一种基于自适应监测的云计算系统故障检测方法

王 焘 顾泽宇 张文博 徐继伟 魏 峻 钟 华

(计算机科学国家重点实验室 北京 100190)

(中国科学院软件研究所 北京 100190)

摘 要 监测技术是保障云计算系统性能与可靠性的关键,管理员通过分析监测数据可以了解系统运行状态,从而采取措施以及早发现并解决问题.然而,云计算系统规模巨大,结构复杂,大量的监测数据需要搜集、传输、存储和分析,给系统带来巨大性能开销.那么,如何在提高故障检测的准确性和及时性的同时,减少监测开销成为亟待解决的问题.为了应对以上问题,该文提出一种基于自适应监测的云计算系统故障检测方法.首先,利用相关分析建立度量间的相关性,利用度量关联图选择关键度量进行监测;而后,利用主成分分析得到监测数据的主特征向量以刻画系统运行状态,进而基于余弦相似度评估系统异常程度;最后,建立可靠性模型以预测系统可能出现故障的时间,基于此动态调整监测周期.实验结果表明,该文所提出的方法能够适应云环境下负载的动态变化,准确评估系统异常程度,自动调整监测频率以提高系统在异常状况下故障检测的准确性与及时性,降低系统在正常运行过程中的监测开销.

关键词 故障检测;自适应监测;云计算;相关分析;主成分分析

中图法分类号 TP311 **DOI号** 10.11897/SP.J.1016.2018.01332

Adaptive Monitoring Based Fault Detection for Cloud Computing Systems

WANG Tao GU Ze-Yu ZHANG Wen-Bo XU Ji-Wei WEI Jun ZHONG Hua

(*State Key Laboratory of Computer Science, Beijing* 100190)

(*Institute of Software, Chinese Academy of Sciences, Beijing* 100190)

Abstract Monitoring is the key technology of guaranteeing the performance and reliability of distributed systems. By analyzing monitoring data, administrators can understand the systems' status to detect, diagnose and solve problems. However, the procedure of collecting, transmitting, storing and analyzing a large amount of monitoring data from large-scale cloud computing systems introduces enormous performance overhead. To address the above issue, this paper proposes an adaptive monitoring approach for fault detection. First, we conduct correlation analysis between different metrics to construct an undirected correlation graph, and monitor only selected important metrics from the graph, which can represent the other ones and reflect the running status of the whole system. Second, we use Principal Component Analysis (PCA) to characterize the running status based on the monitoring data from a sliding window to estimate the abnormality degree and predict the possibility of system faults by comparing the current and the historical collected monitoring data. Finally, we dynamically adjust the monitoring period

收稿日期:2016-04-22;在线出版日期:2016-10-19.本课题得到国家自然科学基金(61402450)、北京市自然科学基金(4154088)、CCF-启明星辰"鸿雁"科研资助计划(CCF-VenustechRP2016007)、国家科技支撑计划(2015BAH55F02)、国家"八六三"高技术研究发展计划项目(2013AA041301)资助.**王 焘**,男,1982年生,博士,副研究员,中国计算机学会(CCF)会员,主要研究方向为云计算系统的故障诊断、软件可靠性和自主计算.E-mail:wangtao@iscas.ac.cn.**顾泽宇**,男,1991年生,硕士,主要研究方向为分布式监测.**张文博**,男,1976年生,博士,研究员,博士生导师,主要研究领域为分布式计算、云计算和中间件.**徐继伟**,男,1985年生,博士,主要研究方向为软件工程和网络分布式计算.**魏 峻**,男,1970年生,博士,研究员,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为服务计算、中间件和软件工程.**钟 华**,男,1971年生,博士,研究员,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为软件工程和分布式计算.

based on the estimated abnormality degree and a reliability model. To evaluate our proposal, we have applied the approach in a TPC-W benchmark deployed in our cloud computing platform. The experimental results demonstrate that the approach can adapt to the dynamic workload fluctuation, accurately estimate the abnormality degree, and automatically adjust the monitoring frequency. Thus, the approach can effectively improve the accuracy and timeliness of fault detection in the abnormal status, and efficiently lower the monitoring overhead in the normal status.

Keywords　fault detection; adaptive monitoring; cloud computing; correlation analysis; principle component analysis

1　引　言

近年来，云计算技术飞速发展，并已经广泛应用于诸多领域，成为当前信息技术产业发展和应用创新的热点. 国内外大型 IT 企业纷纷推出云计算平台（如 Amazon EC2、阿里云等），同时开源云计算平台（如 Eucalyptus、OpenStack 等）的出现也促进了云计算技术研究与应用的发展. 目前，电子商务、互联网金融、社交网络等在线服务已经成为人们日常工作和生活中不可或缺的一部分，其中众多应用部署在云计算平台，依托于云服务（如 Saleforce CRM、Netflix 等）. 本文将这些部署在云计算平台上的分布式软件系统称为云计算系统.

应用的多样性以及部署环境的动态性使得云计算系统时常会出现故障，从而严重影响人们正常的工作、生活，甚至在商业方面造成巨大的经济损失[1]. 例如 2013 年 8 月亚马逊网站宕机约 45 min，在宕机期间消费者无法通过网站以及移动客户端进行购物，从而造成经济损失约达 530 万美元①. 在云计算系统运行过程中，高效监测是及时检测系统故障并准确定位问题原因的前提条件.

云计算系统规模巨大，结构复杂，监测系统需要从众多节点上搜集多个层次（如网络层、硬件层、虚拟机层、操作系统层、中间件层和应用软件层）各种资源的监测数据，以持续跟踪云计算系统的运行状态. 然而，搜集、传输、存储与分析大量监测数据将会带来巨大资源开销，从而影响系统性能以及商业监测系统（如亚马逊的 CloudWatch）只支持固定的较长的监测周期（如每分钟搜集一次监测数据）. 同时，从用户角度考虑，租用云监测服务需要支付的费用与监测的对象和频率成正比，而监测花费占到了总共运行成本的 18%②. 这样就造成了：一方面，管理员和用户希望减少监测对象和降低监测频率（即单位时间内的搜集监测数据的次数）以减少开销和降低成本. 另一方面，故障可能在连续监测的时间间隔内发生，监测对象过少以及监测频率过低会减少可用监测数据量，从而降低检出故障的准确性与及时性. 那么，如何设置监测对象与频率，成为监测云计算系统并保障其可靠性的关键.

当前，大规模数据中心监测主要关注于监测系统架构和传输协议设计，以降低监测对网络造成的压力. 管理员通常根据领域知识，针对不同应用场景，选择特定监测对象，人工设定数据搜集内容和频率调整规则，这种方法适用系统有限，且规则设定的优劣直接影响监测效果. 云计算环境下，应用呈现多样性，且应用对云平台是透明的，系统管理员难以设定面向特定领域的监测规则.

针对上述问题，本文提出一种基于自适应监测的云计算系统故障检测方法. 首先，利用相关分析分析度量间的相关性，从众多度量中选取反映系统运行状态的关键度量. 然后，根据滑动窗口中搜集的监测数据，利用主成分分析（Principal Component Analysis，PCA）刻画系统运行状态以评估系统异常程度，预测系统故障发生的可能性. 进而，基于可靠性模型与异常程度，动态调整监测周期. 实验结果表明，本文所提出的方法能够适应云环境下负载的动态变化，准确评估系统异常程度，自动调整监测频率以提高系统在异常状况下故障检测的准确性与及时性，同时降低系统在正常运行过程中的监测开销.

本文第 2 节介绍所提出方法的整体思路；第 3 节给出基于故障检测的自适应监测方法；第 4 节通

① Amazon.com goes down for 45 minutes, loses 5M in business. http://techcircle.vccircle.com/2013/08/20/amazon-web-site-goes-down-on-monday-company-loses-5m-in-business. 2013.8.20

② Amazon EC2 CloudWatch. http://aws.amazon.com/cloud-watch/. 2012

过实验验证本文所提出方法的有效性;第 5 节分析并比较相关工作;第 6 节对文章内容进行总结.

2 整体思路

监测对象和监测周期是影响系统监测开销、故障检测准确性与及时性的关键.因此,本文首先基于历史数据分析度量间的相关性从而选取反映系统运行状态的关键度量,而后根据系统出现故障的可能性自适应调整监测周期.

在监测对象选择方面,云计算系统各个层次存在的众多资源的使用情况需要监测,会造成巨大的监测开销.减少监测对象是提高监测效率的重要途径.系统中不同度量对运行状态的表现力是不同的,因此需要从众多度量中选取最能反映系统出现故障可能性的度量.

