·韧性信息系统构建专题·　　doi：10.15908/j.cnki.cist.2018.06.006

基于数据分析的云监控系统*

李志娟　吴卓霏　王思家　赵　靖

（哈尔滨工程大学计算机科学与技术学院　哈尔滨 150001）

摘　要：为了提高云平台及其服务的韧性，设计了具有异常检测和预测功能的云监控系统。针对基于阈值异常检测方法存在的误警率过高，以及无法区分持续高负载和短暂爆发流的问题，采用了基于统计的异常检测算法，提高了检测精度，降低了误警率。为了实现故障发生前的告警，提出了基于经验模态分解（EMD）的时间序列预测算法，对未来一段时间的数据进行预测。通过在 OpenStack 云平台上部署 TPC-W 应用和监控系统试验表明，该系统可准确发现异常，其预测精度高于传统时间序列预测精度。

关键词：云监控系统；异常检测；时间序列预测

中图分类号：TP277　　**文献标识码**：A　　**文章编号**：1674-909X(2018)06-0030-06

Cloud Monitoring System Based on Data Analysis

LI Zhijuan　WU Zhuofei　WANG Sijia　ZHAO Jing

(School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract: To improve the resilience of cloud platform and its services, the cloud monitoring system with abnormal detection and prediction functions is designed. Aimed at the problem of the high false alarm rate of the threshold-based abnormal detection method and the problem of inability to distinguish the continuous high load and the transient burst flow, the statistical-based abnormal detection algorithm is used to improve the detection accuracy and reduce the false alarm rate. To achieve the pre-failure alarm, the time series prediction algorithm based on the empirical mode decomposition (EMD) is proposed, it can predict the future data. The experimental of deploying TPC-W application and monitoring system on the OpenStack cloud platform shows that the system can accurately detect abnormity and its prediction accuracy is higher than the traditional time series prediction.

Key words: cloud monitoring system; abnormal detection; time series prediction

0　引　言

云平台通常由大量的物理机、高速大容量存储设备、高速网络和虚拟机实例组成，可为用户提供社交网络、内容分发和大数据处理等多种云服务及应用，且这些服务及应用具有不同的组成、配置和部署需求[1]。云平台及其服务的持续可用建立在所有软硬件无故障的前提下，涉及服务器与云环境的部署、应用程序开发、产品逻辑设计、用户行为以及基础网络构建等。由于电子元件或线路老化会引起硬件缺陷[2]，软件开发设计过程中也会引进各种缺陷等[3-4]，因此云平台及其服务状态将随时间推移发生

*基金项目：装备发展部"十三五"预研课题资助项目。

收稿日期：2018-10-10

引用格式：李志娟，吴卓霏，王思家，等. 基于数据分析的云监控系统[J].指挥信息系统与技术，2018，9(6)：30-35.

LI Zhijuan, WU Zhuofei, WANG Sijia, et al. Cloud monitoring system based on data analysis[J]. Command Information System and Technology, 2018, 9(6): 30-35.

不确定变化。

随机过程可对不确定性下系统状态给出数学描述。平稳过程可描述稳定状态下的系统及其服务状态，即主要统计特性（如均值和方差）将不随时间推移而改变。部分系统状态的分布偏离其正常分布称故障[5]。暂态过程是从一个稳定状态到另一个稳定状态经历的过程。如果系统发生故障，故障引起的暂态将导致系统状态的均值或方差改变，该现象可作为检测故障的依据。失效不同于故障，失效指系统作为一个整体停止向用户提供服务。

故障或失效的概率不可能降至 0，较好办法是设计容错机制以防因故障而导致失效。能预料并应对故障的系统特性称容错或韧性[6]，故可将韧性作为衡量云平台及其服务可靠性的一项关键指标，即面对各种故障和威胁时，云平台及其服务应提供并保持用户可接受的服务等级的能力。韧性具体体现在以下 4 个方面：1）高可用：采用分布式架构规避单点故障；2）网络安全：当攻击者寻找到机会绕过防范手段时，系统能够根据收集的安全日志和流量等信息，及时检测出异常，识别攻击路径，并快速做出阻断和恢复响应；3）服务质量：从用户角度看，服务响应时间无明显减少；4）优雅降级：保障核心业务永续在线和关键业务数据安全。

