Computer Science and Application 计算机科学与应用, 2020, 10(11), 2006-2016
Published Online November 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.1011212

Hans汉斯

面向云计算环境的动态故障检测与诊断方法

刘　涵，田春岐

同济大学计算机科学与技术系，上海
Email: onfnju@163.com, tianchunqi@tongji.edu.cn

收稿日期：2020年11月3日；录用日期：2020年11月18日；发布日期：2020年11月25日

摘　要

在现代云计算系统中，通常会有成百甚至上千个云服务器通过多层网络相互连接，在如此大规模的复杂系统中，发生故障是一件很常见的事情。主动性故障管理是表征系统行为并预测云环境下故障动态的一项关键技术。为了预测云环境下的故障情况，我们需要监视系统的执行情况并收集与运行状况相关的运行时性能数据。但是，在新部署或者托管的云系统中，这些数据通常是没有标签的，在这种情况下，基于监督学习的方法是不合适的。在本文中，我们提出了一种使用贝叶斯模型集成的无监督故障检测方法，它可以表征系统的正常运行状态并检测异常行为，由系统管理员验证异常之后，对这些数据打上标签。在我们搭建的Hadoop集群下运用此种方法的实验结果表明，我们的方法可以实现较高的真阳率和较低的错误率。

关键词

云计算，故障预测，贝叶斯模型集成，无监督学习

Dynamic Fault Detection and Diagnosis Method for Cloud Computing Environment

Han Liu, Chunqi Tian

Department of Computer Science and Engineering, Tongji University, Shanghai
Email: onfnju@163.com, tianchunqi@tongji.edu.cn

Received: Nov. 3rd, 2020; accepted: Nov. 18th, 2020; published: Nov. 25th, 2020

Abstract

In a modern cloud computing system, there are usually hundreds or even thousands of cloud servers connected to each other through a multi-layer network. In such a large-scale and complex system, failure is a very common thing. Proactive fault management is a key technology that cha-

文章引用：刘涵，田春岐．面向云计算环境的动态故障检测与诊断方法[J]．计算机科学与应用，2020，10(11): 2006-2016. DOI: 10.12677/csa.2020.1011212

racterizes system behavior and predicts fault dynamics in cloud environments. In order to predict the failure of the cloud environment, we need to monitor the execution of the system and collect runtime performance data related to the operating status. However, in newly deployed or hosted cloud systems, these data are usually unlabeled. Methods based on supervised learning are inappropriate in this case. In this paper, we propose an unsupervised fault detection method using an ensemble of Bayesian models, which can characterize the normal operating state of the system and detect abnormal behavior. After the system administrator verifies the abnormality, labeled data are available. The experimental results of using this method under our Hadoop cluster show that our method can be implemented to achieve a higher true positive rate and a lower error rate.

Keywords

Cloud Computing, Failure Prediction, Ensemble of Bayesian Models, Unsupervised Learning

1. 引言

当今时代，数据中心和云计算系统正在变得越来越复杂，同时由于不断地添加和删除系统组件、更改执行环境、频繁更新和升级，它们也在不断地动态变化着[1]。经典的可靠性理论和常规方法很少会考虑系统的实际运行状态，因此也无法动态地检测系统运行时的故障，这类方法通常仅适用于设计反映系统长期或者平均运行状态的模型[2]。随着云计算系统的复杂性和动态性不断增长，对云环境实行主动故障管理已经成为提高系统可靠性的有效方法，而故障检测方法则成为此类技术的关键[3]。故障检测方法利用系统的运行时状态和观察到的故障的历史信息来预测云环境下未来发生的故障，因此能够为资源分配、计算能力的重新配置和系统维护提供有价值的信息。

现有的大多数故障预测方法都是基于统计学习技术[4]。他们使用监督学习模型来估计各种性能特征上故障发生的频率。这些方法的基本假设都是已经对训练数据集进行了标记，即对于用于训练预测器的每个数据点，设计人员已经知道它对应的状态是正常执行状态还是故障状态[5]。但是，标记的数据在现实世界的云计算系统中并不总是可用，尤其是对于新部署或托管的系统。如何准确预测此类系统中的故障发生是一项不小的挑战。

