基于资源依赖分析的资源相关配置项检测方法

李云峰 [1] **李姗姗** [1] **刘晓东** [1] **冯志敏** [1] **王智明** [1]
(国防科技大学 410073)[1]

摘要随着云计算和大数据的发展，越来越多软件的配置项数量都呈现爆炸式增长，软件通过不同的配置参数值获得期望的系统功能和性能特性。软件配置空间爆炸以及配置使用灵活已经成为制约软件性能的重要因素之一。很多研究致力于配置性能预测，利用机器学习的方法预测最优性能的配置组合，但是建库的开销不可接受，已有工作表明配置会影响软件性能，并且性能与资源联系紧密。因此，检测性能相关的配置项对软件性能的提升意义重大。文中提出配置项通过影响资源的方式，影响软件性能的观点，并实现一种基于资源依赖分析的资源相关的配置项检测方法，核心思想是检测配置项影响的语句中是否包含与资源相关的函数，检测与资源相关配置项可以有效减少通过调整配置优化软件性能的开销。文中实现了一个基本框架 RCDetect（Resource-related Configuration Option Detection），并应用于 3 个开源分布式软件。实验结果表明，RCDetect 能够检测与特定资源相关的配置项，并且平均正确率达到 70%以上。

关键词资源依赖分析，资源相关，配置

中图法分类号 TP311.5 文献标识码 A

Resource-related Configuration Option Detection Based on Resource Dependency Analysis

Yunfeng Li[1] Shanshan Li[1] Xiaodong Liu[1] Zhimin Feng[1] Zhiming Wang [1]
(National University of Defense Technology, 410073, China)[1]

Abstract With the development of cloud computing and big data, more and more software configuration options are exploding, and software obtains desired system functions and performance characteristics through different configuration parameter values. The explosion of software configuration space and the flexibility of configuration have become one of the important factors that restrict software performance. Many researches have focused on performance predictions, using machine learning methods to predict optimal performance configuration combinations, but the overhead of building a model is unacceptable, and prior studieshave shown that configuration can affect software performance, and performance is closely relatedto resources. So detecting performance-related configuration options is significant for software performance improvement. We propose the viewpoint that configuration options affect the usages of resources and further affect software performance, and we implement a resource-related configuration option detection method based on resource dependency analysis. The key idea is to detect whether the statements affected by the configuration options contain resource-related functions, detecting resource-related configuration options can effectively reduce the overhead of optimizing software performance by adjusting configurations. A basic framework RCDetect (Resource-related Configuration Option Detection) is implemented in this paper and applied to three open-source distributed software. The experimental results show that RCDetect can detect configuration options related to specific resources, and the average correct rate reaches 70%.

Keywords Resource dependency analysis, Resource-related, Configuration

1 引言

随着软件规模的不断扩大，软件配置项数量也不断增加，大量的配置参数可以使得软件更加灵活以适用不同的场景，不同的环境。目前，包括数据库，Web 服务器，编译器，分布式软件等在内的软件，配置项对软件的功能属性和非功能属性都有着不同的影响[1]。但是配置空间的不断增大，配置项之间的约束不断增多，且软件配置与性能之间的联系更加紧密，也对开发人员和用户正确配置软件带来了巨大的挑战。用户通常使用默认配置或者只改动一个配置项以获得期望

到稿日期：返修日期：
本文受国家自然科学基金项目（61872373 和 61872375）资助。
李云峰（1993-），男，学士，主要研究方向为软件可靠性，E-mail：liyunfeng12@nudt.edu.cn。