本文设计了一套监控系统检测并预测系统中存在的故障，以提高云服务系统的韧性。该系统将云平台中的存储、虚拟机、网络、服务和任务等均视为资源，从而对各类资源数据进行采集和分析。

1　云平台监控系统架构

本文设计了云平台监控系统，其架构如图 1 所示。其中，UDP 为用户数据报协议；HTTP 为超文本传输协议。系统包括采集器、收集器、分布式消息系统Kafka[7]和基于Spark[8]集群的异常检测与趋势预测等模块，系统数据流程如下：1）采集器模块获取资源数据，并将资源数据及其他相关属性（如主机名、资源项和采集时间点等）组织成 JavaScript 对象简谱（JSON）格式的数据，发送给收集器；2）收集器将收到的数据转发给 Kafka；3）异常检测模块从 Kafka 中读取数据检测系统异常；4）趋势预测模块从 Kafka 中读取数据对资源进行趋势预测。

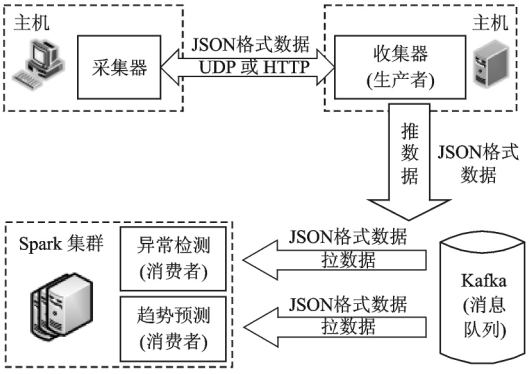


图 1　云平台监控系统架构

2　关键技术及实现

2.1　数据采集技术

数据采集模块在宿主机上运行。由于监测数据来自分布式与异构的硬件、软件和网络等，种类繁多，且监测数据的采集方式和周期等均影响资源的异常分析，故设计监测策略如下：通过修改配置文件参数和自定义监控脚本等策略，有针对性地设置监控项及其监测周期等，以便实现监控项和监控粒度的可调节。

2.1.1　数据采集器

数据采集器由采集主进程和各类脚本组成，其核心是各种用于获取资源数据的脚本程序，根据应用特点制定采集策略，包括采集指标项和采集周期。其中，采集指标项包括主机的 CPU 使用率、内存和网卡吞吐量等。针对 Web 应用，可编写脚本获取 Java 虚拟机（JVM）堆内存使用量、JVM 堆内存总量及用户请求页面响应时间；针对 MySQL 数据库服务，可编写脚本获取数据库连接数等。数据采集器架构如图 2 所示，采集脚本置于以采集周期命名的文件夹，采集主进程周期性地启动、关闭和更新采集脚本。采集脚本获取的数据输出到标准输出（stdout），采集主进程从 stdout 中读入采集数据，并组织成以下 JSON 格式数据：{"metric"：metricname，"host"：hostname，"value"：data，"timestamp"：timestamp，"tag"：tagvalue}。其中，metric 为指标项名称；host 为主机名；value 为采样数据；timestamp 为采样时间点；tag 为自定义的标识，可有多个。

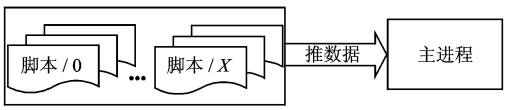


图 2　数据采集器架构

2.1.2　数据收集器和消息队列

数据收集器作为采集器和 Kafka 间的中间件，负责将数据采集器的数据发送给 Kafka（见图 1），解决了任一云主机均需安装 Kafka 驱动的问题，实现

了监控系统的通用性和可扩展性。数据收集器构建支持表现层状态转移(Restful)接口的 HTTP 或 UDP 服务器,数据采集器使用 httpclient 或 udp-socket 将数据发送到数据收集器。数据收集器主机安装 Kafka 驱动,可将数据通过 Kafka 服务器(Kafka Broker)发送给指定的消息主题 Topic,其中多个 Broker 连接到相同 Zookeeper 组成的 Kafka 集群。