在本文中，我们提出了使用贝叶斯子模型的方法来动态地预测云计算系统环境中的故障。它以无监督的学习方式工作，并处理未标记的数据集。当云服务器正常运行时，该模型估计收集的运行时性能数据的概率分布，在系统管理员验证了检测到的异常行为之后，将标签添加到这些数据点。通过使用此种方法，我们实现了主动式故障管理框架的原型，并在搭建的 Hadoop 集群系统上评估了其性能。实验结果表明，本文提出的方法可以准确预测故障动态。

本文的主要工作可以分为以下几个部分：

1. 搭建 Hadoop 集群来模拟分布式的云环境，并使用工具收集该环境下的性能数据；
2. 对收集到的数据进行清洗，并通过采用基于互信息和 PCA 的算法来降低维度和特征之间的冗余；
3. 使用基于贝叶斯子模型集合的无监督学习算法来训练分类器，得到性能数据的概率模型，基于此概率模型来判断数据点是否是故障数据；

本文第 2 节介绍云环境下故障检测的相关工作，第 3 节介绍本文的整体实验架构，包括搭建模拟环

境、采集性能数据、提取特征、训练分类器等，第 4 节展示了此框架在我们模拟环境下的实验结果，第 5 节为总结和未来展望。

2. 相关工作

近年来，云平台故障诊断技术在工业与学术界均引起了广泛关注，在云计算系统上的故障检测也有了丰富的研究成果[6]。目前国内外关于云环境故障检测方法主要包括基于心跳策略的检测方法和基于性能数据的检测方法。

基于心跳策略的检测方法主要通过心跳包检测方式，通过对云平台分布式系统不同节点间心跳通信检测来保证系统的稳定性与可用性，云平台属于分布式系统，其故障检测框架大致分为广播型检测框架和层次型检测框架[7]。节点间通信模型一般基于以上两种框架，目前应用比较广泛的是动态心跳策略，其核心思想是根据节点间心跳通信的过程动态设置心跳超时限制Δ，当心跳包间隔时间超出Δ，则认为节点或者进程宕机。文献[8]提出了侦测故障的 QoS 度量标准，给出了无需统计的高度动态算法，动态估算超时限制Δ，修改心跳消息的发送周期，以适应分布式系统中动态变化的节点状况和网络情况。

动态心跳策略适用于绝大部分云计算网络，它能够主动调节检测进程间心跳发送周期来动态适应云计算环境下负载多变的网络状态[9]。但是这种方法通常采用统计学方法，比如回归等方式来实现动态心跳规划[10]，同时算法需要基于比较大的样本空间，无法处理当样本数据不遵守特定概率特征时的情况[11]。此外，心跳策略的决定还需要各个节点的样本特征，每个节点都保存样本特征会导致一定程度的资源浪费，同时不同集群下不同的特征分布可能会误导动态心跳策略，导致心跳策略不合理[12]。

基于性能数据的检测方式是通过对云计算运行时产生的性能数据(如 CPU、IO、内存、硬盘等)进行错误类别检测[13]。基于性能数据的故障检测方法是目前应用最为广泛的方法，主要分为机器学习方法、统计学方法等。文献[14]提出了一种基于模糊 KNN 算法的云计算故障数据检测方法，它首先利用密度聚类的方法对训练集进行预处理，然后采取模糊熵和互信息相结合的故障特征加权方法，将模糊 KNN 和分层检测与云计算特征加权结合进行样本训练，最后使用基于最大隶属度的自我学习进行结果检测。文献[15]提出了一种云环境下大规模数据特征的降维算法，它采用基于 FastICA 的检测样本优化的方法，利用加权相关系数来改进牛顿迭代算法，降低了信息的丢失和时间的消耗。同时引入模拟退火算法来改进隐马尔可夫模型，进而提高了故障的检测率。文献[16]中，ChiragN.Modil 提出一种将贝叶斯和 Snort 结合的故障检测系统，Snort 负责收集云计算平台下的数据，利用贝叶斯分类器做未知攻击检测。该方法有效降低了误检测率，同时能够检测到未知故障。

基于云计算性能数据的分析方法同样也存在一些问题，比如在训练样本更新或者有新的未知错误出现时[17]。这是由于云环境极其复杂，又经常遭受不同程度的攻击，错误数据类别一直会更新，云环境下的硬件环境也经常会有更新的情况[18]，但是很多监督学习算法需要人为判定后再进行训练，这就会导致云计算环境下错误样本集可能存在长期未更新的情况[19]。