性能[1]。即使拥有领域知识的专家和开发人员也不能了解配置和软件性能之间的具体关系。

针对上述问题，许多研究致力于预测性能较优的配置组合[1-3]。机器学习的方法构建配置和软件性能模型，即使使用了不同的启发式采样方法，但是建库的开销往往不可接受。许多研究[4-6]表明，配置会影响软件性能。本文提出配置影响软件资源，从而影响软件性能的观点。为了深入理解配置与软件资源的关系，本文研究了 5 个开源软件（Hadoop，HDFS，MapReduce，Yarn，ZooKeeper）的 100 个配置项，发现配置项通过影响资源相关的函数（例如，malloc 等）影响软件资源的使用。如图 1 所示，dfsthroughput.buffer.size 在源码中的存储变量是 BUFFER_SIZE，然后 BUFFER_SIZE 被作为参数调用申请缓存 data，如果 BUFFER_SIZE 越大，那么该缓存区所占用的内存将会更高，该配置项对软件的内存资源产生影响。

```
/*org\apache\hadoop\hdfs\BenchmarkThroughput.java*/
public int run(String[] args){
   BUFFER_SIZE = conf.getInt(
                "dfsthroughput.buffer.size", 4 * 1024);
   ……;
}

private Path writeLocalFile(...){
   ……;
   byte[] data = new byte[BUFFER_SIZE];
}
```

图 1 配置影响资源示例

基于配置项影响软件资源的观点，本文提出一种基于资源依赖分析的资源相关配置项检测方法。检测资源相关的配置项，可以减少开发人员和用户优化解决性能问题的开销，通过调整资源相关的配置项以满足用户对软件性能的需求。该方法的核心是检测配置影响的否包含资源相关的语句。首先，本文利用静态分析方法，获取配置项影响的语句；然后，基于模式匹配检测配置项影响的语句中是否包含资源相关的语句；最后，提取与资源相关的配置项。

但是，仍要需要面对以下挑战。第一，不同类型的配置项使用方式不同，难以获取切片的 seed 语句（即配置项的读取函数）；第二，难以判断一条语句是否与资源相关；第三，在没有第三方库函数源码的情况下，判断该函数是否与资源相关具有很大的挑战。

针对以上挑战，本文利用基于函数名及参数分析的方法鉴别配置项的读取函数；分析 5 个开源软件的源码，从中分析提取与资源相关的语句，作为基本的资源相关函数库；针对第三方库函数，本文使用函数名关键字匹配的方法判断第三方库函数是否与资源相关。

本文研究了配置与软件资源之间的关系。结果表明配置影响软件资源的使用，并提出一种基于资源依赖分析的资源相关配置项检测方法。实现了 RCDecet 的基本框架，并初步验证了其有效性。将 RCDecet 应用于 3 个开源软件，结果表明，RCDecet 检测资源相关的配置项准确率达到 70%以上。

本文组织结构如下：第二章介绍配置于软件性能领域相关工作；第三章介绍配置影响软件资源使用观点；第四章介绍 RCDetect 的体系结构和设计实现；第五章将 RCDetect 应用于 3 个软件以验证 RCDetect 的准确率。第六章总结本文工作并说明未来研究方向。

2 相关工作

2.1 优化配置预测

基于机器学习的方法主要是在不同配置组合获得软件的性能指标，并利用机器学习的方法将配置与软件系统性能（响应时间，网络等）建模，描述配置和软件性能之间的关系，并结合不同的采样算法，有效的减小训练样本的开销并精确的预测配置。主要包括 Norbert Siegmund、JianmeiGuo、Sven Apel 等人[1-3]提出的使用不

同的采样算法和机器学习算法。Norbert Siegmund[1]等人提出将配置项和软件性能构建表达式，并首次将 bool 类型的配置项和数值类型的配置项结合起来，使用皮克布莱曼采样方法和梯度线性回归算法预测软件性能最好的配置组合。JianmeiGuo[2]提出了一种基于随机采样的渐进采样和基于变量的软件系统性能预测方法，该方法使用统计学习技术 CART 来构建一个显式的性能模型，该模型表示 feature 选择与性能之间的相关性。Atri Sarkar、JianmeiGuo[3]等人研究渐进采样和投影采样对可配置软件系统的性能预测的影响以及研究 T-way 和频率特征作为启发式之间的差异。

