计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191200186

云计算环境下关联节点的异常判断

雷 阳 姜 瑛
云南省计算机技术应用重点实验室 昆明 650500
昆明理工大学信息工程与自动化学院 昆明 650500
(201015700@qq.com)

摘 要 当前,越来越多的用户选择将服务部署到云计算环境中。然而,云计算服务的多样性以及部署环境的动态性,会导致云计算节点出现异常。传统的节点异常检测方法只针对异常的单一节点,忽略了异常节点对关联节点的影响,从而造成异常传播和关联节点失效等问题。文中提出了一种云计算环境下关联节点的异常判断方法。首先,将 Agent 部署在各节点上,并通过 Agent 以特定时间间隔采集节点运行数据,根据节点之间的关联关系建立节点关系图;其次,使用运行数据训练异常检测模型,计算运行数据的权值和综合评分,通过基于滑动时间窗口的方法判断单一节点是否出现异常;最后,在单一节点出现异常的情况下,使用基于标准互信息的方法找出受异常节点影响的其他关联节点。在搭建的云计算平台上,通过模拟各类异常情况,并观察注入异常下节点的状态,验证了文中单一节点异常判断方法和关联节点判断方法的有效性。实验结果表明,该方法在判断单一节点异常的正确率和特异度时都优于其他方法,且在多节点结构下可以准确找到关联的异常节点,具有较高的准确率和稳定性。

关键词:云计算环境;单一节点;关联节点;异常判断;标准互信息
中图法分类号 TP311

Anomaly Judgment of Directly Associated Nodes Under Cloud Computing Environment

LEI Yang and JIANG Ying
Yunnan Key Lab of Computer Technology Application,Kunming 650500,China
Faculty of Information Engineering and Automation,Kunming University of Science and Technology,Kunming 650500,China

Abstract Currently,more and more users deploy their services on cloud computing environment. Due to the services diversity and dynamic deployment,the anomalies will be occurred on nodes under cloud computing environment. The impact of anomaly nodes on associated nodes are usually neglected in the traditional node anomaly detection methods,which will result in anomaly propagation and nodes failure. In this paper,a method of anomaly judgment for directly associated nodes under cloud computing environment is proposed. At first,the Agent is deployed on each node and the running data of nodes are collected through the Agent at specific time intervals. The node relationship graph is established based on the relationship between the nodes. Secondly,the anomaly detection model is trained by the running data. Then the weights and comprehensive scores of the running data is calculated. The anomaly of the single node is judged by the sliding time window-based method. Finally,other nodes affected by the anomaly nodes are found through the normalized mutual information in the case of a single node anomaly. In this paper,the relevant experiments are carried out on the cloud computing platform. In order to simulate all kinds of anomaly situations,the anomaly conditions are injected during the experiment and the state of nodes under the injection anomaly is observed. The validity of single node and directly associated node of anomaly judgment method was verified by experiments. The experimental results showed that the accuracy and specificity of the method in this paper are better than other methods about single node anomaly judgment. Under the multi-node structure,the method of this paper could find the directly associated anomaly node with the higher accuracy and stability.

Keywords Cloud computing environment,Single node,Directly associated nodes,Anomaly judgment,Normalized mutual information

1 引言

近年来,云计算技术飞速发展,已被广泛应用于诸多领域。云计算环境聚集了大量的物理资源和虚拟资源,并提供了 IaaS,PaaS 和 SaaS 等多个层次的服务[1]。云计算环境中的虚拟机和物理机资源被称为云计算节点。由于云计算服务

到稿日期:2019-12-31 返修日期:2020-06-13 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61462049,61063006,60703116);云南省应用基础研究计划重点项目基金(2017FA033)
This work was supported by the National Natural Science Foundation of China(61462049,61063006,60703116) and Key Project of Yunnan Applied Basic Research(2017FA033).
通信作者:姜瑛(jy_910@163.com)

的多样性以及部署环境的动态性，云计算节点时常会出现异常，且异常通常会转化为严重的故障，造成服务失效、资源浪费[2]。此外，在运行过程中出现异常的节点可能会导致与其相关联的其他节点发生异常，并进一步引起大规模的服务失效。通过及时的异常判断，系统管理员可采取相应的措施来避免或降低异常的影响，以满足云计算服务的需求。因此，云计算环境下关联节点的异常判断是一个亟待解决的问题。