3. 云换境下动态故障检测实验架构

本章节设计实验来提高云计算平台系统的可靠性。实验部分分为真实环境模拟、数据采集、数据处理、以及贝叶斯分类器训练四个模块。

3.1. 真实环境模拟

由于贝叶斯 EM 算法属于无监督学习算法，且在本文计算方法中用于计算数据点属于正常生成数据的概率，因此能否生成合理、全面的数据集是实验的首要工作。为了模拟真实运行的情况，本文采取虚

拟机(ubuntu)的形式搭建了基于 Hadoop 的云平台来采集实验数据，搭建的 Hadoop 集群下有 3 个节点，包含一个 Master 节点和两个 Slave 节点，三个节点的具体配置完全相同。虚拟表配置如表 1 所示(注：虚拟内存不设置限制以满足较大数据的计算需求)。

Table 1. Node configuration
表 1. 节点配置

项目	配置
CPU	Intel Core i7 2.2 GHz
内存	4096 MB
硬盘	30 GB
操作系统	Ubuntu 18.0

我们采用 Hibench 大数据基准测试套件来测试该集群的运行情况，该套件可用于测试 Hadoop 环境下的运行速度、吞吐量和资源利用率，我们使用其来模拟真实环境下程序的运行状况。其主要工作负载包括排序、词频计数、TearSort、重分区等。表 2 中详细列出了本次实验使用的 workload 模版以及其主要功能。

Table 2. System resulting data of standard experiment
表 2. 标准试验系统结果数据

测试项目	主要功能
Sort	使用 RandomTextWriter 生成随机数据，然后该功能负责将这些数据进行排序
WordCount	使用 RandomTextWriter 生成随机数据，然后该功能负责统计其中出现的词频数目
TeraSort	使用 Hadoop TeraGen 程序生成数据，然后执行 TeraSort 算法对数据进行排序
Enhanced DFSIO	同时执行写入程序和读取程序，测试 Hadoop 平台吞吐量
Bayesian Classification	使用 zipfian 分布的文档进行贝叶斯分类器训练
K-means clustering	使用 GenKMeansDataset 生成基于高斯分布的输入数据，进行 K-means 聚类
Logistic Regression	输入数据使用基于随机平衡决策树的 Logistic Regression Data Generator 生成，执行线性回归工作
Principal Components Analysis	输入数据通过 PCADataGenerator 生成，执行主成分分析工作
Support Vector Machine	输入数据通过 SVMDataGenerator 生成，之后执行支持向量机分类工作
PageRank	输入数据通过爬取网页数据后，执行 PageRank 算法
NWeight	执行 Nweight 算法
SQL Scan, Join & Aggreagte	数据通过爬取网页数据，执行过程基于文献[17]

列表中的操作覆盖了大量的计算机经典算法，涉及到 CPU、磁盘、网络、内存等重要计算机资源，为了模拟真实环境下的运行结果，我们执行三个数量级下的操作(以 WordCount 为例)：Small (3,200,000/bytes)级别、Large (320,000,000/byte)级别、Huge(3,200,000,000 ca/byte)级别。

3.2. 实验数据采集

实验数据采集模块主要负责采集云计算平台性能数据(CPU、IO、内存、网络等)。本文采用 Sysstat 进行数据收集，每分钟采集 82 个性能指标的数据，Sysstat 是监测系统性能和效率的一组工具，如 CPU 占比、网络吞吐数据等。其主要工具和作用如表 3 所示：

Table 3. Sysstat tooltable
表 3. Sysstat 工具表

工具	具体作用
iostat	监视输入输出，输出磁盘吞吐效率的数据和 CPU 使用率
mpstat	提供一个 cpu 或若干 cpu 相关数据
pidstat	关于 runtime 的进程/任务、CPU、内存等信息
sar	工具负责报告、采集和存储系统信息
sa1	负责记录、存储系统动态数据并存储到二进制的文件中。通过 cron 来运行，它是为 sadc 专门设计的前端程序
sa2	负责整理系统活跃信息，将信息总结后写入报告
sadc	收集系统动态数据，存储到二进制文件中