在监测周期调整方面,当系统出现故障的可能性较高时,缩短监测周期,搜集更多监测数据,以密切跟踪系统运行状态,从而提高故障检测的准确性和及时性.反之,当系统出现故障的可能性较低时,延长监测周期,从而降低监测开销.由于在整个系统运行过程中,故障出现的概率相对较少,动态调整监测周期可以减少大量监测开销,对于云环境下高效可扩展监测相当重要.

IBM提出了自主计算参考模型MAPE-K(Monitor、Analyzer、Planner、Executor 和 Knowledge),该模型由若干相互联系的自主元素组成,这些元素通过交互和协作实现计算系统自适应的自动管理[2].本文基于该模型实现面向云计算系统的自适应监测框架,根据云计算系统运行状态自适应调整监测对象和监测周期,以减少系统在正常运行状态的监测开销,并提高系统在异常状态的错误检测的准确性与及时性.如图 1 所示,框架中包括 MAPE-K 模型中的自主元素.

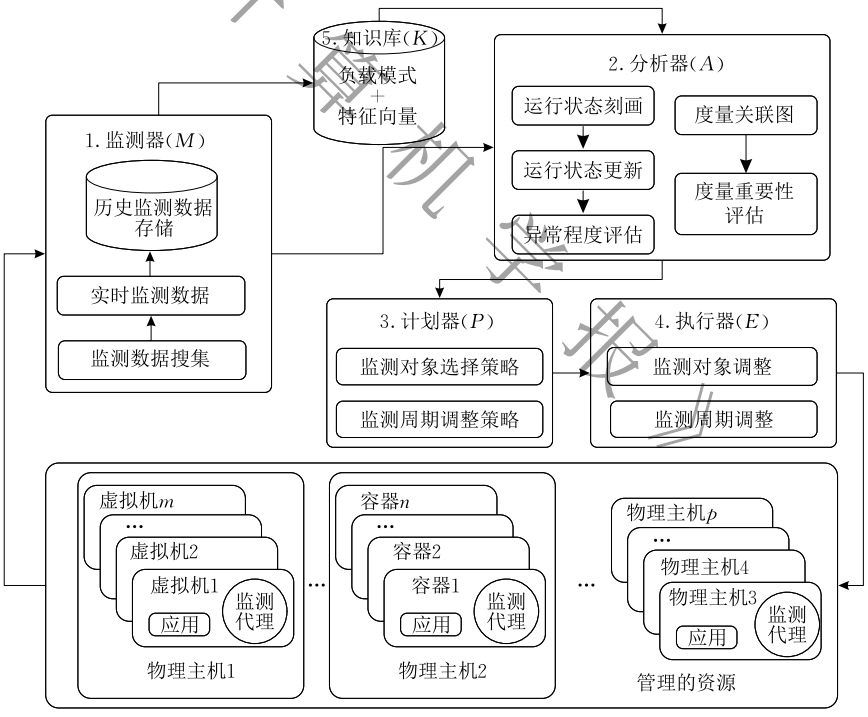


图 1 云计算系统自适应监测框架

(1)监测器(Monitor).监测代理部署在物理主机、虚拟机或者容器上,用以搜集各层次的监测数据并进行持久化存储.

(2)分析器(Analyzer).根据历史监测数据,建立度量间的相关性,形成度量关联图以评估度量的重要程度,并利用 PCA 计算监测数据的特征向量以量化评估系统异常程度.

(3)规划器(Planner).根据监测对象重要性选择下阶段需要监测的对象.利用泊松过程建立软件系统可靠性模型,基于异常程度预测系统故障发生的可能性以调整监测周期.

(4)执行器(Executor).利用监测代理执行监测对象与监测周期的动态调整.

(5)知识库(Knowledge).记录运行过程中学习到的负载模式以及对应的特征向量,以刻画系统正常运行状态.

3　自适应监测方法

3.1　关键度量选择

云计算系统中多种资源利用之间往往具有相关性[3]，需要监测的系统度量之间普遍存在稳定的线性关系[4]. 皮尔逊(Pearson)相关系数用来测量两个变量间的线性相关程度，取值在−1～1 之间，系数取值为 1 表示两个变量完全正相关，系数取值为−1 表示两个变量完全负相关，系数取值为 0 表示两个变量完全不相关. 本文采用 Pearson 相关系数来计算度量间的线性相关性，通过一个度量来反映与其相关的其他度量. 这样就可以只监测一部分度量，就能够推断出其余度量的变化，从而减少监测度量数量，达到降低监测开销的目的. 算法描述如算法 1 所示，具体步骤如下.

算法 1.　关键度量选择.

输入：监测数据集合 $DS=\{p_1,p_2,\cdots,p_i,\cdots,p_n\}$，其中，$p_i=(x,y,\cdots)$，$x,y$ 为度量监测值.

输出：关键度量集合 MS.

1. 计算任意两个度量 x,y 之间的相关系数：

$$r_{xy}=\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})\Big/\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2},$$

其中，$\bar{x}=\sum_{i=1}^{n}x_i$，$\bar{y}=\sum_{i=1}^{n}y_i$.

2. 建立度量相关图 $\boldsymbol{A}_{nn}$：

$$\boldsymbol{A}[i][j]=\begin{cases}1, & r_{ij}^2\in[0.64,1.0]\\ 0, & r_{ij}^2\notin[0.64,1.0]\end{cases},$$

其中，$1\le i,j\le n$，n 为 DS 中数据实例数量.

计算任意度量 i 的权重：

$$w[i]=\sum_{j=1}^{n}\boldsymbol{A}[i][j].$$

3. WHILE(TRUE)

```
{
选取数组 w 中最大值 w[a];
IF(w[a]=−1)
  THEN RETURN(集合 N);
w[a]=−1;
节点集合 a→N;
FOR(j=a+1;j<n;j++)
  IF (A[a][j]=1) THEN
  {
    w[j]=−1;
    FOR(k=k+1;k<n;k++)
    {
      IF(A[j][k]=1) THEN
        w[k]=w[k]−1;
    }
  }
}
```

(1) 计算任意两个度量之间的相关系数，如表 1 所示，当 $r_{xy}\in[-1.0,-0.8]\cup[0.8,1.0]$ 度量 x 和度量 y 强相关.

表 1　相关性定义

相关系数(r)	r^2	相关强度
[0,0.2)	[0,0.04)	不相关
[0.2,0.4)	[0.04,0.16)	一般
[0.4,0.6)	[0.16,0.36)	弱相关
[0.6,0.8)	[0.36,0.64)	相关
[0.8,1.0]	[0.64,1]	强相关

(2) 用 n 节点无向图来描述度量间的相关性，图中节点代表度量，当两度量强相关则节点间有无向边连接，用 $n\times n$ 矩阵来可刻画度量相关图 $\boldsymbol{A}_{nn}$. 每个节点的权重为与其连接的节点数量(边数).

(3) 选择权重最大的节点，删除与其相连的边和节点，并更新图以及各节点权值. 重复以上操作，直到图中节点数目为 0.

算法 1 中，相关系数阈值的设定直接影响到监测的开销以及故障检测与问题定位的准确性. 如果阈值设置过高，则众多度量会被认为不相关，那么这些度量就不能够被其他度量所代表，都会被纳入监测范围之内，从而增加了监测开销. 相反，如果阈值设置过低，众多度量虽然相关度低，但也会被其它度量所代表，当出现与这些度量相关的问题，异常表现也并不明显，从而造成故障的漏报，或者由于缺少这些度量的监测数据而不能准确定位问题原因. 为了保证故障检测以及问题定位的准确性，本文付出部分监测开销的代价. 根据文献[5]，将相关性的阈值设置为 0.8，即当度量间是强相关的时候才将这两个度量做相关处理，以适当减少监测度量.

3.2　异常程度评估

本文利用 PCA 计算监测数据的特征向量以刻画系统运行状态，通过计算当前与历史监测数据特征向量的偏差来评估系统异常程度. 当被监测系统异常程度较高时，缩短监测周期以密切跟踪被监测系统运行状态，从而提高故障(fault)预测与检测的准确性和及时性. 反之，当被监测系统异常程度较低时，延长监测周期，从而降低监测开销. 由于在整个系统运行过程中，故障出现的概率相对较少，动态调整监测周期可以减少大量监测开销.