2.2 配置故障检测

许多研究都致力于错误配置，但并未关注软件性能问题。之前的许多工作都提出使用静态程序分析或统计分析[4-6]来识别和修复不正确或异常的配置。这些技术主要针对与功能相关的错误配置，并且不适用于性能问题，因为性能问题的适当设置在很大程度上取决于动态工作负载和环境，并且很难根据常见/默认设置进行统计判断。还提出了用于诊断错误配置故障和与配置错误相关的性能问题的技术。

3 分析与调研

很多研究致力于软件系统的性能问题，配置项会影响软件性能[4-6]，且 59%的性能问题均与配置相关，为了深入理解配置与软件性能之间的关系，本文深入研究 5 个开源软件随机选取的 100 个配置项，发现配置影响软件资源的使用。

3.1 数据集

本文主要研究 Hadoop，HDFS，MapReduce，Yarn，ZooKeeper 等 5 个 Java 开源分布式软件系统[7]。这些分布式软件是搞配置软件的典型代表，每一种软件的发展历程均在十年以上。为了深入理解配置与资源之间的关系，本文研究配置项及其所影响的源代码。如表 1 所示，本文从每个软件随机选取 20 个配置项，研究配置项如何在代码中影响软件资源使用。

表 1 软件及选取配置项

软件	配置项数量	软件描述
Hadoop	20	分布式系统基础架构
HDFS	20	分布式文件系统
MapReduce	20	用于并行处理大数据集的软件框架
Yarn	20	Hadoop 资源管理器
ZooKeeper	20	分布式的，开放源码的分布式应用程序协调服务

3.2 配置项如何影响软件资源使用

为了研究配置如何软件资源使用，本文研究配置项及其所影响源代码。本文发现这些配置项总是与资源相关的函数有关（例如，socket()，String()等），配置项所影响的代码块通常是调用与资源相关的函数，资源相关的函数调用内核函数获取响应的软件资源。例如图 1 所示，缓存区 data 由 BUFFER_SIZE 控制大小，data 调用 byte()函数，申请 BUFFER_SIZE 大小的内存，JVM 虚拟机分配 BUFFER_SIZE 的内存给 data，配置影响软件内存使用的大小。

```
/*org\apache\zookeeper\server\persistence\
FileTxnLog.java*/

String size =
        System.getProperty("zookeeper.preAllocSize");
preAllocSize = Long.parseLong(size) * 1024;

public static long padLogFile(FileOutputStream f,long
currentSize,
        long preAllocSize) throws IOException{
    ……;
    if (position + 4096 >= currentSize) {
      currentSize = currentSize + preAllocSize;
      ……;
      f.getChannel().write(fill, currentSize-
fill.remaining());
    }
    return currentSize;
```

图 2 单个配置项影响磁盘

```
/*org\apache\zookeeper\
server\SyncRequestProcessor.java*/

public void run(){
  ……;
  int randRoll = r.nextInt(snapCount/2);
  while (true) {
      ……;
     logCount++;
     if (logCount > (snapCount / 2 + randRoll)) {
        randRoll = r.nextInt(snapCount/2);
        zks.getZKDatabase().rollLog();
        if (snapInProcess != null &&
snapInProcess.isAlive()) {
            LOG.warn("Too busy to snap, skipping");
        } else {
            snapInProcess = new
ZooKeeperThread("Snapshot Thread") {
            ……;
        };
        snapInProcess.start();
     }
}
```