经 Sysstat 收集但未经处理的数据指标共 250 个，其中主要指标的意义如表 4 所示：

Table 4. Indicators for collecting data
表 4. 采集数据指标

采集类型	采集指标	指标描述
CPU	usr	用户处理级别下的 CPU 利用率
	system	系统级别下 CPU 利用率
	idle	CPU 空闲时间百分比
	iowait	当存在未完成的磁盘 I/O 请求时，描述 CPU 或 CPU 处于空闲状态的时间
IO	PROC/s	每秒创建的进程总数
	Tps	传输给物理磁盘的请求
	Bread/s	设备中读取的总数据量
	Wtps	设备中写入的总数据量
	Rtps	每秒发送的读请求总数
	Memused	内存使用百分比
	Bufpg/s	每秒系统用作缓冲区的额外内存页数。负值表示系统用作缓冲区的页面数减少
Mem	Campg/s	每秒由系统缓存的额外内存页的数目。负值表示缓存中的页面更少
	KBMEMFREE	空闲内存(单位为千字节)
	KBMEMFUSED	占据内存(单位为千字节)
	KBBUFFERS	内核用作缓冲区的内存量(单位为千字节)
	KBCHACED	内核用作缓存的内存量(单位为千字节)
	KBSWAPFREE	空闲交换空间(单位为千字节)
	KBSWAPUSED	占据的交换空间(单位为千字节)
	KBSWPCAD	缓存的交换内存(单位为千字节)的数量，这是曾经被换出、被换回但仍然在交换区域中的内存(如果需要内存，则不需要再次交换，因为它已经在交换区域中了)
Network	%swapused	使用交换空间的百分比
	%memused	使用内存的百分比
	PxPCK/s	每秒接收的包数
	RxBYT/s	每秒接收的字节数
	TxBYT/s	每秒传输的字节数
	RxCMP/s	每秒接收的压缩包数(用于 cslip 等)
	Coll/s	传输数据包时每秒发生的冲突数
	TCPSCK	目前正在使用的 TCP sockets 数量
	RAWSCK	目前正在使用的 RAW sockets 数量
	RxFIFO/s	每秒传输的包数

3.3. 数据预处理

大型数据中心和云系统可能有数百甚至数千个特征，它可以是系统或用户级别的 CPU 利用率、CPU 空闲时间、内存利用率、I/O 操作等等。大量的性能指标通常会使数据模型变得极其复杂，因此我们需要提取数据特征来减少数据之间的相关性、降低数据之间的冗余并移除无用的数据。考虑到程序的执行时间，在非监督学习下处理数据冗余以及相关性的方法我们采取了互信息和PCA (Principal components analysis)的方法。

3.3.1. 互信息(Mutual Information)

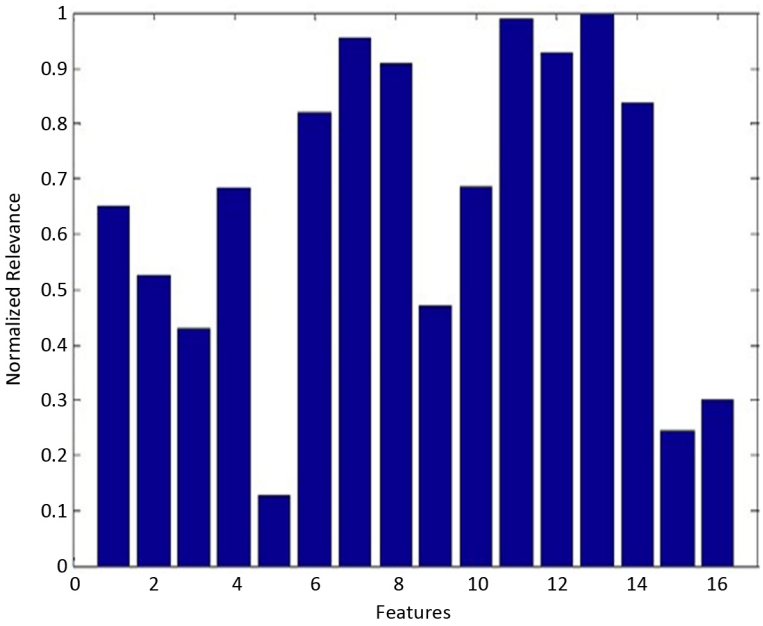

Figure 1. Mutual Information of CPU and Memory Related Features
图 1. CPU 和内存的特征对之间的互信息