主成分分析是将 m 个相关变量通过线性变换形成一组较少个数 k($k<m$)的无关成分的多元统计分析方法，这 k 个成分能够表达 m 个变量所要表达的信息，因而常用来进行高维数据降维. 利用 PCA

可以将多个监测度量抽象为少数几个主成分并形成特征向量,将其作为数据分布的方向.度量之间普遍存在着线性关系[3],而 PCA 能够通过特征向量很好的表现度量间的线性相关性.当系统处于正常状态,无论度量值发生多大的变化,各度量值间的线性相关性总会保持稳定,那么监测数据集合的特征向量的主方向也会保持稳定.相反,当系统出现故障,各度量值间的线性关系发生变化,那么监测数据集合的特征向量的主方向也随之发生变化.如果新监测数据为异常点,则加入该点后特征向量会变化,数据分布方向也会发生偏离,可以根据数据分布方向的偏离程度来衡量当前监测数据的异常程度.本文所提出的异常程度评估的具体步骤如下:

(1)监测数据搜集

建立滑动窗口的长度为 n,搜集多度量监测数据为 $X=(x_1,x_2,\cdots,x_m)$,其中,每次搜集的监测数据包括 m 个度量(运维人员可以根据需要设定 m 值,m 为正整数),x_i 为第 i 个度量的值,将监测数据按时间先后顺序存入滑动窗口,将滑动窗口中的监测数据组成 n 行 m 列矩阵 $\boldsymbol{A}_{nm}$;

(2)运行状态刻画

①将 $\boldsymbol{A}_{nm}$ 的每一列的监测度量进行标准化处理,使其均值为 0,方差为 1,$z_i=(x_i-\mu_i)/\sigma_i$,其中,μ_i 为第 i 列数据集合的均值,σ_i 为第 i 列数据集合的标准差.

②求出协方差矩阵:$\boldsymbol{\Sigma}_{\boldsymbol{A}}=\begin{bmatrix}\sigma_{11}^2 & \cdots & \sigma_{1m}^2\\ \vdots & \ddots & \vdots\\ \sigma_{m1}^2 & \cdots & \sigma_{mm}^2\end{bmatrix}$,其中,$x_i$ 和 x_j 的协方差 $\sigma_{ij}^2=\sum_{k=1}^{n}z_{ki}z_{kj}/n$,表现两个变量的相关性.

③计算 $\boldsymbol{\Sigma}_{\boldsymbol{A}}$ 的特征向量,作为数据分布的主方向 u.

(3)运行状态更新

①新的监测数据 x_t 到来时,为了放大离群点对主方向改变的影响,将样本复制 nr 次,其中 $r\in[0,1]$ 是当前样本的复制次数与当前样本大小的比例,得到更新矩阵:$\widetilde{\boldsymbol{A}}=\boldsymbol{A}\cup\{x_t,x_t,\cdots,x_t\}$.

②更新矩阵平均值和协方差矩阵:$\tilde{\boldsymbol{\mu}}=\dfrac{\mu+rx_t}{1+r}$,$\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{A}}}=\dfrac{Q}{1+r}+\dfrac{r}{1+r}x_t\boldsymbol{x}_t^{\mathrm{T}}-\tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^{\mathrm{T}}$,$\boldsymbol{Q}=\dfrac{\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}}{n}$.更新特征向量主方向为

$$u_t=\frac{\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{A}}}u}{\|\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{A}}}u\|}\tag{2}$$

由式(2)可知,只需要记录上一次的平均值,时间和空间复杂度为 $O(p)$,其中 p 是样本的维数.

(4)异常程度评估

使用余弦相似度来代表两个主特征向量方向的偏离,以评估新搜集监测数据的异常程度.两个主特征向量相似度越低,偏离程度越大,异常程度越高,那么本文描述异常程度为

$$\mathrm{AnoSig}_t=1-\left|\frac{\langle u_t,u\rangle}{\|u_t\|\|u\|}\right|\tag{3}$$

在云计算环境下,负载具有动态性,不断发生变化,主要体现在并发数量和负载模式[6].在并发数量变化方面,本文利用 PCA 刻画度量之间的相关性,通过分析相关性的变化来检测异常状态,而并发数量变化并不会造成度量相关性的改变,因此不会影响本文所提出方法的准确性.在应对负载模式变化方面,本文此前工作[7]提出一种基于在线增量式聚类的负载模式识别算法,在运行过程中学习并匹配负载模式,由于篇幅限制不在本文赘述,本文利用该方法来识别负载模式.负载模式的变化会引起度量相关性的变化,从而导致关键度量的改变.同时,由于负载具有稳定性和周期性[8],算法 2 首先识别当前时段的负载模式,在特定负载模式下,监测相应的关键度量值,进而与基准度量向量比较以检测系统故障.

算法 2. 异常程度评估.

输入:当前监测数据点为 $p_t=(x,y,\cdots)$,x,y 为监测度量,当前负载向量为 $\boldsymbol{wv}_t$.

输出:当前系统异常程度 AnoSig_t.

1. 建立以下数据结构:

①运行状态集合:$ss=\{s_1,s_2,\cdots\}$,其中,$s_i=(wp_i,ms_i,fd_i)$,wp_i 为第 i 种负载模式,ms_i 为第 i 种关键度量集合,fd_i 为 wp_i 所对应的数据主方向;

②负载模式集合:$wps=\{wp_1,wp_2,\cdots\}$,其中,wp_i 为第 i 种负载模式;

③滑动窗口:

$SW=\{(p_1,\boldsymbol{wv}_1),(p_2,\boldsymbol{wv}_2),\cdots,(p_i,\boldsymbol{wv}_i),\cdots,(p_n,\boldsymbol{wv}_n)\}$,其中,$p_i$ 为第 i 个监测数据点,$\boldsymbol{wv}_i$ 为第 i 个负载向量,n 为滑动窗口大小.

2. 更新滑动窗口:

当前监测数据 p_t,当前负载向量 $\boldsymbol{wv}_t$,利用$(p_t,\boldsymbol{wv}_t)$更新 SW.

3. 识别当前负载模式:$wp_t=PR(\boldsymbol{wv}_t)$;

计算 SW 中数据集合的特征向量:$\boldsymbol{fd}_t$.

4. IF（$wp_t \in wps$）//判断当前负载模式是否与已知的负载模式相匹配{
计算当前状态异常程度：
$\text{AnoSig}_t = \text{CalAnomaly}(\boldsymbol{fd}_t, fd_i)$；
}
ELSE{
根据算法1，计算当前负载模式下的关键度量集合：ms_t
$wps = wp_i \cup wps$；
$s_t = (wp_t, ms_t, \boldsymbol{fd}_t)$；
$ss = s_t \cup ss$.
}

3.3　监测周期动态调整

当系统运行环境处于不断变化过程中，故障触发或者多线程竞争资源等原因，会导致系统出现随机故障，此类故障只与运行环境相关而与运行时间没有关系，那么系统出现这类故障符合泊松过程[9]. 因此，本文采用指数分布来建模预测出现故障的时间点. 系统故障包括瞬时故障和累积故障[10]，前者是偶然触发了某种条件，比如在 spinlock 错误中，某时刻多线程/进程竞争共享资源，造成循环等待的死锁场景，表现为 CPU 利用率瞬间上升；后者是量变导致质变，比如在内存泄漏错误中，线程每次执行创建对象，占用内存空间，而不能及时释放，造成执行程序由于得不到足够的内存空间而最终导致不能正常执行，表现为内存缓慢增加. 泊松过程是可靠性工程经典的故障预测模型，根据历史故障数据来预测下次出现故障的时间[9]，本文对模型进行了改进，引入异常程度评估以替代历史故障数据，预测下次系统故障时间. 这样，通过评估系统异常程度，对于瞬时错误可以及时做出响应，将监测周期立即调整到最小值；对于累积错误可以根据异常程度来逐渐调整监测周期.

随机变量 N 为在 x 秒内出现故障的数量，如果系统出现故障的频率为每秒 λ 次，N 符合均值为 λx 的泊松分布：$P(X>x)=P(N=0)=e^{-\lambda x}, x\geq 0$.

X 的累积分布函数为 $F(x)=P(X\leq x)=1-e^{-\lambda x}, x\geq 0$. 其中，$X$ 是以 λ 为参数的指数随机变量，表示泊松过程中的连续出现故障的时间间隔；λ 为泊松过程中单位时间内平均出现故障的次数. 由于在泊松过程中，一定时间间隔内出现一定数量故障的概率只与间隔时间长短有关，X 的开始时间点的选取与预测故障发生的时间点无关.

设系统出现故障的概率为 $F(t)=w$，那么可以由此计算出下一次出现故障的时间间隔：$t=-\ln(1-w)/\lambda, x\geq 0$，因此本文将当前的监测周期调整为

$$T=\begin{cases} T_\beta, & 0\leq s_t<\beta \\ \ln\left(\dfrac{1}{1-s_t}\right)\Big/\lambda+e, & \beta\leq s_t\leq\alpha \\ T_\alpha, & \alpha<s_t\leq 1 \end{cases} \tag{4}$$

其中：T_β 为最小监测周期；T_α 为最大监测周期；λ、e 为调整参数.