图 3 单个配置项影响 CPU

通过研究配置与资源的关系，本文发现资源使用可能受到单个配置项影响或多个配置项联合影响。单个配置项影响资源的使用占比达到 85%。例如，如图 2 所示，preAllocSize 是 ZooKeeper 配置项之一，当开始写事务日志时，server 每次都会分配这个配置项指定的大小的磁盘块。padLogFile 调用 preAllocSize，通过 f.getChannel().write() 函数写磁盘，影响 ZooKeeper 服务器占用磁盘空间。如图 3 所示，snapCount 是两次事务快照之间可执行事务的次数，默认值为 100000。如果 logCount > (snapCount / 2 + randRoll，那么就会启动 snapInProcess 线程进行快照操作，快照会影响软件性能，占用 CPU 资源和磁盘。两个及以上配置项影响软件资源占比达到 15%。例如，如图 4 所示，end = start + self.getInitLimit（）* self.getTickTime（），其中 getInitLimit 获取配置项 initLimit 的值，getTickTime 获取配置项 tickTime 的值，start 是系统时间的初始值。当 electionFinished 设置为 false 时，循环等待 cur> end。这两个配置项影响循环次数，并进一步占用更多 CPU 资源。

```
/*zookeeper/server/quorum/leader.java*/

public void waitForNewLeaderAck(…){
  ……;
  long start=System.currentTimeMillis();
  long cur=start;
  long end=
       start+self.getInitLimit()*self.getTickTime();

  while(!electionFinished&&cur<end){
    electingFollowers.wait(end-cur);
    cur=System.currentTimeMillis();
  }
  ……;
}
```

图 4 两个配置项影响 CPU

```
/*java/org/apache/hadoop/security.java*/

public void GroupCacheLoader() {
  ……;
  if (reloadGroupsInBackground) {
     ThreadFactory threadFactory = ...;
    ThreadPoolExecutor parentExecutor = new
          ThreadPoolExecutor(
               reloadGroupsThreadCount,...);
  ……;
  }
}
```

图 5 配置项间接影响资源

根据配置影响资源的方式，可以分为以下两种：（1）配置项直接影响软件资源；（2）配置项间接影响软件资源。73%的配置项直接影响软件按资源。例如，图 1 所示，dfsthroughput.buffer.size 作为参数被 byte()调用以申请 data 缓存，改配置项直接作用于内存资源。例如，图 2 所示，preAllocSize 直接影响 if 判断条件，进一步影响磁盘写入。这一类配置项直接影响软件资源，不受到其他配置项的影响。图 27%的配置项间接影响软件资源。例如，图 5 所示，reloadGroupsInBackground 通过控制 reloadGroupsThreadCoun 间接影响资源的使用，

reloadGroupsThreadCoun 控制并发后台用户组缓存条目刷新的数量。如果 reloadGroupsInBackground 为真且 reloadGroupsThreadCoun 较大，则会影响更多资源并导致性能问题。

4 RCDetect 结构与实现

本章主要描述 RCDetect 的基本框架和实现细节，RCDetect 的目标是针对特定类型的资源，检测与该资源相关的配置项。

4.1 RCDetect 体系结构

本文匹配配置项所影响的所有语句中是否包含资源相关的语句（例如，内存相关 string()等），设计并实现了 RCDetect 的基本框架。RCDetect 主要包含以下 2 个模块：

（1）配置项影响的语句获取，该模块以字节码和配置文件为输入，通过切片方法得到配置项所影响的代码语句；

（2）资源相关语句匹配，该模块匹配配置项影响的语句中是否包含资源相关的配置项；

4.2 RCDetect 框架设计与实现

本文设计并实现 RCDetect。RCDetect 首先获取配置项影响的语句，然后匹配所有语句中是否含有与资源相关的语句以判断该配置项是否与资源相关。

4.2.1 配置影响的语句获取

本文根据配置影响的语句中是否含有资源相关的语句判断该配置项是否与资源相关。因此，需要获取该配置项影响的所有语句。本文使用切片技术 WALA[8]，一种过程间静态切片方法，能够通过指定控制依赖项和数据依赖项获取不同粒度的配置项直接影响和间接影响的语句。