2 相关工作

云计算环境下各类服务以不同方式部署在节点上。为保证云计算服务的正常运行，必须及时获取节点的运行状态。有效的异常检测方法不仅可以降低节点异常的影响，还可以提高服务质量。

云计算环境下节点异常检测方法通常包括单一节点异常检测方法和基于节点比较异常检测方法。

单一节点异常判断方法通常从节点采集运行数据（如CPU、内存、I/O和网络）来表征节点状态，并结合异常检测算法来确定节点状态是否异常。Fu等[3]引入性能指标（节点中每个组件的统计信息包括CPU使用率、进程创建、内存等）的互信息来量化性能指标之间的相关度和冗余度，采用增量搜索算法实现性能指标的选择，最后采用主成分分析（Principal Component Analysis，PCA）实现特征提取，通过决策树分类器的检测机制识别异常。Lin等[4]针对云环境提出了一种虚拟机异常检测框架，首先采用特征提取算法对性能指标数据降维；然后采用聚类算法将降维后的性能指标数据聚类成多个簇；最后通过判断新的性能指标数据是否偏离最近的簇来进行异常检测。Guan等[5]提出了一种自适应异常识别机制用于检测云平台的异常行为，该机制首先采用PCA找出与每种故障类型最相关的主元，然后基于云平台性能数据，采用自适应卡尔曼滤波进行异常检测。

现有的节点比较异常检测研究方法假设节点具有对等且可比较的环境，其中预期执行可比较活动的节点表现出了类似的行为，如果节点表示出异常行为，则认为节点异常。Lan等[6]提出了一种异常自动识别机制，通过节点分组模块动态地对系统资源进行分组，使得同一组中的节点显示相同的行为。首先用PCA降低数据维度，然后使用基于单元的算法识别异常节点。Kang等[7]提出的故障诊断方法Peerwatch，引入典型关联分析（Canonical Correlation Analysis）对部署在不同虚拟机上的相同应用的多个实例间的关联性进行建模，以检查每个应用实例的状态；与其他众多实例的关联性发生改变的实例，被检测为故障实例。Li等[8]采用分组策略，将一组系统节点分成若干个子组，同时保持同行可比环境，采用非参数聚类和两阶段多数表决的分散方法检测大规模系统中的性能异常。Pertet等[9]采用同行比较方法识别主动复制系统中的异常节点。Kasick等[10]通过比较节点之间的系统度量，在分布式环境中进行异常检测。

上述研究虽然已经解决了部分问题，但是仍然存在不足之处。单一节点异常检测方法[3-5]尽管具有广泛的可用性，但只针对单一节点的异常进行研究，缺少对与之关联的其他节点的判断。大部分基于节点比较的异常检测方法[6-10]基于两个假设：首先，节点处于对等且可比较的环境，预期执行类似活动的节点表现出类似的行为；其次，正常节点占多数，并且它们具有相似的行为。但在云计算环境下，由于节点资源分配不均等情况，即使在相同环境下，节点也有可能表现出不同的行为；并且这些方法没有考虑到节点异常传播等影响导致的节点失效等问题。

为了解决云计算环境下节点异常判断的问题，本文提出了一种针对关联节点异常判断的方法。首先，查找异常节点；其次，对异常节点的关联节点进行分析；最后，找出受异常影响的节点。

3 云计算环境下关联节点的异常判断方法

为了判断节点异常，首先需要对各节点的运行数据进行监控。本文采用Agent技术[11]，利用Agent的自主性、反应性、协作性等特性自动分析节点运行数据并发现异常节点。本文根据Agent执行功能的不同，将其分为SAgent和MAgent。其中，SAgent部署到云计算环境的每个节点中，用于收集和分析数据；MAgent根据SAgent收集的信息分析节点之间的状态，并判断异常。云计算环境下关联节点的异常判断结构如图1所示。