云环境下的故障检测通常会采集很多不同类型的性能数据，这就会导致特征的维度很高。而在无监督学习的情况下，高维度的特征会导致特征彼此之间的区别性降低，同时高度相似的特征也会阻碍无监督学习中分组和表征数据的性能，我们可以通过使用互信息来评估特征之间的相似性。在下面的讨论中，我们用 $X_1, X_2, \cdots, X_n$ 来表示不同特征的离散随机变量，那么两个特征的相互信息定义如下：

$$I\left(X_i, X_j\right) = \sum_{x_i} \sum_{x_j} p\left(x_i, x_j\right) \log \frac{p\left(x_i, x_j\right)}{p\left(x_i\right) p\left(x_j\right)} \tag{1}$$

它量化特征 X_i 和特征 X_j 之间共享的信息量。$I\left(X_i, X_j\right)$ 可以从全局上度量两个特征之间的相关性。具有高相关性的特征对之间具有大量的相互信息，如果 X_i 和 X_j 是独立的，则这两个特征之间的互信息取到了最小值 0；如果 X_i 和 X_j 是同一特征，则它们之间的互信息取到了最大值 1。使用互信息的目的是减少所选特征子集之间的相关性，因此当一个特征和其它特征之间的互信息值很大的时候，就应该考虑将该特征从特征集中去除。在本文中，我们将相关性评估的指标定义如下：

$$Index\left(X_i\right) = \sum_{j=1}^{i-1} I\left(X_i, X_j\right) + \sum_{j=i+1}^{n} I\left(X_i, X_j\right) \tag{2}$$

可以证明，具有线性关系的两个特征之间具有较高的 *Index* 值，这两个特征之间表现出明显的相关性。图 1 中列出了 CPU 统计类别中每对特征根据上述公式计算出的互信息值，根据图中互信息的大小，最终

我们挑选出%user、%system、%idle、kbbbuffer、pgpgout/s、fault/s、rxpck/s、txpck/s 和 tps 这八个特征值。

3.3.2. PCA (Principle Component Analysis)

在对特征进行选择时，仅仅从特征集中挑选出相互独立的特征子集是远远不够的，特征之间的冗余会带来很多问题。主成分分析方法是应用最为广泛的降维算法之一。它的主要思想是将 n 维特征映射到 k 维上，映射出来的 k 维特征相互正交，我们将其称之为主成分。本文中我们使用特征值分解协方差矩阵实现 PCA 算法，假设特征集合 $X=\{x_1,x_2,x_3,\cdots,x_n\}$ 需要降维到 k 维，其算法流程如下所示：

1.对所有的样本进行中心化，即对每个样本数据执行 $x_i=x_i-\frac{1}{m}\sum_{j=1}^{m}x_j$ 操作；

2.计算样本的协方差数据 $C=\frac{1}{M}XX^{\mathrm{T}}$；

3.求出协方差矩阵的特征值及对应的特征向量；

4.将特征向量按对应特征值大小从上到下按行排列成矩阵，并取前k行组成矩阵P；

5.令 $Y=PX$ ，则Y即为降维后的数据；

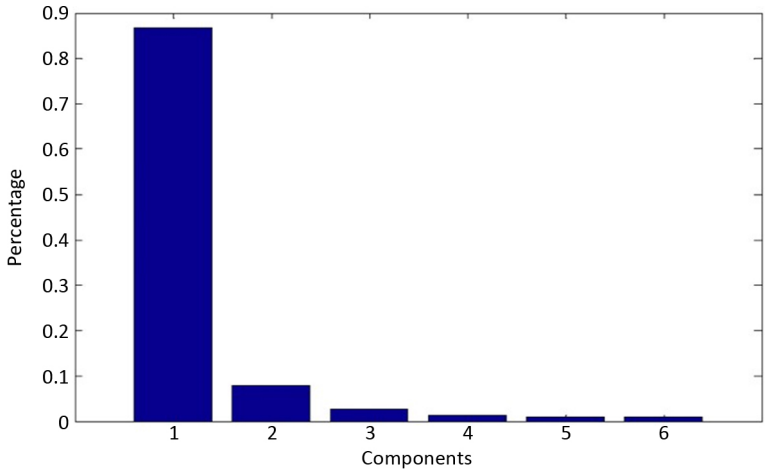

Figure 2. Redundancy Deduction by Principle Component Analysis
图 2. 使用 PCA 算法进行降维