以下对其取值进行讨论：

(1) 最小监测周期 T_β，需要考虑系统允许的监测所带来的最大开销，同时可以基于经验值或由系统当前负载所决定，例如，负载为 50 个请求每分钟，那么如果监测周期设定为 1 s，则不能够得到所期望的监测值.

(2) 最大监测周期 T_α，需要考虑系统检测故障的及时性，例如，若设定 α 为 60%，就意味着在两次监测之间有 60% 的概率系统已经出现了故障.

(3) 由于当 $s_t=\omega_\beta$ 时，$T=T_\beta$，当 $s_\alpha=\omega_\alpha$ 时，$T=T_\alpha$，因此只需要设定 T_α、T_β，并将其代入式(4)中，即可以计算得到参数 λ、e 的取值.

对函数进行分析可以得到，监测周期在设定的最大监测周期和最小监测周期之间，随着系统异常程度增加而缩短，并且随着异常程度的加剧监测周期缩短的幅度增加，即异常越严重监测周期缩短得越快，这是期望得到的结果.

本文量化了异常程度而不是简单地判断异常与否，通常情况下异常程度也是逐渐变化的连续过程，监测周期也随异常程度不断动态变化，能够反映系统异常变化的趋势. 同时，如果在监测周期动态调整的过程中出现了突发性的异常，监测周期的最大阈值也保证了故障检测的准确性和及时性并不会受到太大影响.

4　实验评价

4.1　实验环境

本文基于由中国科学院软件研究所自主研发的类似于亚马逊 EC2 的云计算服务平台 OnceCloud，部署开源的典型多层电子商务基准测试系统 Bench4Q①，搭建实验环境. 我们将所提出的动态监测方法应用在该实验平台，向系统中随机注入故障以验证方法的有效性. 具体实验环境如图 2 所示，包括 4 台刀片服务器构成，每台服务器启动若干 Docker 容器，部署应用程序，系统各组件配置信息如表 2 所示.

① Bench4Q. http://www.ow2.org/view/ActivitiesDashboard/Bench4Q

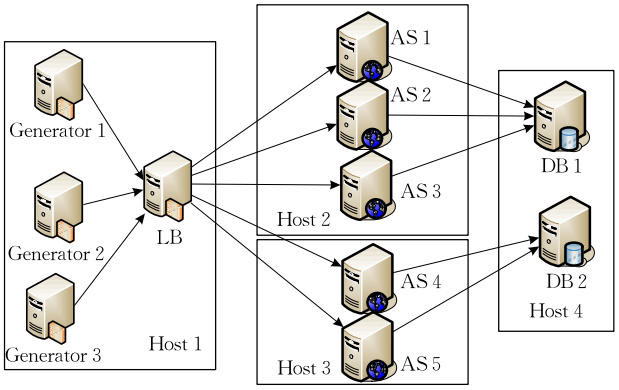


图 2 实验架构

表 2 实验配置

主机	硬件配置	软件配置
1	Intel Xeon CPU E5645, 2.40GHz, 24 Cores, 24GB RAM	操作系统:Centos 7.1; 负载生成器:Bench4Q 1.0; 负载均衡器(LB):Apache 2.4.18
2、3	Intel Core™ i5-3470, 3.20GHz, 24 Cores, 16GB RAM	操作系统:Centos 7.1; 应用服务器(AS):Tomcat 9.0; Web 应用:Bench4Q 1.0
4	Intel Core™ i5-3470, 3.20GHz 24 Cores, 32GB RAM	操作系统:Centos 7.1; 数据库服务器(DB):MySQL 5.6

Bench4Q 是遵循 TPC-W 规范①的基准测试应用,采用企业应用的典型 3 层架构,其中包括 3 种常用的开源服务器系统,用负载均衡器 Apache②,进行请求分发,用应用服务器 Tomcat③集群处理业务逻辑,用数据库服务器 MySQL④提供数据操作. TPC-W 是当前广泛采用的企业应用基准测试规范,遵循该规范的 Web 应用实现了在线电子商务应用. TPC-W 定义了 3 种负载模式,包括浏览、查询和购买商品. 客户在一个会话中通过一些列连续操作来访问 Web 站点. 为了模拟实际云计算环境下负载变化的动态性,本文扩展了 Bench4Q 原有的负载发生器,并发数量与负载模式在运行过程中动态变化.

本文利用滑动窗口技术临时存储监测数据,主要实现两方面的功能. 首先,计算当前滑动窗口中的度量向量集合的特征向量,将其与基准特征向量进行比较以检测故障;其次,根据当前滑动窗口中的负载向量集合学习新的负载模式,在特定负载模式下检测故障. 每当新的监测数据到来则加入滑动窗口末尾,并将窗口最前的监测数据删除. 窗口长度越大,故障表现越不明显,故障检测的及时性越差,从而导致较高的漏报率;相反,窗口长度越小,故障表现越显著,但噪音监测数据所造成的影响会越大,从而导致较高的误报率. 实验结果表明,窗口长度在 15～25 的时候,故障检测的效果较为理想,且差别不大,因此在此实验中将滑动窗口大小设置为 20,由于篇幅限制本文未给出具体实验数据.

4.2 实验结果

4.2.1 监测度量选择

在监测数据搜集方面,本文对开源分布式监测系统 Zabbix⑤进行扩展,从处理器、内存、磁盘和网络 4 类资源搜集到 26 个系统监测度量. 对于每类资源,利用相关分析选取其中的关键度量进行监测分析,最终选择的结果如表 3 所示,进行在线分析,以评估系统异常程度.

表 3 Docker 关键监测度量

资源类型	度量索引值	说明
CPU	system_cpu_usage	系统调用所占 CPU 时间片
网络	rx_packets	接收的数据包个数
	tx_errors	发送的数据包个数
内存	used	当前容器已使用内存量
	pgpgin	从磁盘或 swap 置换到内存的字节数
磁盘	read	读取的字节数
	write	写入的字节数

由于篇幅限制,本文仅以网络相关度量为例. 网络相关搜集到的度量共 8 项,如表 4 的网络监测度量.

表 4 网络度量说明

度量索引	说明
tx_packets	发送的数据包个数
rx_bytes	收到的字节数
tx_bytes	发送的字节数
rx_errors	接收数据时出现的错误数
tx_errors	发送数据时出现的错误数
rx_dropped	接收数据时丢弃的包的个数
tx_dropped	发送数据时丢弃的包的个数
rx_packets	收到的数据包个数

根据 3.1 节所提出的方法,计算这些度量的相关性如图 3 所示,最后得到关键度量 rx_packets 和 tx_errors 进行进一步的监测分析.

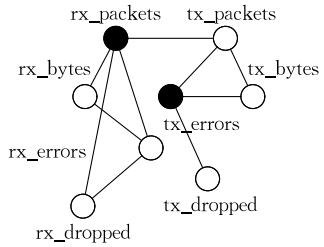


图 3 网络度量相关性

① TPC-W. http://www.tpc.org/tpcw
② Apache. http://httpd.apache.org
③ Apache Tomcat. http://tomcat.apache.org
④ MySQL. http://www.mysql.com
⑤ Zabbix. http://www.zabbix.com

4.2.2　动态负载的适应性

在云计算环境下，负载动态变化，而负载由并发数量与负载模式共同刻画[6]. 本文通过模拟不断变化的负载来验证 PCA 能够适应负载的动态变化，即系统在正常运行过程中，虽然并发数量以及负载模式在不断改变，但由 PCA 计算得到的特征向量的主方向依然保持稳定.

在本实验中，并发数量由 0 以步长为 2 逐渐增加到 300，每种并发数量持续 5 min，实验持续 750 min，每 5 min 搜集一组监测数据，可以获取 150 组数据. 本文中滑动窗口长度设置为 20，那么本文将前 20 组数据用作基准训练数据计算 PCA 特征向量的主方向，此后的数据可用作在线故障检测.

本文将异常程度阈值设置为 0.2，即当高于阈值则判定为异常. 本文根据系统异常程度，动态调整监测周期，与阈值无关，不影响监测周期调整. 该阈值只是与发出异常警报相关，如果阈值设定过高，系统异常程度升高而不报警，从而影响检测的及时性；如果阈值设定过低，系统处于正常状态也会报警，从而造成误报. 在实际系统中，报警阈值通常根据系统管理员经验来设定，例如当前系统管理员会设定 CPU 利用率阈值为 95%，当超过才报警. 本文利用离线测试将正常运行状况下 95%数据所处区域的异常值设定为阈值，离线测试结果可参考图 4.