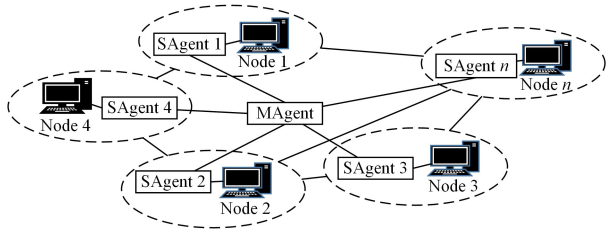

图1 云计算环境下关联节点的异常判断结构

Fig. 1 Anomaly judgment structure of associated nodes under cloud computing environment

为了判断云计算环境下关联节点的异常，首先由SAgent对每个节点的运行数据进行监控；然后判断单一节点的状态；若节点出现异常，再由MAgent判断与当前节点关联的其他节点是否存在异常。云计算环境下关联节点的异常判断过程如图2所示。

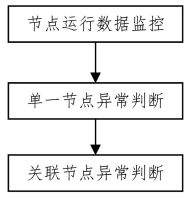

图2 云计算环境下关联节点的异常判断过程

Fig. 2 Anomaly judgment process of associated nodes under cloud computing environment

3.1 节点运行数据监控

在监控云计算环境下的节点运行数据时，首先应采集节点的运行数据。由于节点之间的关系是不断变化的，还需要对节点之间的关系进行判断。

3.1.1 节点运行数据采集

由于节点的CPU、内存、网络等资源的消耗不仅能够反映出节点当前的运行状态[12]，还能够帮助判断当前节点是否出现异常，并且节点异常会引起多个运行数据发生变化，因此本文通过采集表示节点状态的运行数据，并结合异常判断算法来判断节点状态。本文部署的SAgent使用SIGAR[13]的

开源 API 和 sysstat 软件包[14]等工具在连续的固定时间间隔内采集节点的运行数据。

由于采集方法不同,运行数据通常具有不同的格式和语义[15],因此难以以统一的方式处理它们。为了解决这些问题,本文对采集的运行数据进行了预处理,包括将可变间隔时间序列转换为固定间隔时间序列、填充缺失的样本、格式化数值等。

3.1.2　节点关系图建立

为了判断云计算环境中关联节点的状态,首先需要建立节点的拓扑结构。云计算环境中节点之间的关联关系包括直接关联和间接关联,本文采用图来描述云计算环境中节点之间的关联关系。

设 NAG(Nodes Associated Graph)为节点关系图,该图可用一个二元组 $NAG=(V,E)$ 表示。其中,$V=\{v_1,v_2,\cdots,v_n\}$是云计算环境中所有节点的集合,$E=\{e_{11},e_{12},\cdots,e_{nl}\}$ 且 $E\in V\times V$ 是节点边的集合,其中 $e_{ij}=(v_i,v_j)$。

通过建立 NAG 可以获取节点间的关系,为关联节点的异常判断提供支持。在建立节点关系图时,首先需要获取云环境下所有节点的信息,本文通过每个节点中的 SAgent 获取所有节点信息(IP 地址、主机号等);其次需要获取每个节点的关联节点,本文通过 SAgent 判断其他节点是否可达来确定节点的关联节点。节点关系图建立算法的具体步骤如算法 1 所示。

算法 1　节点关系图建立算法

输入:通过 SAgent 获取的节点信息 nList(包括 IP、主机名等)

输出:NAG

1. IF NAG 为空
2.　　新建 NAG,并依据 nList 初始化 NAG 中的集合 V
3. ELSE
4.　　更新集合 V
5. END IF
6. FOR all i in V DO
7. FOR all j in V DO
8.　　IF 节点 i 到节点 j 可达
9.　　更新集合 E,令 e_{ij}=1/*e_{ij}的取值为 1 或者 0,1 代表节点 i 到节点 j 可达,0 代表节点 i 到节点 j 不可达*/
10.　　ELSE
11.　　更新集合 E,令 e_{ij}=0
12.　　END IF
13. END FOR
14. END FOR
15. RETURN NAG