然后我们用 PCA 算法来减少上述 8 个特征之间的冗余，图 2 为使用 PCA 算法对八个特征进行降维的效果，我们可以看到，降维后前两个主成分贡献了原数据 94.37%的信息量，因此我们可以进一步将维度减少到二维。

3.4. 分类器训练模块

为了检测可能的故障，我们分析采集的性能数据并基于它构建统计模型。对于一个经过降维处理的数据点 d，我们为其选择一个概率模型 f，它以数据点 d 作为输入，并输出该数据点为健康数据的概率，其中 f 的参数以无监督学习的方式从训练数据中学习。当设置阀值为 t 时，对于任一数据点 d_i，若 $f(d_i)<t$，则认为 d_i 属于故障数据或不常出现于正常数据中。这种构建模型的思想本质上是认为当一个数据点出现在健康数据中的概率较低时，此数据点应该被定义为错误数据或者异常数据。

3.4.1. 贝叶斯子模型集合算法推导

为了构建此概率模型并确保较高的检测精度，我们采用贝叶斯子模型集合算法来处理这类多态分布问题。每个子模型均为非参数数据模型，我们也没有假设单个参数的概率密度，而是根据其在训练数据

中的频率计数来确定其概率分布，每个子模型都有一个预估的先验概率 $p(m)$，其中 m 是指每个子模型的索引值，则数据点 d 的概率可以视为：

$$p(d)=\sum_{m\in submodels} p(d\mid m)*p(m) \tag{3}$$

其中 $p(d\mid m)$ 为子模型 m 生成数据点 d 的概率。

由于子模型概率分布是通过统计训练数据在子模型中的分布计数而来，由此我们可以得到子模型的概率计算公式如下：

$$p(m)=\frac{count(m)}{\sum_{m\in submodels} count(m)} \tag{4}$$

其中，$count(m)$ 代表满足条件的数据点的数量，其数学表达式为：

$$count(m)=\sum_{d\in trainset} p(m\mid d) \tag{5}$$

概率 $p(m\mid d)$ 代表数据点 d 分配给子模型 m 的概率。式(3)中右半部分中 $p(m)$ 可以通过式(4)计算得出，下面我们推导 $p(d\mid m)$ 的数学表达式。在之前的性能数据处理模块中，我们通过数据预处理和特征提取得到了数据点 d 相互独立且其分布属于离散区间的 k 个特征，因此可以得出 $p(d\mid m)$ 的数学表达式：

$$p(d\mid m)=\prod_{i=1}^{k} p(d_i\mid m) \tag{6}$$

由于我们使用的是贝叶斯离散模型，可知 d_i 值属于有限范围内，假设 d_i 值为 v，则 $p(d_i\mid m)$ 的计算方式如式(7)所示：

$$p(d_i=v\mid m)=\frac{count(d_i=v \text{ and } m)}{count(m)} \tag{7}$$

其中 $count(d_i=v \text{ and } m)$ 可以由下面的公式计算得出：

$$count(d_i=v \text{ and } m)=\sum_{d\in trainset} p(m\mid d)*I(d_i,v) \tag{8}$$

在这里 $I(d_i,v)$ 的定义如下所示：

$$I(d_i,v)=\begin{cases}1, & d_i=v\\ 0, & d_i\neq v\end{cases} \tag{9}$$

3.4.2. 贝叶斯子模型集合算法步骤

为了训练该模型，我们首先需要先决定子模型的数量，受到数据处理模块中特征选择的启发，我们选择了四个子模块。

贝叶斯子模型集合算法的具体步骤如下：

1.初始化参数数据：随机分配数据点给子模型，分配过程中初始化 $p(m)$ 与 $p(d_i\mid m)$；

2.E 步：确定子模型数据的条件概率，也就是 $p(m)$ 和 $p(d_i\mid m)$，然后再根据式(6)计算得到 $p(d_i\mid m)$，再根据公式(3)计算出 $p(d)$。这样 E 步就能够计算出每个数据点的概率 $p(d)$，从而进一步由式(10)计算出子模型 m 生成数据点 d 的概率，即 $p(m\mid d)$。

$$p(m\mid d)=p(d\mid m)*\frac{p(m)}{p(d)} \tag{10}$$

其中 $p(d\mid m)$、$p(m)$、$p(d)$ 均已计算出。

3.M 步：E 步完成之后，M 步依照 $p(m\mid d)$ 得到数据的分布情况，并根据公式(7)与式(4)更新 $p(d_i\mid m)$ 与 $p(m)$。这样即遵照了期望值重新计算了分布情况，又使期望值最大化。