首先，需要指定切片的 seed 语句，通过对 Hadoop 等软件的调研，发现这些软件通过指定类读取配置项，并且每一个配置项对应 get/set 函数，get 函数获取配置项的值，set 函数更新配置项的值，并且这些配置项遵循着很好的模式。例如，图 1 所示，conf.getInt(dfsthroughput.buffer.size, …)函数获得 dfsthroughput.buffer.size 的值。本文研究源码中配置项的使用方式，总结以下两种配置项 get 函数的模式：（1）get+配置名，（2）get+配置项类型(配置名)。基于函数名及参数分析方法，获得配置的 get 函数，从而获得 seed 语句。

4.2.2 资源相关语句匹配

基于配置项影响的语句，本文利用文本匹配的方法检测配置项影响的语句中是否包含与资源相关的语句。本文分析 Hadoop 等软件的源代码，通过分析软件源码不同的 API 和基本库中 API 的具体实现方法，提取与资源相关的 API。表 3 列举部分与资源相关的函数。由于切片得到大量的语句集合，因此，本文根据经验值设定一个 P 值，该值表示与资源相关函数的数量 S，当 S>P 时，则认为该语句与资源相关；P 值越大，则该语句与资源相关程度越高。

表 3 部分资源相关的函数

资源	函数
内存	String(), byte(), …
网络	Socket(), url, …
磁盘	File.write(), File(), …
…	…

如果该语句是第三方库函数或来自不同语言的源文件，切片技术则不能获得对应的所有语句。例如，ZooKeeper 中有部分 C++代码，但是，本文只针对 Java 代码，当 Java 调用 C++文件时，导致切片失效，不能获得 C++中文件中与资源相关的语句。因此，本文采用以下方法处理（如图 6 所示）：（1）如果该第三方库存在，则直接分析该包种对应的源代码是否存在资源相关的函数即可，进行迭代判断；（2）如果不存在或者调用其他语言的函数，则基于关键字匹配，判断

该函数名是否包含资源相关的关键字，以提高 RCDetect 的精确度。例如，如果函数名种包含 socket，string，memory 等关键字，那么该函数很有可能与资源相关（5.4 节评估该方法的准确率及影响程度）。

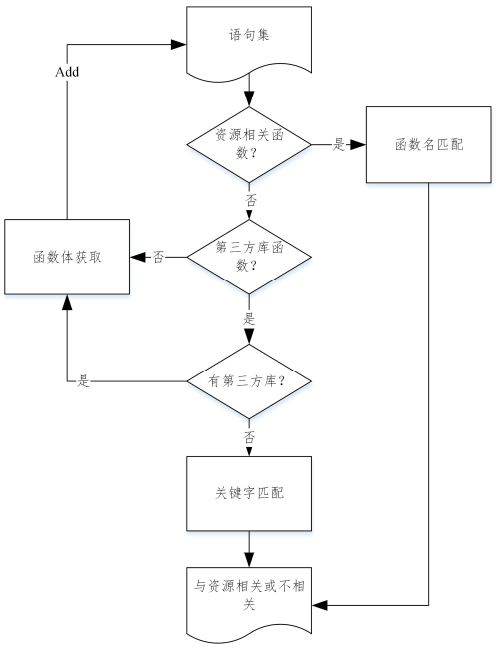

图 6 第三库函数是否与资源相关判定

5 实验评估

本文将 RCDetect 的初步框架应用于 HDFS，Hadoop，MapReduce，初步验证 RCDetect 是否可以有效地检测资源相关的配置项，以及验证配置项使用识别的准确率。本文以内存资源为例，其他实验结果及部分数据可访问 https://github.com/a944982774/RCDetect。

5.1 实验环境

表 4 硬件及软件设置

CPU	Intel i5 7200U
内存	8GB
磁盘	120GB SSD
操作系统	Windows 10
IDE	Eclipse

本文以 HDFS，Hadoop，MapReduce 的字节码包.jar 文件和对应的配置文件 hdfs-siite.xml、core-site.xml 和 mapred-site.xml 为输入，版本号均为 2.6.5。实验机器硬件及软件环境如表 4 所示。