2.2 异常检测技术

过载异常指云平台中的资源使用量、服务响应时间及服务连接数等指标超过可接受范围的现象。基于阈值的方法为传统过载异常检测方法,存在误警率高、识别率较低以及无法区分持续的高负载和短暂的爆发流等问题。为了解决上述问题,本文采用基于统计的异常检测方法。

文献[9]提出了基于统计的过载异常检测算法,又称桶算法。该算法先收集并计算正常情况下待测指标的均值 μ_x 和方差 σ_x,再对待测指标的 n 个连续测量的样本取值,最后对 n 个样本的均值和目标值进行比较,判断是否发生异常。该算法可消除系统中出现瞬期波动偏离较大的待测指标所带来的影响,使检测到的少量显著偏离的较大的值被很好地"平滑"掉。基于统计的过载异常检测原理如图 3 所示。

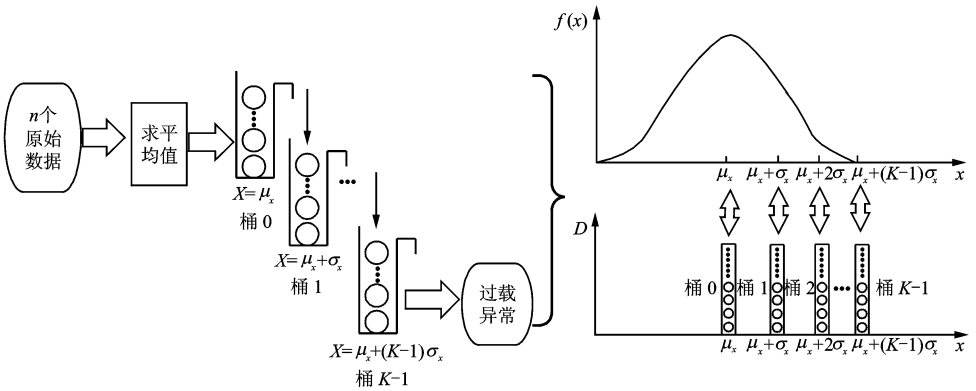


图 3 基于统计的过载异常检测原理

该算法包括 n、K、D、μ_x 和 σ_x 共 5 个参数。其中,n 为连续检测到的样本个数,为了方便描述,参数 K 和参数 D 分别表明桶的数目和深度。假设有 K 个桶,且每个桶的深度为 D,用 N 代表桶的索引,$N=0,1,2,\cdots,K-1$。从第 1 个桶开始,对于检测到的 n 个样本均值存在 $\overline{x}_n>\mu_X+N\sigma_X$ 时,向第 N 个桶中放入 1 个球;相反,若 $\overline{x}_n\leqslant\mu_X+N\sigma_X$ 时,则从该桶中取出 1 个球。当桶中的球超过了桶的容量时,称上溢,并将无法放入的球放入第 $N+1$ 个桶中。若当前桶 $N(N>0)$ 已空,且又发生 $\overline{x}_n\leqslant\mu_X+N\sigma_X$ 时,称下溢,并将已装满球的第 $N-1$ 个桶中的球移出 1 个。当最后 1 个桶发生上溢时,则判定待测指标异常。

离散流(DStream)作为 Spark Streaming 的一个基本抽象,提供高层应用程序编程接口(API)进行 Spark Streaming 程序开发[8]。DStream 代表了一系列连续的弹性分布式数据集(RDD),DStream 中每个 RDD 均包含特定的时间间隔数据,对 DStream 的操作最终将转换成底层 RDD 操作。Spark Streaming 编程步骤如下:1) 创建输入的数据流 DStream;2) 对 DStream 进行各种算子操作,得到新的 DStream;3) 将处理结果存储于存储介质。基于 Spark 的桶算法实现如图 4 所示。其中,虚线表明异常检测程序从指定 Kafka Broker 中读取数据,形成原始 DStream;方框表明 1 个时间间隔内的数据(设为 1 s)形成 1 个 RDD。