4.迭代：反复执行 E、M 步直到 $p(m|d)$ 收敛。

迭代完成后，对于任意需要验证是否为错误数据的数据点 d，假设其 k 个特征值为：

$$d=(d_1=v_1,d_2=v_2,d_3=v_3,\cdots,d_k=v_k) \tag{11}$$

则由正常数据生成的概率 $f(d)$ 的计算公式如下所示：

$$f(d)=\sum_{m\in submodels} p(m)*\prod_{i=1}^{k} p(d_i=v_i,m) \tag{12}$$

假设阈值为t，则数据点d是否为健康数据就可以由下式(3.16)判断：

$$d=\begin{cases}\text{error} & \text{if } f(d)<t\\ \text{healthy} & \text{if } f(d)\geq t\end{cases} \tag{13}$$

4. 实验结果分析

本文中我们采用 TPR (True Positive Rate)、TNR (True Negative Rate)和 CDR (Correct Detection Rate)来验证模型的精度。TPR 表示模型验证结果准确且样本为健康样本的比例，TNR 为模型验证结果准确且样本为故障样本的比例，CDR 则是两者的均值。三者的数学表达式如下所示：

$$\text{TNR}=\frac{\text{TN}}{\text{TN}+\text{FP}} \tag{14}$$

$$\text{TPR}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{15}$$

$$\text{CDR}=\frac{\text{TNRR}+\text{TP}}{2} \tag{16}$$

由于本文中 TPR 和 TNR 的值都受到阈值的影响，因此我们最大化 CDR 的值作为目标设定阈值。实验中的 TNR、TPR、CDR 值，阈值和训练时间如图 3 所示：

```
threshold:  2476976491.6206493
TPR:  0.8637532133676092
TNR:  0.74
CDR:  0.8018766066838046
Time: 2587s
```

Figure 3. The value of TPR, TNR and CDR, threshold and training time
图 3. TNR、TPR、CDR 值，阀值和训练时间

由图中的实验结果可以得知，该分类器能够在一定程度上起到鉴别错误数据的能力，可以通过一段时间内性能数据的报错率来得知云环境中是否出现了故障，当样本采样选择在 100 份中有超过 30 份错误数据时，此时能准确识别出故障数据的概率 TNR 为 83%，而误识别正确数据为错误数据的率 FNR 仅为 7%。

5. 总结与未来展望

本文提出了一种面向云计算环境下的动态故障检测与诊断方法，通过搭建 Hadoop 分布式集群来模拟实验所需的云环境，然后通过使用 Sysstat 工具来收集服务器运行时的性能数据，每分钟记录 83 个性能指标的值。这些性能指标涵盖了云服务器上各个组件的统计信息，包括 CPU 使用率、进程创建、任务活动切换、内存和交换空间利用率、分页和页面错误、中断、网络活动、I/O 和数据传输、电源管理等等。

收集的数据被推送到 Hadoop 集群中的主服务器上，以进行系统范围内的运行状况分析。

在收集到性能数据后，我们首先对这些数据进行了清洗，对于数据集中存在着缺失值的某些特征项，我们使用其相邻的两个特征值的平均值进行填充，然后将数据解析并转换为统一格式。清洗完数据之后，对数据集进行特征选择，基于互信息(mutual Information)的特征选择算法能够选择出捕获大多数信息的独立特征，最终我们挑选出八个互相独立的特征。接着我们使用 PCA 算法来减少这 8 个特征值之间的冗余，实验结果显示第一和第二主成分占原始数据集变量的 94.37%以上，因此我们可以进一步将特征维度降至到二维。最后我们使用贝叶斯 EM 算法训练分类器，以无监督的学习方式检测异常行为，实现了对云环境下故障的动态检测和识别。实验结果表明，我们的方法可以实现较高的真阳率(TPR)和较低的错误率。

在本文的工作中，我们使用贝叶斯 EM 算法进行故障检测和预测，来实现云环境下的动态故障管理。但同时，我们也注意到即使使用最先进的算法，故障的预测准确率也无法达到 100%。在未来的工作中，我们打算探索其它先进的故障检测方法，比如利用反应性故障管理技术(如检查点和冗余执行)来处理错误预测。作为未来的工作，我们将整合这两种故障管理方法来进一步增强云的可靠性。

参考文献

[1] CNCERT. 2018 年中国互联网网络安全报告[R]. 国家计算机网络应急技术处理协调中心, 2009.