3.2　单一节点异常判断

为了判断关联节点的状态,必须先判断单一节点是否存在异常。根据第 3.1.1 节采集的节点运行数据,我们先使用正常数据训练异常检测模型,并计算滑动时间窗口内运行数据的权值,再使用式(1)计算数据的综合评分用于识别异常数据[16]。

$$Comprehensive\ Score=\sum_{i=1}^{n}(mindist(i)\times count(i)) \quad (1)$$

其中,$minist(i)$表示第 i 个运行数据与异常检测模型的最短距离,$count(i)$表示第 i 个运行数据的方差贡献率。*Comprehensive Score* 表示数据的综合评分,代表数据的偏离程度。经过多次实验,本文选择 1.5 作为 *Comprehensive Score* 的阈值。

以上方法基于时间点来判断节点是否出现异常。然而,由于节点本身具有高动态性,这将造成节点的数据出现较大的波动,一些正常的数据波动可能会被误认为异常,并且通过单个时间点的节点数据很难准确检测出节点的运行状态。为了解决以上问题,本文采用基于时间窗口的异常判断方法[17]来判断节点的运行状态。基于时间窗口的异常判断方法根据时间窗口内正常数据和异常数据的比例来判断节点最终的运行状态。对于时间窗口的设计,本文采用滑动时间窗口的方法。滑动时间窗口基于先进先出的方法,并且能够保留节点的最新数据。

为了对单一节点的状态进行判断,本文将节点的运行数据放入滑动时间窗口内,先使用基于时间点的方法判断节点状态;如果出现异常,再计算滑动时间窗口内异常数据和正常数据的比值,若该比值大于数据异常程度的阈值 μ,则节点状态为异常,否则为正常。单一节点异常判断算法的步骤如算法 2 所示。

算法 2　单一节点异常判断算法

输入:节点数据 DataList,滑动时间窗口大小 winSize

输出:TRUE/FALSE /*返回值代表节点状态,TRUE 为节点异常,FALSE 为节点正常*/

1. 滑动时间窗口内数据条数 winNum=0
2. FOR all i in DataList DO
3. winNum++
4. 将第 i 条数据以先进先出的方式放入滑动时间窗口内
5. IF 第 i 条数据的 Comprehensive Score>1.5 //判断数据是否异常
6.　　异常数据个数 anomalyNum++
7.　　ELSE
8.　　正常数据个数 Num++
9.　　END IF
10. IF winNum== winSize /*滑动时间窗口内数据已满*/
11.　　IF anomalyNum / Num>μ /*异常数据和正常数据所占的比例大于 μ,即节点异常*/
12.　　ETURN FALSE
13.　　END IF
14. END IF
15. END FOR
16. RETURN TRUE

本文经过多次实验,将 μ 的取值设为 0.3,超过 μ 则认为数据异常。

3.3　关联节点异常判断

由于云计算环境的多样性,当单一节点出现异常时,也可能造成其他关联节点发生异常。本节通过判断关联节点的运行状态,寻找由于单一节点异常而引发的其他节点异常。

在寻找异常影响的节点时,首先需要了解节点之间数据的相关性,然后通过这些相关性找出受到异常影响的节点。互信息[3]表示两个变量共享的信息量。通过互信息这一特性,可以度量节点之间数据的相关性。

由于互信息没有上界,为了更容易比较,需要一个取值范围为 0 到 1 之间的标准化的互信息。Strehl 等[18]提出了互信息的标准化,称为标准互信息(Normalized Mutual Information,NMI),以弥补互信息的缺点。NMI 的定义为:

$$NMI(X_i;Y_i)=\frac{I(X_i;Y_i)}{\sqrt{H(X_i)H(Y_i)}} \quad (2)$$

NMI 的取值在 0 到 1 之间，0 代表两个数据完全无关，1 代表完全相关。*NMI* 越趋近于 1，表示两个数据越相关。一般地，NMI 不考虑变量之间的具体形式，只要两个变量相关性越大，它们的 *NMI* 值就越高。因此，NMI 提供了两个变量之间相关性的良好度量。