首先,原始 DStream 中的 RDD 包含原始的 JSON 格式数据,利用 map 函数对 RDD 中的数据进行归类,形成转换后的 DStream,数据格式如下:((hostip, metricname),[isAbnormal, (N, D), orgJsonValue])。其中,hostip 为主机地址;metricname 为指标项名称;isAbnormal 为是否发生异常;

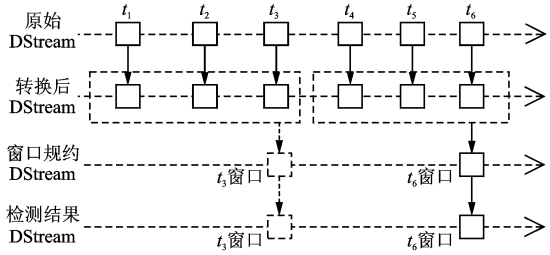


图 4 基于 Spark 的桶算法实现

orgJsonValue 为原 JSON 格式数据。然后，采用 reduceByKeyAndWindow 函数对同一指标的若干采样数据求均值，本文设定该函数的窗口和滑动时间均为 3 s，如图 4 中窗口规约 Dstream 所示。最后，用 updateStateByKey 函数对窗口规约 DStream 中的数据进行统计分析，判断是否发生异常，形成检测结果 DStream。

2.3　趋势预测技术

云服务系统中的任务一旦发生故障，将会带来很大的系统开销和人力物力资源的浪费，如果能预知系统中的哪些资源会发生过载异常、哪些服务会面临失效威胁，就能采取预防措施避免故障发生。因此，本文使用基于时间序列分析方法进行资源趋势预测，以便在故障发生前进行告警，并采用经验模态分解(EMD)＋时间序列分析的方法提高预测精度。

EMD 算法可将成分复杂和性能不好的时间序列分解为一系列性能良好的本征模函数(IMF)和残差项 r。由于 IMF 为平稳数据序列，基本满足差分整合移动平均自回归模型(ARIMA)的使用要求，因而通过 ARIMA 模型对各 IMF 进行预测可得较高预测精度。此外，经过多次分解，原始数据残差项 r 的性质也越来越简单，使用多项式即可较好地拟合。将各分量预测结果合并即可得到原始时间序列的预测结果。EMD-ARIMA 算法解决了 ARIMA 模型适用性的问题，在时间开销较小的情况下实现了较高精度。

EMD-ARIMA 算法原理示意图如图 5 所示，描述如下：1) 依照 EMD 原理对观察值序列进行分解，获得若干个较平稳的 IMF 分量和残差项；2) 对每个 IMF 分量和残差项进行平稳性检测和纯随机性检测，一般情况下，分解初期的 IMF 分量噪声干扰较严重，通过序列的相关系数图可判断分量是否具有物理意义；3) 去除无意义的分量，保留余下 IMF 分量和残差项，求和重构出不含噪声干扰的新序列；4) 基于无噪声干扰序列建立相应的时间序列模型，检验拟合模型的有效性，并用有效模型预测出结果。