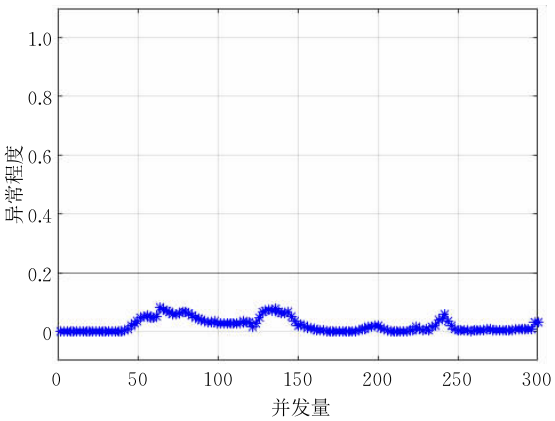


图 4　异常程度随并发量变化

(1) 负载模式不变，并发数量变化

负载模式保持 Browsing 不变，实验结果如图 5 所示，横坐标为并发数量，纵坐标为异常程度为(1－|余弦相似度|)，取值在 0～1 之间，异常程度最大值为 1，表示系统已出现问题，最小值为 0，表示系统状态完全正常. 在负载模式不变，并发数量变化的情况下，没有正常监测数据检测为异常情况的误检现象的发生. 因此，对并发数量动态变化的情况有较好的适应性.

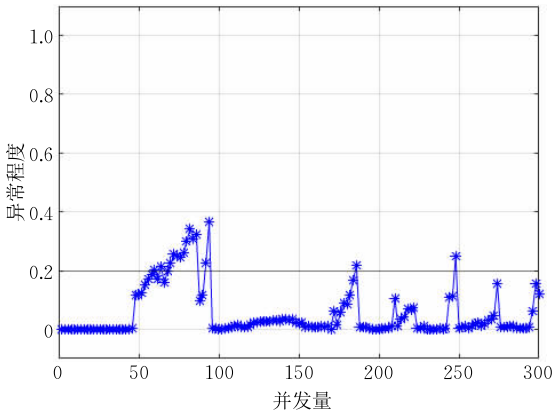


图 5　异常程度随负载变化

(2) 并发数量和负载模式都发生变化

负载模式以 Browsing、Shopping、Ordering 3 种方式顺序变化，每种负载并发递增 50，即 Browsing(并发：1～50，151～200)、Shopping(并发：51～100，201～250)、Ordering(并发：101～150，251～300). 实验结果如图 5 所示，在并发数量和负载模式变化的情况下，15 个数据点检测为异常，误检率为 5%. 因此，所提出的方法具有较好的鲁棒性，能够适应负载不断变化的云计算环境.

4.2.3　异常程度评估的准确性

由于目前尚无云应用故障注入的 Benchmark，为了保证注入故障的典型性，本文参考文献[11]分析 Web 应用服务失效的常见原因，选取 Web 应用的 4 类典型故障进行注入，包括 CPU、磁盘、网络和内存，如表 5 所示. 系统运行过程中，当所注入的额外操作代码被调用则触发故障. 这些故障本文通过所注入的故障代码调用压力测试工具的命令行来实现. 对于 CPU 和磁盘等资源占用，本文利用 StressLinux① 来

表 5　故障注入列表

资源类型	故障类型	故障描述	执行的操作
CPU	进程死锁	模拟多进程竞争共享变量，造成CPU占用过高	额外操作分别占用CPU时间：(1) 20%，(2) 40%，(3) 60%(StressLinux)
磁盘	I/O干扰	创建Docker容器，部署应用执行磁盘I/O操作，造成I/O性能干扰	额外操作使用2个线程执行sync()函数，另一个线程执行write()函数，分别向磁盘写入：(1) 128 MB，(2) 256 MB，(3) 384 MB(StressLinux)
网络	网络传输拥塞	模拟发包请求占用一定量的网络带宽	额外操作通过向其他网络节点发送数据包分别占用当前节点：(1) 20%，(2) 40%，(3) 60%的网络带宽(iPerf)
内存	内存泄露	创建对象引用静态变量，造成该对象不被垃圾回收，逐渐耗尽内存资源	每当故障触发则创建对象，并指向全局静态变量，以免被垃圾回收，逐渐占用2 GB内存

① StressLinux. http://www.stresslinux.org/sl/

实现,其广泛用于压力测试和调试系统组件失败.对于网络资源占用,本文利用网络性能测试工具 iPerf① 来实现,其可以通过 TCP 和 UDP 发包来占用带宽资源.

本实验中,并发数量由 0 以步长为 2 逐渐增加到 240,每种并发数量持续 5 min,负载模式从 Browsing、Shopping 及 Ordering 3 种模式中随机选取,每 5 min 搜集一组监测数据,每个实验持续 600 min,可以获取 120 组数据.本文中滑动窗口长度设置为 20,那么本文将前 20 组数据用作基准训练数据计算 PCA 特征向量的主方向,此后的数据可用作在线故障检测.在每个实验中注入一种错误,实验结果如图 6～图 8 所示,运行过程包括以下阶段:正常运行(并发:0～60)、注入故障(1)(并发:61～120)、注入故障(2)(并发:121～180)、注入故障(3)(并发:181～240),在 CPU、磁盘、网络等相关故障注入系统后,随着注入故障的严重程度增加,系统的异常程度也逐渐增加.因此,本文所提出方法可以检测到故障所引起的系统异常,并且能够反映异常的严重程度.

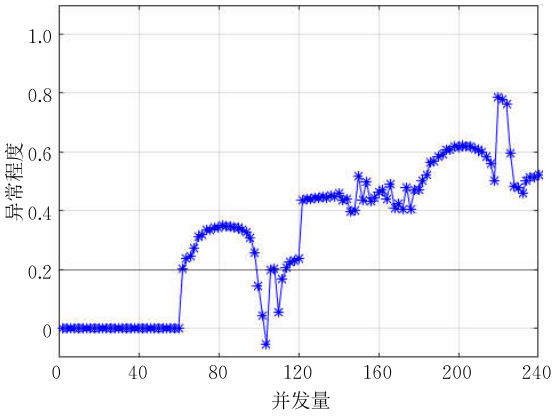


图 6 CPU 相关故障注入

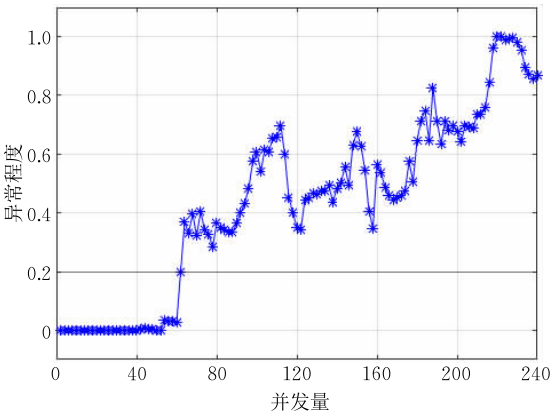


图 7 磁盘相关故障注入

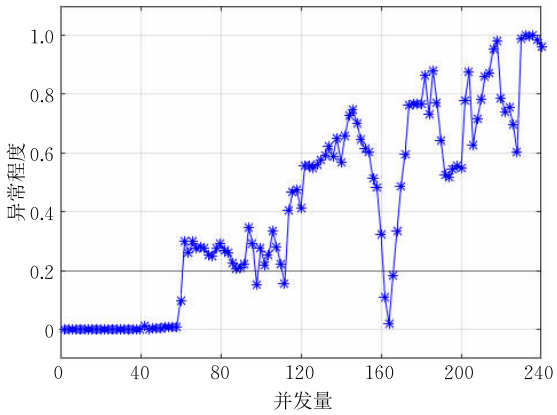


图 8 网络相关故障注入

4.2.4 监测开销对比

在不注入错误的情况下,实验持续 60 min.固定监测周期与动态调整监测周期的监测方法对应的 CPU 利用率如图 9(a)所示.从图中可以看出,动态调整监测周期的方法对应的 CPU 利用率始终维持在较低水平.这是由于系统运行状况稳定,异常程度较低,监测系统采用了较大的监测周期,监测数据收集次数较少.从图 9(b)可以看出监测周期的变化对网络流量的影响.固定监测周期的方法每分钟内发送的网络流量始终维持在固定范围内,而动态调整监测周期的方法由于系统异常程度较低,会逐渐采用较大的监测周期,进行数据收集的次数减少,因此每分钟内发送的网络流量呈逐渐减少的趋势.