5.2 RCDetect 有效性

本文将使用 RCDetect 检测 HDFS，Hadoop，MapReduce 中所有与内存相关配置项。以配置文件和.jar 文件为输入，P 值为 20（不同 P 值实验结果可通过 github 访问），得到内存相关的配置项实验结果如表 5 所示。

表 5 资源相关配置项准确度

软件	#配置项	内存相关	准确度
HDFS	366	62（44）	70.5%
Hadoop	246	37（27）	73.3%
MapReduce	181	27（20）	75.0%

为了验证 RCDetect 检测内存相关的配置项准确度，本文人工通过分析配置项及其所影响的源代码和阅读相关配置文件以及官方文档验证了其准确度，并且由 3 名经验丰富的本科生交叉验证，有 2 人以上确定为配置相关则认定该配置项与资源相关。经人工验证，RCDetect 可以有效检测 44，27，20 个内存资源强相关的配置项。准确度达到了 70.5%，73.3%，75.0%。RCDetect 的准确度较低，主要受到以下 3 个因素的影响。第一，人工验证的主观性，3 人交叉验证仍然会带来漏报和误报；第二，代码中字符串的频繁使用，增加了误报；第三，切片技术不能获得不同语言的语句。同时，本文分析所得到的配置项与内存不相关的配置项的类型，主要是字符串类型的配置项（60%），其余类型为枚举（25%），数值类型等。其中，字符串类型主要规定了软件的一些信息。例如，软件版本信息，服务器的 IP 等。

5.3 配置项 get 函数识别的准确率

为了验证本文工作的正确性，需要验证 RCDetect 识别配置项 get 函数准确率。因此，本文分析 RCDetect 所检测到的资源相关配置项的准确率和召回率。

本实验将 RCDetect 应用于 HDFS，Hadoop，MapReduce。实验结果如表 6 所示。配置项 get 函数的召回率在分别为 73.5%，67.1%，76.8%，本文分析识别召回率较低主要有以下 3 种原因。第一，不是所有的配置项都在源代码中使用，并不是所有的配置项在源码中都有特定的处理函数和存储的变量；第二，部分配置项的 get 函数不符合本文总结的两种模式，部分配置项直接以字符串读入存储到变量中，不经过 get 函数使用，这一类主要是 string 类型的配置项；第三，部分配置项没有从特定类中读入。召回率越高，实验的结果越准确，低召回率导致部分配置项（≈25%）无法进行切片，从而无法判断该部分配置项是否与资源相关，在配置读入方面仍需要进一步研究。

表 6 配置读入有效性（#配置项：配置项的数量；所有项：RCDetect 可识别入口的配置项数量；正确项：所有项中正确识别的配置项数量；召回率：所有项/#配置项；准确率：正确项/所有项；）

软件	#配置项	Get 函数		召回率	准确率
		所有项	正确项		
HDFS	366	269	245	73.5%	91.1%
Hadoop	246	165	144	67.1%	87.3%
MapReduce	181	139	128	76.8%	92.1%

同时，为了验证 RCDetect 的准确率，即 RCDetect 识别的 get 函数中是否是该配置项的 get 函数，本文人工分析每个配置项对应的 get 函数，准确率如表 6 所示。可以发现，RCDetect 在识别准确度上基本上达到 90%，即表明使用基于函数名分析的方法可以准确识别配置项 get 函数。

5.4 第三库函数资源相关准确率

为说明第三库函数对结果的影响。本文按比例分别从 HDFS、Hadoop、MapReduce 中随机选取的 8、4、4 个配置项，并通过计算发现每个配置项切片结果中第三方库语句占比为 5%-15%。本文分别从三个软件中随机选取 100 条第三方库（运行库中）语句作为研究，分析第三方库对实验结果的影响。

表 7 准确率（N：选取的第三方库函数数量）

软件	N	资源相关	关键字匹配	召回率
HDFS	100	26	18	69%
Hadoop	100	23	13	56%
MapReduce	100	31	18	58%
总计	300	80	49	61%

