2012.

[60] Wang HJ, Platt J, Chen Y, Zhang R, Wang YM. PeerPressure: A statistical method for automatic misconfiguration troubleshooting. Technical Report, MSR-TR-2003-80, 2003. http://research.microsoft.com/en-us/um/people/helenw/papers/msr-tr-03-80.pdf?0sr=ar

[61] Su YY, Flinn J. Automatically generating predicates and solutions for configuration troubleshooting. Technical Report, 2007. http://citeseerx.ist.psu.edu.sci-hub.org/viewdoc/download?doi=10.1.1.148.7963&rep=rep1&type=pdf

[62] Naik M. JChord. http://jchord.googlecode.com

[63] Nevill-Manning CG, Witten IH. Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research, 1997. 67-82.

[64] Sridharan M, Fink SJ, Bodik R. Thin slicing. In: Proc. of the ACM SIGPLAN 2007 Conf. on Programming Language Design and Implementation (PLDI). 2007.

[65] Chang CC, Lin CJ. LIBSVM: A library for support vector machines. ACM Trans. on Intelligent Systems and Technology (TIST), 2011,2(3):27. [doi: 10.1145/1961189.1961199]

[66] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987,32(1):57-95. [doi: 10.1016/0004-3702(87)90062-2]

[67] Kuhn DR, Reilly MJ. An investigation of the applicability of design of experiments to software testing. In: Proc. of the Annual NASA/IEEE Software Engineering Workshop (SEW). Los Alamitos: IEEE Press, 2002. 91-95. [doi: 10.1109/SEW.2002.1199454]

附中文参考文献:

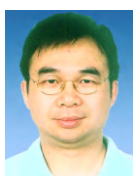
[25] 李福亮,杨家海,吴建平,安常青,姜宁.互联网自动配置研究.软件学报,2014,25(1):118-134. http://www.jos.org.cn/1000-9825/4458.htm

[30] 张路,谢冰,梅宏,邵维忠,杨芙清.基于构件的软件配置管理技术研究.电子学报,2001,29(2):1-3.

陈伟(1980－),男,江西高安人,博士,高级工程师,CCF 会员,主要研究领域为网络分布式计算,软件工程.

黄翔(1982－),男,博士,主要研究领域为网络分布计算,大数据.

乔晓强(1977－),男,博士,副研究员,CCF 会员,主要研究领域为网络分布计算,服务计算,软件工程.



魏峻(1970－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.

钟华(1971－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布式计算,软件工程.