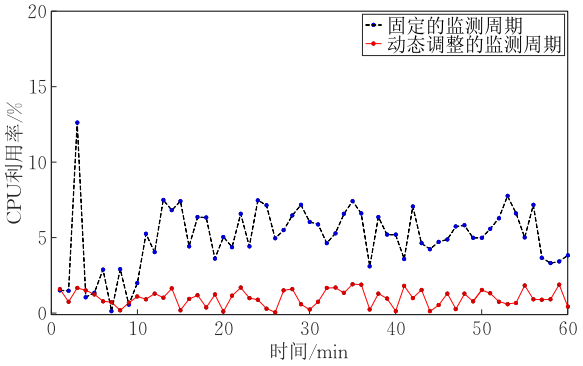
我们从 MySQL 的 bug 列表中选取 MySQL 5.5.5-m3 版本的 54332 号错误.该错误的产生是由于两个连接访问同一个表,每个连接都用 LOCK TABLE WRITE 将表锁定,在另一个表上执行 INSERT DELAYED 会产生死锁.该错误的注入方法如表 6 所示.

表 6 MySQL5.5.5-m3 的 54332 号错误注入方法

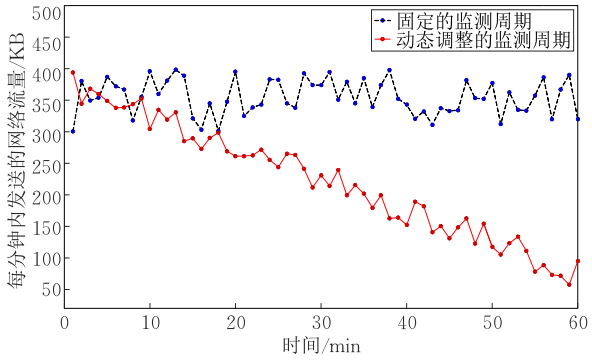
```
CREATE TABLE t1 (a INT);
CREATE TABLE t2 (a INT);
connect (con1, localhost, root);
LOCK TABLE t1 WRITE;
connection default;
LOCK TABLE t2 WRITE;
—send INSERT DELAYED INTO t1 VALUES (1)
connection con1;
—sleep 1
INSERT DELAYED INTO t2 VALUES (1);
```

在注入错误以后,不同监测方式对应的 CPU 利用率如图 9(c)所示,从图中可以看出,在 30 min～55 min 和 90 min～115 min 这两个注入错误的时间

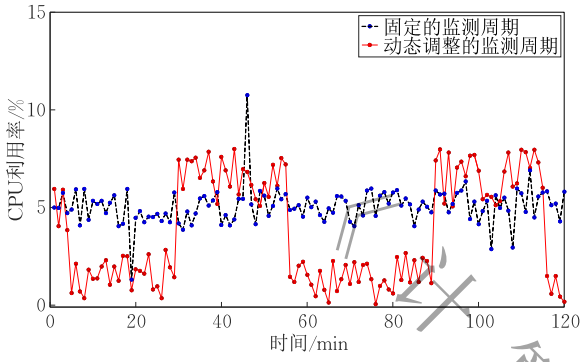
① iPerf. https://iperf.fr/



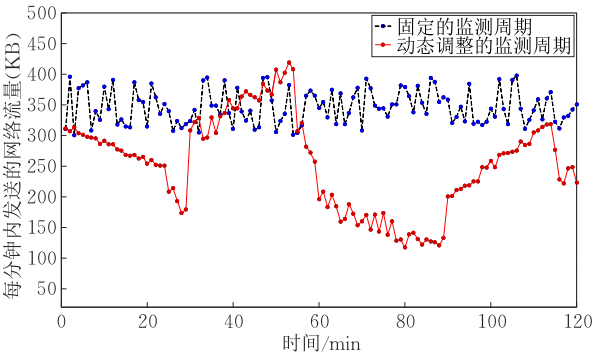
(a) 不注入错误时CPU利用率对比



(b) 不注入错误时网络流量对比



(c) 注入错误时CPU利用率对比



(d) 注入错误时网络流量对比

图 9　监测开销对比

段，动态调整监测周期的方法比采用固定监测周期的方法消耗的 CPU 略高，因为这两个时间段系统异常程度较高，动态调整监测周期的方法会使监测系统采用更小的监测周期，执行更多的监测数据收集操作. 在其他时间段系统异常程度较低，监测系统使用较小的监测周期，因此 CPU 利用率要低于采用固定监测周期的方法. 在整个实验过程中，动态调整监测周期方法的总 CPU 利用率低于固定监测周期的方法，在系统运行状况相对稳定的时候这种优势更加明显.

不同监测方式对应的网络流量的对比分别如图 9(d)所示，从图中可以看出，在错误注入的时间点，由于系统异常程度突然变高，动态调整监测周期的方法会通知监测系统立即采用较小的监测周期，因此动态调整监测周期的方法每分钟内发送的网络流量会呈现快速增长的趋势；当系统异常程度较低的时候，动态调整监测周期的方法使得监测系统逐渐采用较大的监测周期，数据收集次数减少，因此每分钟内发送的网络流量在不存在错误的时间段呈减少的趋势. 固定监测周期的方法每分钟内发送的网络流量不随错误的注入与消除产生较大的变化. 整个实验过程中，动态调整监测周期的方法发送的网络流量低于固定监测周期的方法.

4.2.5　实例分析：内存泄露

本节以表 3 中内存泄露故障为例，来分析与固定监测周期的方法相比，本文所提出方法在故障检测有效性、及时性以及监测开销方面的优势. 在实验中，根据经验本文将最大、最小监测周期分别设置为 300 s、10 s. 最初系统异常值为 0，则以最大监测周期 300 s 进行监测，本文在搜集到第 30 个监测数据后注入故障，由图 10 可以看到异常程度的变化，由图 11 可以看到监测周期动态调整情况.

(1) 故障检测的有效性

如图 10 所示，在第 30 个监测点注入故障后，由于内存泄露是逐渐严重的过程，异常程度总体上呈上升趋势，并且在第 49 个监测点达到了异常报警的条件(连续 10 个点超过阈值 0.2). 因此，所提出方法能够有效的检测到所注入的内存泄露故障.

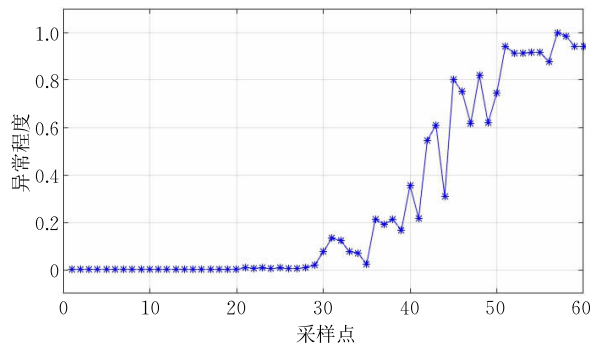


图 10　内存泄露故障注入

(2)故障检测的及时性

如图 11 所示,在故障注入后,系统异常程度不断增加,监测周期不断减小.那么,从注入故障到检测到故障(监测点 31～49)本文所提出的方法经历了 1959 s.对于传统固定周期的监测方法,如果设定监测周期为 150 s(即最大监测周期的 1/2),获取同样数量的监测数据则需要经历 2850 s.因此,在系统出现异常的情况下,本文所提出方法所需要的检测时间为传统方法的 68.7%.

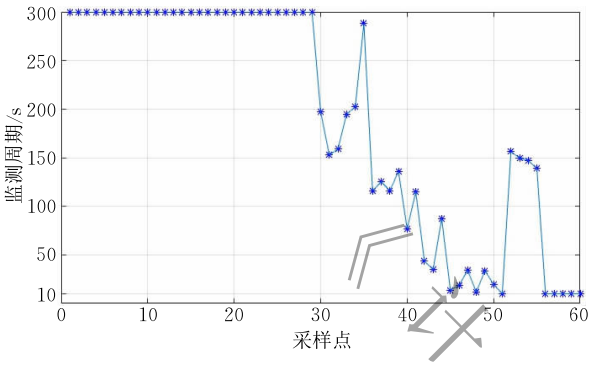


图 11 监测周期调整

(3)监测开销比较

如图 11 所示,在未进行故障注入时,本文所提出方法的监测周期始终保持最大值,搜集监测数据量为传统固定周期监测方法的一半.显然,在系统正常运行的情况下,本文所提出方法的监测开销为传统方法的 50%.在实验中,第 1～30 个监测点,系统处于正常运行状态,在第 30 个监测点注入错误,在第 49 个监测点达到了如(2)中定义的故障报警条件.那么从第 31～49 个监测点,共搜集了 19 个监测点,经历了 1959 s,而对于固定周期方法,这个期间搜集监测点的数量为 14 个,本文方法的监测开销为固定周期方法的 135.71%.因此,本文方法监测开销在正常状态下为固定周期方法的 50%,在故障状态下多于固定周期方法的 35.71%,以监测开销的提升换来了如(2)中分析的故障检测及时性提升 31.3%.