为了度量两个变量的相关性，还需要一个合适的阈值 t_{NMI} 来区分变量强相关性和弱相关性。本文使用文献[18]中的方法，利用决定系数 r^2 估计强相关的 t_{NMI} 阈值，如式(3)所示：

$$t_{NMI}=\min_{r^2_{X_iY_i}>t_{r^2}} NMI(X_i;Y_i) \tag{3}$$

本文通过多次实验计算 r^2 和 *NMI* 的值，从而得到了给定 t_{r^2} 的 t_{NMI} 估计值为 0.6。

在单一节点发生异常的情况下，为了对关联节点的状态进行判断，首先需要了解节点间的关系，本文通过第 3.1.2 节中建立的节点关系图 NAG 来获取异常节点的关联节点；然后通过本节所述方法计算异常节点与其关联节点的 *NMI*，分析节点数据之间的相关性；最后判断关联节点是否受异常节点影响。关联节点异常判断算法的具体步骤如算法 3 所示。

算法 3 关联节点异常判断算法

输入：NAG=(V,E)，异常节点 abNode
输出：关联的异常节点 abNodeList
1. 遍历 NAG，获取节点 abNode 的关联节点集合 reNodeList
2. FOR all i in reNodeList DO:
3. 获取滑动时间窗口内异常节点 abNode 和节点 i 的运行数据 abDataList，DataList /* 数据格式为 m×n 大小的矩阵，m 为数据条数，n 为数据的维度 */
4. 对 abDataList 和 DataList 进行归一化
5. FOR all j in n DO:
6. FOR all k in n DO:
7. 计算 abDataList 中第 j 维数据和 DataList 中第 k 维数据之间的标准互信息 NMI
8. IF NMI>t_{NMI} /* NMI 超过阈值 t_{NMI} 则说明节点受到异常节点影响 */
9. 将节点 i 添加到 abNodeList 中
10. END IF
11. END FOR
12. END FOR
13. RETURN abNodeList

4 实验及分析

本文搭建了基于 OpenStack 的弹性云计算平台，共部署了 10 个虚拟机节点，拓扑结构如图 3 所示，并在节点上搭建了 Hadoop 并发计算框架[19]。

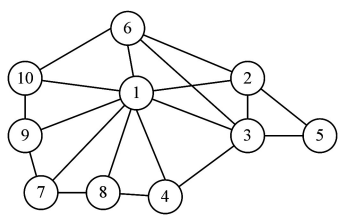

图 3 节点拓扑结构

Fig. 3 Node topology

图 3 中各节点的硬件配置如表 1 所列。

表 1 节点的硬件配置

Table 1 Hardware configuration of nodes

节点	配置		
	CPU	Meory/GB	Ethernet/Mbps
1	4 core	8	1000
2—9	2 core	2	1000
10	8 core	16	1000

为了验证第 3 节中的方法，本文开发了一个云计算环境下关联节点异常判断原型软件，并将其部署在云计算节点中进行数据采集及异常判断。

在数据的采集过程中，如果采集间隔时间太长，则很容易遗漏节点的异常数据；如果收集间隔太短，将对节点的性能产生很大的影响。本文经过实验，选择间隔 5s 记录一次节点的运行数据，这样不会遗漏重要数据，也不会过于占用节点的资源；并采用先进先出的方法将采集的运行数据放入滑动时间窗口内。

为了采集尽可能多的代表节点状态的数据，本文分别从 CPU、内存、I/O 和网络中收集运行数据，如表 2 所列。

在实验过程中，不能保证节点会出现各种异常。针对这一问题，普遍的解决方案是在节点中注入异常，并观察注入异常下节点的行为[6]。

本文采集了 10 个节点在正常运行状态下的 7200 条数据，并在随后的实验环境中注入异常，采集注入异常后的数据 1440 条，并针对不同情况设计了对比实验。其中，实验 1 判断单一节点状态，实验 2 判断关联节点状态。

为了验证本文方法的有效性，采用灵敏度(Sensitivity)、特异度(Specificity)、正确率(Correct Detection Rate)对其性能进行度量[20]。灵敏度也称为召回率(Recall)，指能正确识别异常的概率；特异度指检测没有发生异常的概率；正确率指正确检测出的异常和正常占所有情形的比例。