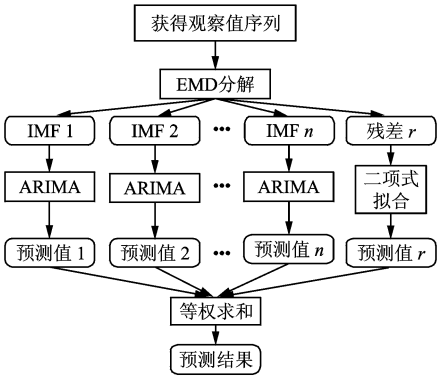


图 5　EMD-ARIMA 算法原理示意图

3　试验分析

本文采用开源的 OpenStack[10] 部署云平台，在虚拟机上部署购书网站作为被监控应用程序，同时采用注入内存泄漏方法加速内存使用。

3.1　测试环境

在 1 台物理服务器上搭建 OpenStack 云平台，物理服务器配置为 128 GB 内存、10 TB 存储和 24 核处理器，OpenStack 由 Nova、Glance 和 Neutron 等组件组成。其中，Nova 组件管理计算资源；Glance 组件管理镜像资源；Neutron 组件管理网络资源。首先，创建 2 台 Web 应用虚拟机实例，Web 应用程序由 Web 服务器、数据库服务器和 1 组客户端组成，数据库和 Web 服务器位于同一台虚拟机上，而所有客户机占用另一台虚拟机，同时在 Web 服务器所在虚拟机上启动采集器；然后，创建 1 台 Kafka 虚拟机实例，Kafka 应用程序由 Kafka 和 ZooKeeper 服务器组成，同时在 Kafka 虚拟机上启动收集器；最后，创建 2 台虚拟机实例组成 Spark 集群，1 台为控制节点，1 台为计算节点，作业调度用 Hadoop Yarn 实现，异常分析和趋势预测模块在 Spark 集群上运行。

Web 应用程序使用模拟在线书店的多层电子商务网站，包括 Java Servlet、数据库服务器(MySQL)、应用服务器(Apache Tomcat)。所有的超文本标记语言(HTML)页面均由服务器动态生成，根据 TPC-W 标准(交易网络基准)[11]，包括主页、畅销页面、新书页面、搜索页面、购物车和订单状态页面等。TPC-W 客户端为 Emulated Browser(EB)，可在会话中访问网站。会话是一系列逻辑连接的请求(从 EB 角度看)。在来自同一个 EB 的 2 个连续请求之间，TPC-W 经历了一个思考阶段，表明用户接收请求网页和生成下一个请求之间的时间。TPC-W 包括浏览、购物和订购 3 种交易模式，本文仅使用购物交易模式进行试验。Web 应用环境如图 6 所示，其中工作负载基于 TPC-W 标准的配置。

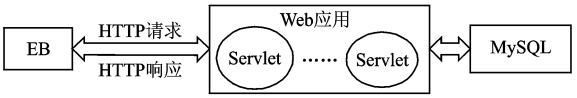


图 6　Web 应用环境

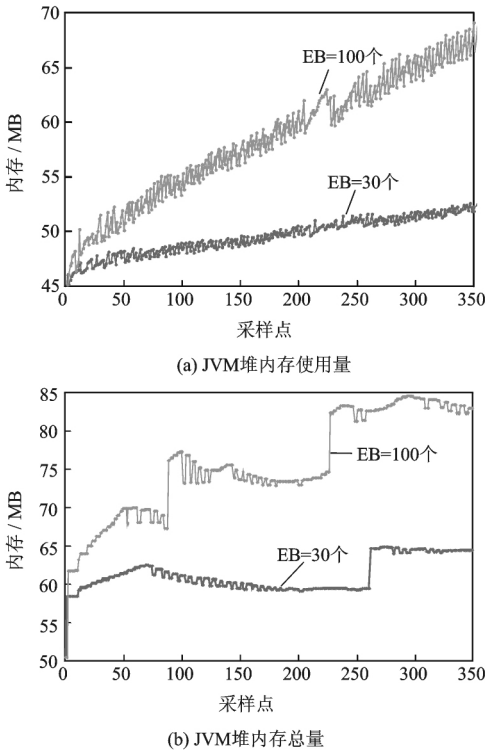
试验中，JVM 最大内存为 899 MB。为了在试验中观察到 JVM 内存和请求响应时间的变化，通过修改 TPC-W 中的 TPCW_search_request_servlet 注入内存泄漏，每次内存泄漏约 1 MB。注入内存泄漏方法是在 TPCW_search_request_servlet 的 doGet() 函数中添加代码占用内存。该代码产生 0～N 之间的随机数 n，每次请求后 $n-1$，经过 n 次请求后占用一段内存，同时产生新的随机数 n'，开始新循环。而页面调用次数与负载有关，本文选择 $N=4$，同时分别用 30 和 50 个 EB 进行测试，获取不同负载下的 JVM 内存总量、内存使用量及页面响应时间。

3.2 试验结果

本文将不注入内存泄漏的情况称正常情况，正常情况下 Web 服务平台各变量统计结果如表 1 所示。图 7 给出了 $N=4$ 且采样时间间隔为 5 s 时，EB=30 个和 EB=100 个的 JVM 堆内存使用量和 JVM 堆内存总量。图 8 给出了 $N=4$ 且采样时间间隔为 5 s 时监控到的页面访问响应时间。