5 相关工作

在本文中,我们提出一种自适应监测方法,根据云计算系统运行状态自适应动态调整监测对象和监测周期,以减少系统在正常运行状态的监测开销,并提高系统在异常状态的错误检测及时性.论文面向的是云计算系统的监测,涉及到监测的两个要素:对象的选择和周期的调整.对于前者,本文根据历史数据分析度量间的相关性从而选取反映系统运行状态的关键度量;对于后者,本文根据系统异常程度适应性动态调整监测周期,涉及到了系统异常程度评估方法以及监测周期调整策略两个方面.由以上分析可以看出,本文涉及到的相关工作主要包括:面向云计算系统的监测框架、监测对象选择、系统异常程度评估和监测周期调整 4 个方面.因此,本节从以上各方面,分别介绍和比较了相关工作.

5.1 云计算系统监测框架

当前面向云计算系统的监测框架通常采用中心式的,即在每个需要监测的主机或者虚拟机上部署代理程序(Agent),然后将监测数据搜集并存储到中心服务器,包括开源工具包括 Zabbix、Nagios① 和 Ganglia②等;此外商业工具有 IBM Tevoli③、HP OpenView④ 和 LiveAction⑤ 等.然而,大量监测数据的搜集和传输会带来巨大的监测开销,给中心节点带来巨大压力.近来一些工作研究了分布式架构.文献[12]提出一种基于分布式监测器的分层监测架构,具有较强的可扩展性以应对大规模系统实时监测与分析需求.为了应对该问题,文献[13]分析了在不同应用场景下,以推(Push)和拉(Pull)模式获取监测数据在效率方面的差别,提出了面向用户需求,结合推拉模式的检测模型以降低监测开销.文献[14]提出一种采用点对点(Peer-to-Peer,P2P)可扩展弹性的分布式监测系统,可以动态的添加和删除需要监测的数据源,适应于不断变化的云计算环境.本文所提出的方法可以整合到以上监测框架,通过在线分析目标节点的监测数据来故障检测,进而调整监测周期以减少开销.近年来一些工作面向云计算环境的特定资源以及特定需求,提供了监控机制.VM-Driver 是将感知 VMM 事件和构建 VM 语义相分离.其通过 Hypervisor 搜集监测度量,与目标 VM 的类型和版本无关,可以监测所有 Guest OS[15].文献[16]采用 REST 形式以树形结构对云计算系统中的各种资源(如计算、网络和存储)进行建模,从而便于用户搜集.文献[17]将监测数据以模型的形式进行刻画,根据用户对监测资源的关注点不同,为不同用户提供差异化视图.mOSAIC 面向不同云应用,为用户提供了获取可定制资源的监测接口[18].

① Nagios, http://www.nagios.org/
② Ganglia, http://ganglia.info/
③ Tivoli, http://www-03.ibm.com/software/products/en/tivosystautoformult/
④ OpenView, http://www8.hp.com/us/en/software-solutions/operations-manager-infrastructure-monitoring/
⑤ LiveAction, http://liveaction.com/

MonPaaS 区分云平台的服务提供者和消费者，使得云服务提供者能够看到完整的云框架，同时将监测作为一种服务提供给云服务消费者，使其能够自动看到其租用的云资源，并且定义其他所需要监测的对象[19]. 本文所提出的方法通过调整监测对象和监测周期来减少监测开销，可以有机地整合到已有的云计算系统监测框架中.

5.2　监测对象选择

监测系统通常为各监测对象设定约束条件，通过检测冲突来发现系统故障. 这种方式通常需要搜集网络中各节点的监测数据，聚集后统一处理（如计算 SUM 值），因而减少监测数据传输的通信开销成为重要问题. 许多工作将冲突条件分解到各个节点，对一定周期内某种事件发生的概率进行估计，调整局部的约束条件[20-22]. 对于网络监测，通常采用同样的方式随机搜集部分监测数据，检测是否与预设条件相冲突[23]. 在已有方法中，由于约束条件通常需要领域知识，系统管理员难以手工设定. 本文基于相关分析挖掘各度量间的相关性，通过监测部分关键度量就可以推断出整个系统的运行状态，监测对象根据历史数据自动选择，无需系统管理员根据系统的特点以及领域知识人为手工设定.

5.3　监测周期调整

监测周期动态调整的工作并不多，与本文最为相关的工作是文献[24]. 作者为了减少通信开销，根据监测度量波动程度来调整监测周期，波动越大监测周期则调整得越短. 同时，监测周期调整简单地借鉴了 TCP 拥塞控制“缓慢开始，快速下降”的思想. 该方法存在两个问题. 首先，在云计算环境下，负载量即客户请求的数量是在不断变化的，会相应造成度量波动，因此度量波动程度并不代表系统出现问题；其次，TCP 拥塞控制的机制不能充分考虑到异常程度，周期调整幅度与异常程度变化不相适应. 本文利用 PCA 考虑各度量间的相关性来评估系统异常程度，而并不只是关注于某个度量，适用于负载动态变化的云计算环境. 同时，建立软件可靠性模型，根据对异常程度的评估以及故障出现时间的预测动态调整监测周期，使得周期调整幅度与异常程度变化相适应.

5.4　异常程度评估

故障检测和预测技术通常对系统的日志或者监测数据进行分析以评估系统的异常程度. LogMaster[25] 解析日志文件，挖掘事件间的相关性，通过监测其变化来预测系统故障. 文献[26]利用贝叶斯网降维，同时利用信息熵和 Euclidean 距离来检测与其他节点行为不一致的节点. 为了及早避免故障的发生，一些工作引入了时间序列模型，通过对一些度量变化趋势的分析来预测故障. 文献[27]利用 Wilcox Rank-sum 对监测数据进行趋势分析，以预测处理器资源利用情况. 文献[28]利用时间相关异常方法 SIGs（Structure-of-Influence Graphs）来推断交互组件间的影响. PCA 技术由于具有较低的时间复杂度，环境适应性较好，也被用于系统管理. 文献[29]利用 PCA 和 ICA 进行特征提取，通过降维来减少需要分析的监测度量的数量，进而利用离群点检测来辨别集群系统中的故障节点. 该方法采用 PCA 对监测数据降维，而并不直接用来检测异常状态. 文献[30]结合源码分析与信息提取技术来解析日志以创建组合特征，由于同组特征向量中日志信息的各维度间存在高相关性，该方法利用 PCA 静态分析与检测相关系数偏离的日志信息. 已有方法通常采用较为复杂的统计学模型，具有较高的时间复杂度，不适合在线监测的场景. 同时，需要设定较为复杂的参数，这些参数随应用场景而变化，若不能选择合适的参数将会影响模型的准确性. 本文根据最近而非全部的监测数据，利用增量式 PCA 动态更新特征向量，并且基于余弦相似度而非距离来评估系统异常程度. 本文所提出的方法具有较低的计算复杂度，能够适应负载的动态变化，适用于在线评估系统异常程度.

6　结　论

运行监测是云计算平台提供的一种重要基础服务，对于保障云计算系统性能与可靠性至关重要. 然而，频繁地搜集、传输、存储和分析海量监测数据会带来巨大性能开销以及云用户费用支出. 相反，如果监测周期过大，则降低了故障检测的及时性，且不能搜集足够的监测数据以供故障分析与定位. 本文面向云计算系统，提出一种基于故障检测的自适应监测方法. 该方法考虑了监测成本、监控需求和故障概率等因素. 监测成本是由监测对象和监测周期两个方面所决定的，本文提出了关键度量选择方法以选择监测对象，以及监测周期动态调整方法，根据故障发生的概率动态调整监测周期，从而达到了在低故障概率状态下具有较低监测开销，在高故障概率状态下具有较高检测及时性的监测要求. 在监测对象选择方面，利用相关分析计算度量间相似性以建立

度量关联图,基于此从众多度量中选取反映系统运行状态的关键度量.在监测周期调整方面,基于PCA分析主特征向量偏差程度以评估检测系统异常程度,进而结合可靠性模型动态调整监测周期.实验结果表明,本文所提出的方法能够适应云环境下负载的动态变化,准确评估系统异常程度,自动调整监测频率以提高系统在异常状况下故障检测的准确性与及时性,同时降低系统在正常运行过程中的监测开销.