表 2 采集的节点运行数据

Table 2 Collected running data

监测内容	描述
cpu_usr	用户进程消耗的 CPU 时间百分比
cpu_sys	内核进程消耗的 CPU 时间百分比
cpu_wai	IO 等待消耗的 CPU 时间百分比
mem	内存使用率
io_r	从块设备读入的数据总量(读磁盘) /(KB/s)
io_w	写入到块设备的数据总量(写磁盘) /(KB/s)
net_r	网卡接收到数据量/(KB/s)
net_s	网卡发送到数据量/(KB/s)
sys_int	每秒产生的中断次数
sys_csw	每秒产生的上下文切换次数

4.1 实验 1

本文使用文献[6]中的异常检测算法与本文方法进行了对比。两种方法对单一节点异常判断的实验结果如图 4 所示。

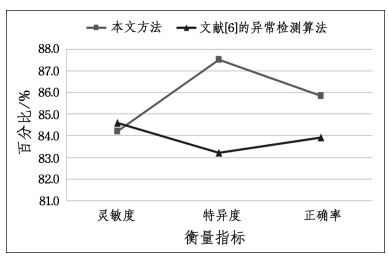

图 4 单一节点的异常判断结果

Fig. 4 Anomaly judgment result of single node

实验结果表明，文献[6]的异常检测算法的灵敏度略优于本文方法，但是本文方法在特异度和正确率方面的性能都高于文献[6]的异常检测算法。经分析，本文方法的灵敏度低于文献[6]中的方法是由于节点受到噪声干扰，造成异常数据被漏报；而文献[6]中的方法通过使用 PCA 可过滤噪声的影响。

4.2　实验 2

本文方法不仅能够判断单一节点状态，还能够对异常节点的关联节点状态进行判断。为了验证本文方法对关联节点状态判断的有效性，本文使用标准互信息的方法来判断关联节点的关系，并通过节点之间互信息值的大小来判断节点之间数据的关联程度。

为了验证标准互信息方法的有效性，本文分别计算关联节点和未关联节点的标准互信息并进行对比。在实验中，使用阈值 $t_{NMI}=0.6$ 区分数据之间的强相关性和弱相关性，并在节点 1 中注入异常，计算节点 1 与关联节点（节点 2）的标准互信息（见表 3），以及节点 1 与未关联节点（节点 5）的标准互信息（见表 4）。

表 3　节点 1 与节点 2 的标准互信息矩阵

Table 3　*NMI* matrix for node 1 and node 2

关联节点	异常节点									
	cpu_usr	*cpu_sys*	*cpu_wai*	*mem*	*io_r*	*io_w*	*net_r*	*net_s*	*sys_int*	*sys_csw*
cpu_usr	**0.6702**	0.4663	0.2289	**0.6524**	0.0736	0.5996	**0.6621**	**0.7490**	0.5640	0.4554
cpu_sys	0.5957	0.3611	0.1673	0.5436	0.0614	0.5057	0.4910	0.6031	0.4875	0.4066
cpu_wai	0.1552	0.1148	0.0123	0.1134	0.0065	0.0714	0.1400	0.1274	0.1184	0.1486
mem	0.5929	0.3858	0.2449	0.5667	0.0614	0.4970	0.5084	0.5819	0.5168	0.4125
io_r	0	0	0	0	0	0	0	0	0	0
io_w	**0.6369**	0.3709	0.1707	0.5273	0.0599	0.5799	0.5661	**0.6974**	0.5120	0.3581
net_r	**0.6795**	0.4621	0.2115	**0.6592**	0.0775	**0.6201**	**0.6885**	**0.7710**	**0.6093**	0.4214
net_s	**0.7036**	0.4274	0.1901	**0.6698**	0.1021	**0.6252**	**0.7134**	**0.7720**	**0.6056**	0.4169
sys_int	0.4926	0.2973	0.0883	0.3784	0.0165	0.3657	0.3404	0.4161	0.3806	0.3405
sys_csw	0.3707	0.2780	0.0494	0.3700	0.0523	0.3248	0.3835	0.3812	0.2985	0.2927