表 1 正常情况下 Web 服务平台各变量统计结果

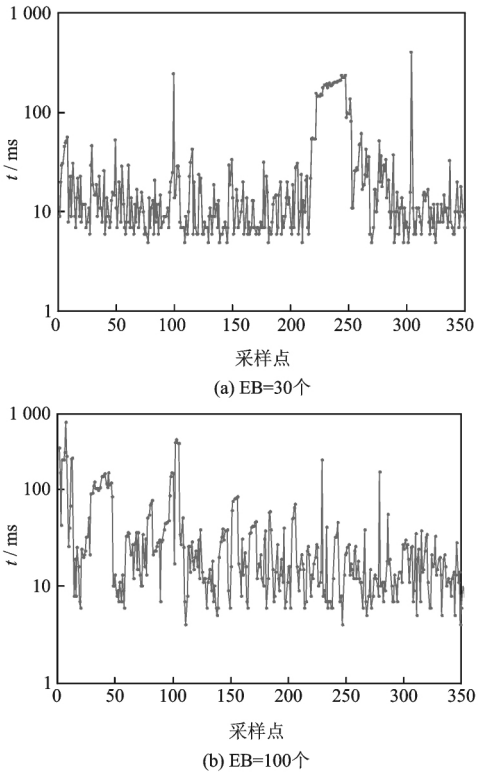
采集指标项	均值	方差
JVM 堆内存使用量/MB	48.93	1.37
JVM 堆内存总量/MB	61.20	1.91
页面响应时间/ms	29.06	48.56



(a) JVM堆内存使用量

(b) JVM堆内存总量

图 7 JVM 堆内存使用量和 JVM 堆内存总量



(a) EB=30个

(b) EB=100个

图 8 页面访问响应时间

图 9 给出了采样时间间隔为 5 s 时的采集点数据到达 Spark 的延时。基于统计的异常检测算法需要 nKD 个样本判断异常，因此经过 (interval+delay)nKD 的时间可以检测到异常。其中，interval 为采样间隔；delay 为传输时延。图 10 给出了 $N=4$、EB=100 个且采样时间间隔为 5 s 时检测到异常

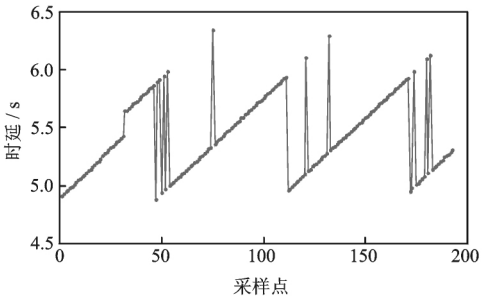


图 9 采集点数据到达 Spark 的时延

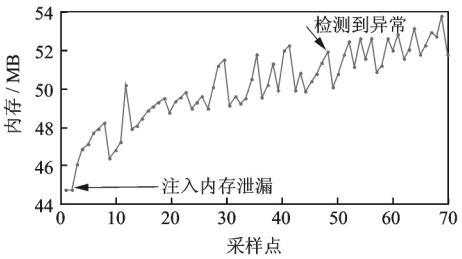
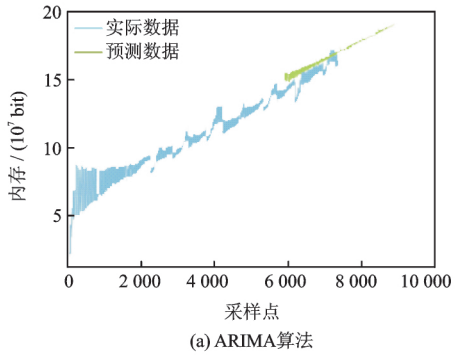