[2] 祝小康. 云存储系统故障自动化处理关键技术研究[D]: [硕士学位论文]. 南京: 南京理工大学, 2013.

[3] 蔡京平, 贾云得. 一种分布计算系统自适应故障侦测方法[J]. 小型微型计算机系统, 2007(1): 136-139.

[4] Modi, C., Patel, D., Borisaniya, B., *et al*. (2013) A Survey of Intrusion Detection Techniques in Cloud. *Journal of Network and Computer Applications*, **36**, 42-57. https://doi.org/10.1016/j.jnca.2012.05.003

[5] Wang, C., Talwar, V., Schwan, K., *et al*. (2010) Online Detection of Utility Cloud Anomalies Using Metric Distributions. 2010 *IEEE Network Operations and Management Symposium NOMS*, Osaka, 19-23 April 2010, 96-103.

[6] Chen, H., Jiang, G., Yoshihira, K. and Saxena, A. (2010) Invariants Based Failure Diagnosis in Distributed Computing Systems. *Proceeding of* 29*th IEEE Symposium on Reliable Distributed Systems*, New Delhi, India, 31 October-3 November 2010, 160-166. https://doi.org/10.1109/SRDS.2010.26

[7] Chen, Z., Xu, G., Mahalingam, V., *et al*. (2016) A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures. *Big Data Research*, **3**, 10-23. https://doi.org/10.1016/j.bdr.2015.11.002

[8] Bertier, M., Marin, O. and Sens, P. (2002) Implementation and Performance Evaluation of an Adaptable Failure Detector. *International Conference on IEEE*, Washington, DC, 354-363.

[9] Yan, Y., Dague, P., Pencolé, Y., *et al*. (2009) A Model-Based Approach for Diagnosing Fault in Web Service Processes. *International Journal of Web Services Research* (*IJWSR*), **6**, 87-110. https://doi.org/10.4018/jwsr.2009092205

[10] Samak, T., Gunter, D., Goode, M., *et al*. (2011) Online Fault and Anomaly Detection for Large-Scale Scientific Workflows. 2011 *IEEE International Conference on High Performance Computing and Communications*, Banff, 2-4 September 2011, 373-381. https://doi.org/10.1109/HPCC.2011.55

[11] Chenetal, M.Y. (2004) Path-Based Failure and Evolution Management. *Proc.* 1st *Symp. Netw. Syst. Design Implement*, Berkeley, CA, 23-36.

[12] Guo, L., Ma, Y., Cukic, B., *et al*. (2004) Robust Prediction of Fault-Proneness by Random Forests. 15*th International Symposium on Software Reliability Engineering*, Bretagne, 417-428.

[13] Suresh, Y., Kumar, L. and Rath, S.K. (2014) Statistical and Machine Learning Methods for Software Fault Prediction Using CK Metric Suite: A Comparative Analysis. *International Scholarly Research Notices*, **2014**, Article ID: 251083. https://doi.org/10.1155/2014/251083

[14] 刘诚诚. 云环境下故障检测方法研究与实现[D]: [硕士学位论文]. 昆明: 昆明理工大学, 2018.

[15] 张亚丹. 云计算平台故障检测关键技术研究[D]: [硕士学位论文]. 北京: 北京交通大学, 2016.

[16] Modi, C.N., Patel, D.R. and Patel, A. (2012) Bayesian Classifier and Snort Based Network Intrusion Detection System in Cloud Computing. 2012 *Third International Conference on Computing, Communication and Networking Technologies*, Coimbatore, 1-7. https://doi.org/10.1109/ICCCNT.2012.6396086

[17] Bala, A. and Chana, I. (2012) Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing. *In-*

ternational Journal of Computer Science Issues (*IJCSI*), **9**, 288.

[18] Wang, T., Zhang, W., Wei, J., *et al*. (2015) Fault Detection for Cloud Computing Systems with Correlation Analysis. 2015 *IFIP/IEEE International Symposium on Integrated Network Management*, Ottawa, 11-15 May 2015, 652-658. https://doi.org/10.1109/INM.2015.7140351

[19] 冯刚. 面向云计算平台的虚拟机故障注入工具研究与设计[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2013.