表 4　节点 1 与节点 5 的标准互信息矩阵

Table 4　*NMI* matrix for node 1 and node 5

未关联节点	异常节点									
	cpu_usr	*cpu_sys*	*cpu_wai*	*mem*	*io_r*	*io_w*	*net_r*	*net_s*	*sys_int*	*sys_csw*
cpu_usr	0.0214	0.0497	0.0110	0.0504	0.0457	0.0441	0.0418	0.0397	0.0306	0.0176
cpu_sys	0	0	0	0	0	0	0	0	0	0
cpu_wai	0	0	0	0	0	0	0	0	0	0
mem	0.1644	0.0365	0.0117	0.0728	0.0383	0.1208	0.1398	0.2072	0.0755	0.0411
io_r	0	0	0	0	0	0	0	0	0	0
io_w	0.1894	0.0671	0.0822	0.0957	0.1601	0.2465	0.1566	0.2724	0.0792	0.0403
net_r	0.1801	0.0678	0.0999	0.0709	0.0890	0.1728	0.1118	0.1546	0.0615	0.0296
net_s	0.1944	0.0797	0.0900	0.0636	0.1411	0.1730	0.1316	0.2143	0.1292	0.0201
sys_int	0.1746	0.0885	0.0767	0.0872	0.0715	0.2065	0.1311	0.2448	0.1065	0.0813
sys_csw	0.2107	0.1661	0.0843	0.1237	0.0571	0.1805	0.1328	0.2618	0.1463	0.1360

表 3 中加粗部分代表标准互信息值大于阈值 t_{NMI}。表 4 中所有数据小于阈值 t_{NMI}，说明异常节点与其关联节点有强相关性，即关联节点受异常节点影响，证明本文用标准互信息判断关联节点的异常状态是有效的。为了进一步验证本文方法对关联节点状态判断的正确率，分别在 6 个节点（节点 1—节点 6）和 10 个节点（节点 1—节点 10）两种结构下进行关联节点异常状态的判断。两种情况下的实验结果如图 5 所示。

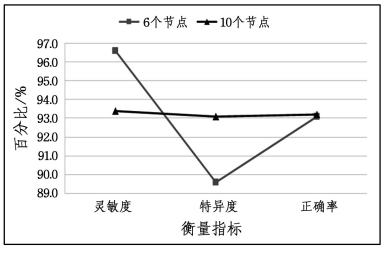

图 5　关联节点的异常判断结果

Fig. 5　Anomaly judgment result of associated nodes

从图 5 可以看出，10 个节点的结构灵敏度和特异度的值都趋近于 93%，正确率略高于 6 个节点的结构。6 个节点的结构灵敏度高于 10 个节点的结构，但特异度低于 10 个节点的结构，原因是在 6 个节点的结构下，节点的资源利用率较高，节点波动较大，对异常判断造成了影响。在多节点结构下，本文方法可以找到关联节点异常，并且具有很高的准确率，体现了更高的稳定性。

相关实验证明，本文方法对单一节点异常的判断结果优于文献[6]中的方法；此外，本文方法能够找出受异常节点影响的其他关联节点，判断出由于异常传播等影响而导致关联节点出现的异常。

结束语　本文提出了一种云计算环境下关联节点的异常判断方法。通过 SAgent 和 MAgent，首先采集节点的运行数据，并建立节点关系图；其次，使用基于时间窗口的方法判断单一节点状态；最后，在单一节点出现异常的情况下，基于标准互信息的方法对关联节点状态进行判断，并找出受到异常影响的所有节点。实验结果表明，该方法既能对单一节点进行异常判断，又能及时找出受异常节点影响的关联节点。

本文目前未能很好地去除噪声数据的影响，只针对直接关联的节点进行了异常判断，下一步将继续研究数据去噪及间接关联节点的异常判断问题。

参 考 文 献

[1]　LEE J. A View of Cloud Computing [J]. Communications of the ACM, 2013, 53(4): 50-58.

[2] WANG T, ZHANG W B, XUN J W, et al. A Survey of Fault Detection for Distributed Software Systems with Statistical Monitoring in Cloud Computing [J]. Chinese Journal of Computers, 2017, 40(2): 397-413.