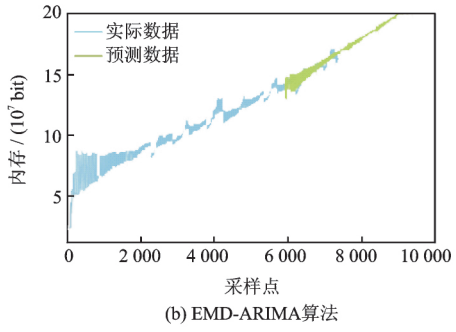
图 10 检测到异常点的时间

点的时间。

图 11 给出使用 ARIMA 和 EMD-ARIMA 2 种算法对 JVM 堆内存使用量的预测结果。本文采用平均绝对误差（MAE）、均方根误差（RMSE）、均方误差（MSE）及泰勒不等系数 4 种误差评估方法对这 2 种算法的预测效果进行评估。2 种算法预测结果评估误差对比如表 2 所示。可见，与 ARIMA 算法相比，EMD-ARIMA 算法预测精度明显提高。



(a) ARIMA算法



(b) EMD-ARIMA算法

图 11　JVM 堆内存使用量预测结果

表 2　2 种算法预测结果评估误差对比

算法	MAE/MB	RMSE/MB	MSE	泰勒不等系数
ARIMA	7.17	8.43	5.10	0.03
EMD-ARIMA	3.46	4.57	2.42	0.02

4　结束语

为保障云服务系统的可靠性，本文设计了基于数据分析的云监控系统，包括采集器、收集器、Kafka 消息队列和数据分析等模块，其中数据分析模块采用大数据分析方法实现过载异常的检测和资源趋势预测。通过在 OpenStack 云平台上搭建基于 Tomcat 的 Web 应用，对 Web 应用的 JVM 堆内存和页面请求响应时间进行监测，并应用加速内存泄漏方法在试验过程中不断增大 JVM 堆内存量，采用基于统计的异常检测方法发现 JVM 堆内存的过载异常，构建 EMD-ARIMA 算法实现对 JVM 堆内存的趋势预测。仿真试验表明，与 ARIMA 算法相比，EMD-ARIMA 算法预测精度更高。

参考文献（References）：

[1] CALHEIROS R N, RANJAN R, DE ROSE C A F, et al. CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services[EB/OL]. [2018-08-15]. https://arxiv.org/ftp/arxiv/papers/0903/0903.2525.pdf.

[2] 陈继延. 计算机系统失效与信息安全[J]. 湖南警察学院学报，2005，17(3)：101-104.
CHEN Jiyan. Losing efficiency of the computer system and its security[J]. Journal of Hunan Public Security College，2005，17(3)：101-104. (in Chinese)

[3] GROTTKE M, LI L, VAIDYANATHAN K, et al. Analysis of software aging in a Web server[J]. IEEE Transactions on reliability，2006，55(3)：411-420.

[4] GROTTKE M, NIKORA A P, TRIVEDI K S. An empirical investigation of fault types in space mission system software[C]//2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Chicago：IEEE，2010：447-456.

[5] 张萍，王桂增，周东华. 动态系统的故障诊断方法[J]. 控制理论与应用，2000，17(2)：153-158.
ZHANG Ping, WANG Guizeng, ZHOU Donghua. Fault diagnosis methods for dynamic systems[J]. Control Theory and Applications，2000，17(2)：153-158. (in Chinese)

[6] KLEPPMANN M. Designing data-intensive applications[M]. Sebastopol：O'Reilly Media，2017.

[7] Apache Kakfa[EB/OL]. [2018-08-15]. http://kafka.apa-che.org/.

[8] Spark[EB/OL]. [2018-08-15]. https://spark.apache.org/.

[9] ZHAO J, JIN Y, TRIVEDI K S, et al. Injecting memory leaks to accelerate software failures[C]// IEEE International Symposium on Software Reliability Engineering. Hiroshima：IEEE，2011：260-269.

[10] OpenStack Ocata[EB/OL]. [2018-08-15]. https://docs.Openstack.org/ocata/.

[11] TPC-W benchmark Java version[EB/OL]. [2018-08-15]. http://www.ece.wisc.edu/~pharm/tpc-w.html.

作者简介：
李志娟，女（1988—），博士研究生，研究方向为车载自组网的可靠性和软件无线电。
吴卓霏，男（1991—），博士研究生，研究方向为车载自组网的可靠性和机器学习。
王思家，男（1991—），硕士研究生，研究方向为软件可靠性。
赵　靖，女（1971—），教授，研究方向为软件可靠性和机器学习。

（本文编辑：马　岚）