参 考 文 献

[1] Patterson D A. A simple way to estimate the cost of downtime// Proceedings of the 16th USENIX Conference on System Administration. Philadelphia, USA, 2002: 185-188

[2] Huebscher M C, McCann J A. A survey of autonomic computing-degrees, models, and applications. ACM Computer Surveys, 2008, 40(3): 1-28

[3] Jiang G, Chen H, Yoshihira K. Modeling and tracking of transaction flow dynamics for fault detection in complex systems. IEEE Transactions on Dependable and Secure Computing, 2006, 3(4): 312-326

[4] Munawar M A, Ward P A S. A comparative study of pairwise regression techniques for problem determination// Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research. Toronto, Canada, 2007: 152-166

[5] Evans J D. Straightforward Statistics for the Behavioral Sciences. Pacific Grove, USA: Brooks/Cole Pub Co, 2005

[6] Wang T, Wei J, Zhang W, et al. Workload-aware anomaly detection for web applications. Journal of Systems and Software, 2014, 89(1): 19-32

[7] Wang T, Zhang W, Wei J, et al. FD4C: Automatic fault diagnosis framework for web applications in cloud computing. IEEE Transactions on Systems, Man and Cybernetics: Systems, 2016, 46(1): 61-75

[8] Williams A, Arlitt M, Williamson C, Barker K. Web workload characterization: Ten years later. Web Content Delivery, 2005, 2(1): 3-21

[9] Shibata K, Rinsaka K, Dohi T. Metrics-based software reliability models using non-homogeneous poisson processes// Proceedings of the 17th International Symposium on Software Reliability Engineering. Raleigh, North Carolina, 2006: 52-61

[10] Salfner F, Lenk M, Malek M. A survey of online failure prediction methods. ACM Computer Surveys, 2010, 42(3): 1-42

[11] Pertet S, Narasimhan P. Causes of failure in web applications. Parallel Data Laboratory, Carnegie Mellon University: Technical Report CMU-PDL-05-109, 2005

[12] Andreolini M, Colajanni M, Pietri M. A scalable architecture for real-time monitoring of large information systems// Proceedings of the 2nd Symposium on Network Cloud Computing and Applications. London, UK, 2012: 143-150

[13] Huang H, Wang L. A combined push-pull model for resource monitoring in cloud computing environment//Proceedings of the 3rd IEEE International Conference on Cloud Computing. Miami, USA, 2010: 260-267

[14] Konig B, Calero J M A, Kirschnick J. Elastic monitoring framework for cloud infrastructures. IET Communications, 2012, 6(6): 1306-1315

[15] Xiang G, Jin H, Zou D, et al. VMDriver: A driver-based monitoring mechanism for virtualization//Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems. Delhi, India, 2010: 72-81

[16] Han H, Kim S, Jung H, et al. A RESTful approach to the management of cloud infrastructure//Proceedings of the IEEE International Conference on Cloud Computing. Bangalore, India, 2009: 139-142

[17] Shao J, Wei H, Wang Q, Mei H. A runtime model based monitoring approach for cloud//Proceedings of the 3rd IEEE International Conference on Cloud Computing. Miami, USA, 2010: 313-320

[18] Rak M, Venticinque S, Txobhr M, et al. Cloud application monitoring: The mOSAIC approach//Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science. Athens, Greece, 2011: 758-763

[19] Calero J M A, Aguado J G. MonPaaS: An adaptive monitoring platform as a service for cloud computing infrastructures and services. IEEE Transactions on Services Computing, 2015, 8(1): 65-78

[20] Kashyap S, Ramamirtham J, Rastogi R, Shukla P. Efficient constraint monitoring using adaptive thresholds//Proceedings of the 24th IEEE International Conference on Data Engineering. Cancún, México, 2008: 526-535

[21] Dilman M, Raz D. Efficient reactive monitoring//Proceedings of the 20th Annual Joint Conference on Computer Communications. Anchorage, USA, 2001: 1012-1019

[22] Sharfman I, Schuster A, Keren D. A geometric approach to monitoring threshold functions over distributed data streams//Proceedings of the ACM SIGMOD International Conference on Management of Data. Chicago, USA, 2006

[23] Estan C, Varghese G. New directions in traffic measurement and accounting. ACM SIGCOMM Computer Communications Review, 2002, 32(4): 323-336

[24] Meng S, Liu L. Enhanced monitoring-as-a-service for effective cloud management. IEEE Transactions on Computers, 2013, 62(9): 1705-1720

[25] Fu X, Ren R, Zhan J, et al. LogMaster: Mining event correlations in logs of large-scale cluster systems//Proceedings of the 31st IEEE Symposium on Reliable Distributed Systems. Irvine, USA, 2012: 71-80

[26] Guan Q, Smith D, Fu S. Anomaly detection in large-scale coalition clusters for dependability assurance//Proceedings of the International Conference on High Performance Computing. Goa, India, 2010: 1-10

[27] Salfner F, Troger P, Tschirpke S. Cross-core event monitoring for processor failure prediction//Proceedings of the International Conference on High Performance Computing & Simulation. Leipzig, Germany, 2009: 67-73

[28] Oliner A J, Kulkarni A V, Aiken A. Using correlated surprise to infer shared influence//Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks. Chicago, USA, 2010: 191-200

[29] Lan Z, Zheng Z, Li Y. Toward automated anomaly identification in large-scale systems. IEEE Transactions on Parallel and Distributed Systems, 2010, 21(2): 174-187

[30] Xu W, Huang L, Fox A, et al. Detecting large-scale system problems by mining console logs//Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles. Big Sky, USA, 2009: 117-132

WANG Tao, born in 1982, Ph. D., associate professor. His research interests include fault diagnosis, software reliability, and autonomic computing for cloud computing systems.

GU Ze-Yu, born in 1991, M. S. His research interest is distributed monitoring.

ZHANG Wen-Bo, born in 1976, Ph.D., professor, Ph. D. supervisor. His research interests include distributed computing, cloud computing and middleware.

XU Ji-Wei, born in 1985, Ph. D. His research interests include software engineering and network distributed computing.

WEI Jun, born in 1970, Ph. D., professor, Ph. D. supervisor. His research interests include service oriented computing, middleware, and software engineering.

ZHONG Hua, born in 1971, Ph. D., professor, Ph. D. supervisor. His research interests include software engineering and distributed computing.

Background

More and more applications are deployed on public cloud computing platforms to provide services. Due to the complexity, dynamism and openness of cloud computing, the development and deployment of services are prone to many types of faults. However, the faults inside services are usually triggered by complex factors in the deployment environment at runtime. This makes it difficult to reproduce the faults. Therefore, debugging and testing services cannot effectively eliminate the inevitable faults triggered in specific contexts.

Monitoring is essential to detect and diagnose faults for guaranteeing the reliability and performance of Internet-based services. Monitoring technologies for cloud computing systems have widely attracted the attention of industrial and academic communities in recent years. Monitoring systems in cloud computing collect metrics from multiple layers (e. g., network, hardware, virtual machine, operating system, middleware, and application) in heterogeneous nodes. However, the procedure of collecting, transmitting, storing and processing a large amount of monitoring data introduces significant overhead, which affects the system performance. The existing commercial monitoring systems (e. g., Amazon CloudWatch) and open source monitoring systems (e. g., Zabbix) only provide a fixed monitoring period. For example, they collect a monitoring data instance every minute. Moreover, the customers of cloud services always pay for the monitoring service provided by a cloud service provider according to the collected metrics and the frequency of collecting monitoring data. Amazon reported that the cost of monitoring a cloud application occupied 18% of the operation cost for customers. Customers can save the monitoring cost and reduce the performance overhead by reducing the monitored items and monitoring frequency. However, monitoring with fewer items and lower frequency decreases the volume of available monitoring data to detect and locate faults, and thus decreases the possibility and timeliness of detecting faults.

To address the above issue, this paper proposes an adaptive monitoring approach, which selects significant metrics and dynamically adjusts the monitoring period according to the fault probability. For monitored items, we select some significant metrics, which can represent other metrics and reflect the system status. For monitoring frequency, we increase the monitoring period to decrease the performance overhead in the normal situation. On the contrary, we decrease that to improve the timeliness and precision of fault detection by closely tracking the system when the system is with a high fault probability. Since the fault probability is low during the whole system operation, adjusting the monitoring period dynamically can reduce a large amount of monitoring overhead.

This work was supported in part by the National Natural Science Foundation of China under Project 61402450 and the Beijing Natural Science Foundation under Project 4154088.