[3] FU S. Performance Metric Selection for AutonOMic Anomaly Detection on Cloud Computing Systems[C]// 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011. Kathmandu: IEEE, 2011: 1-5.

[4] LIN M, CHEN S. An Efficient Anomaly Detection Framework for Cloud Computing Environment [J]. Journal of Computers, 2015, 10(3): 155-165.

[5] GUAN Q, FU S. Adaptive Anomaly Identification by Exploring Metric Subspace in Cloud Computing Infrastructures[C]// 2013 IEEE 32nd International Symposium on Reliable Distributed Systems. Braga: IEEE, 2013: 205-214.

[6] LAN Z, ZHENG Z, LI Y. Toward Automated Anomaly Identification in Large-scale Systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(2): 174-187.

[7] KANG H, CHEN H, JIANG G. PeerWatch: A Fault Detection and Diagnosis Tool for Virtualized Consolidation Systems[C]// Proceedings of the 7th International Conference on Autonomic Computing. ACM, 2010: 119-128.

[8] LI Y, LAN Z. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(7): 1902-1914.

[9] PERTET S, GANDHI R, NARASIMHAN P. Fingerpointing Correlated Failures in Replicated Systems [C]// Proceedings of the Second Workshop on Tackling Computer Systems Problems with Machine Learning. 2007: 220-230.

[10] KASICK M P, TAN J, GANDHI R, et al. Black-Box Problem Diagnosis in Parallel File Systems[C] // 8th USENIX Conference on File and Storage Technologies. San Jose: USENIX Association, 2010: 23-26.

[11] QIANG C. Research on Cloud Computing Resource Management Model Based on Multi-Agent System[C]// 2016 12th International Conference on Computational Intelligence and Security. Wuxi: IEEE, 2016: 378-381.

[12] ALHAMAZANI K, RANJAN R, MITRA K, et al. Clams: Cross-layer Multi-cloud Application Monitoring-as-a-service Framework[C]// 2014 IEEE International Conference on Services Computing. Anchorage: IEEE, 2014: 283-290.

[13] REDDY P V V, RAJAMANI L. Performance Comparison of Different Operating Systems in the Private Cloud with KVM hypervisor using SIGAR framework[C]// 2015 International Conference on Communication, Information and Computing Technology(ICCICT). Mumbai: IEEE, 2015: 1-6.

[14] WANG X, HUANG S, FU S, et al. Characterizing Workload of Web Applications on Virtualized Servers [C] // Architectural Support for Programming Languages and Operating Systems. 2014: 98-108.

[15] WANG D Q, WANG X X. Large Data Optimization Particle Swarm Clustering Algorithm Based on Cloud Storage [J]. Electronic Design Engineering, 2017, 25(2): 26-30.

[16] LEI Y, JIANG Y. Anomaly Judgment for Nodes Based on Agent under Cloud Environment[C]// 2019 IEEE International Conference on Computer Science and Educational Informatization (CSEI). Kunming: IEEE, 2019: 161-167.

[17] REN T. Research on Virtual Machine Anomaly Detection System Oriented IaaS[D]. Chongqing: Chongqing University, 2014.

[18] JIANG M, MUNAWAR M A, REIDEMEISTER T, et al. Automatic Fault Detection and Diagnosis in Complex Software Systems by Information-theoretic monitoring[C]// 2009 IEEE/IFIP International Conference on Dependable Systems & Networks. Lisbon: IEEE, 2009: 285-294.

[19] O'DRISCOLL A, DAUGELAITE J, SLEATOR R D. 'Big data', Hadoop and cloud computing in genomics[J]. Journal of biomedical informatics, 2013, 46(5): 774-781.

[20] LIU C C, JIANG Y. Fault Detection Method for Cloud Computing Using Improved Fuzzyk Nearest Neighbor[J]. Journal of Chinese Computer Systems, 2018, 39(10): 159-164.

LEI Yang, born in 1992, postgraduate, is a member of China Computer Federation. His main research interests include cloud computing and big data analysis.

JIANG Ying, born in 1974, Ph. D, professor, Ph. D supervisor, is a senior member of China Computer Federation. Her main research interests include software quality assurance and testing, cloud computing, big data analysis and intelligent software engineering.