结果如表 7 所示，本文通过关键字匹配得出的资源相关函数，得到 18，13，18 个资源相关的函数。并验证资源相关函数的数量，从而分析关键字匹配对整体结果的影响。第三方库函数关键字匹配技术，可以从 80 个资源相关的函数中发现 49 个资源相关的函数，召回率达到 61%。

由于第三方库语句占比为 5%-15%，那么第三方库函数识别的召回率 59%对实验结果的影响为 10%*（100/320）*（59/100）=1.85%。总体来说，第三方库语句对总体实验结果的影响偏小。

6 结束语

配置影响软件性能，本文提出配置影响软件资源的使用，从而影响软件性能。本文验证了该观点，并提出一种基于资源依赖分析的资源相关的配置项检测方法，设计并实现了 RCDetect 的基本框架，并将 RCDetect 应用于 3 个分布式应

用。实验结果表明，RCDetect 能够有效检测与资源相关的配置项，准确率达到 70%以上。同时，本文验证了 RCDetect 识别配置项 get 函数的准确率，准确度达到 90%以上。

但是 RCDetect 仍然存在以下问题：（1）切片不能获得第三库的源代码和不同语言的语句，使得 RCDetect 准确度较低；（2）配置读入识别召回率较低；（3）RCDetect 的实用性和可靠性需要进一步验证。

针对 RCDetect 的问题，本文未来将主要研究以下三个方面：（1）切片获取第三库源代码的方法以及跨语言的切片技术；（2）配置项 get 函数识别的召回率偏低，配置项 get 函数识别受到 5.3 节中所述多种因素影响，有效提高其召回率将会进一步提高整个工作的准确度；（3）进一步验证 RCDetect 框架的有效性和实用性，本文初步验证了 RCDetect 的有效性，但仍需进一步将 RCDetect 应用于更多的软件。

致谢

感谢我的导师李姗姗老师的指导。本文受国家自然科学基金项目(61872373 和 61872375）资助。

参考文献

[1]Siegmund N, Grebhahn A, Apel S. Performance-influence models for highly configurable systems[C]// Joint Meeting. 2015:284-294.

[2]Siegmund N, Kolesnikov S S, Apel S, et al. Predicting performance via automated feature-interaction detection[C]// International Conference on Software Engineering. IEEE, 2012:167-177.

[3]Sarkar A, Guo J, Siegmund N, et al. Cost-Efficient Sampling for Performance Prediction of Configurable Systems (T)[C]// Ieee/acm International Conference on Automated Software Engineering. IEEE, 2016:342-352.

[4].Han X, Yu T. An Empirical Study on Performance Bugs for Highly Configurable Software Systems[C]// ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, 2016:23.

[5].Yan C, Cheung A, Yang J, et al. Understanding Database Performance Inefficiencies in Real-world Web Applications[C]// ACM on Conference on Information and Knowledge Management. ACM, 2017:1299-1308.

[6].Wang S, Li C, Sentosa W, et al. Understanding and Auto-Adjusting Performance-Related Configurations[J]. 2017.

[7].http://www.apache.org/

[8].http://wala.sourceforge.net/wiki/index.php/Main_Page].

[9]Rabkin A, Katz R. Precomputing possible configuration error diagnoses[J]. 2011, 2012(4):193-202.

[10]Xu T, Jin X, Huang P, et al. Early detection of configuration errors to reduce failure damage[C]// Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2016:619-634..

[11]Xu T, Jin L, Fan X, et al. Hey, you have given me too many knobs!: understanding and dealing with over-designed configuration in system software[C]// Joint Meeting on Foundations of Software Engineering. ACM, 2015:307-319.

[12]Nagaraja K, Oliveira F, Bianchini R, et al. Understanding and dealing with operator mistakes in internet services[C]// DBLP, 2004:61-76.例：W3C.Web Service Choreography Interface(WSCI)(Version1.0)[EB/OL].www.w3.org/TR/wsci.