基于改进 Eclat 算法的资源池节点异常模式挖掘*

高 强[1]，张凤荔[1†]，陈学勤[1]，王馨云[1]，耿贞伟[2]，周 帆[1]
（1. 电子科技大学 信息与软件工程学院，成都 610054；2. 云南电网有限责任公司 信息中心，昆明 650217）

摘 要：云计算环境中，资源池节点异常模式挖掘对于快速诊断节点状态具有重要作用。针对云环境下计算资源池、存储资源池、网络资源池节点数据特征，对资源池节点状态信息进行预处理，利用关联规则算法挖掘资源池节点参数状态信息之间的关联关系，如高位—高位和低位—高位模式等。提出了一种基于限制属性连接并具有垂直数据格式的关联规则算法 i-Eclat 算法。i-Eclat 算法通过转换资源池节点状态数据格式、建立非频繁 2-项集以减少连接次数，并构建信息存储结构体来限制冗余属性连接。实验表明，所提出的方法可以有效发现云计算资源池节点之间的隐藏关系；同时，i-Eclat 比经典算法计算性能更优，特别是针对较大数据集的处理。

关键词：模式异常挖掘；关联规则；资源池；i-Eclat 算法；云计算
中图分类号：TP391 **文献标志码**：A **文章编号**：1001-3695(2018)02-0333-06
doi:10.3969/j.issn.1001-3695.2018.02.003

Mining anomaly pattern of nodes in resource pool based on improved Eclat algorithm

Gao Qiang[1], Zhang Fengli[1†], Chen Xueqin[1], Wang Xinyun[1], Geng Zhenwei[2], Zhou Fan[1]
(1. *School of Information & Software Engineering, University of Electronic Science & Technology of China, Chengdu* 610054, *China*; 2. *Information Center, Yunnan Power Grid Co., Ltd., Kunming* 650217, *China*)

Abstract: In cloud computing, mining anomaly pattern of nodes plays an important role in promptly diagnosing node states of resource pool. According to the data characters of computing resource, storage resource and network resource, it preprocessed the information of node status. And it used association rule algorithm to mine the rules among the nodes' parameter status of resource pool, e.g., "high level-high level" and "low level-high level". This paper proposed an improved Eclat algorithm, called i-Eclat, which based on vertical data format of restrictive attribute connection. i-Eclat reduced the number of connections by transforming the format of state data and constructing a non-frequent 2-itemsets, as well as it constructed the storage structure to limit the redundant attribute connections. Extensive experiments have been conducted to demonstrate that this method can find the hiding rules among resource pool's nodes. And it also shows that i-Eclat outperforms traditional methods on computing efficiency, especially on processing big data sets.

Key words: anomaly pattern mining; association rule; resource pool; i-Eclat algorithm; cloud computing

0 引言

云计算技术经过十多年的发展，成为研究人员研究的热门领域之一[1]。云计算是以虚拟化技术为基础，通过虚拟化计算服务器集群、存储服务器集群和以软件定义网络(SDN)技术为支撑的中高端交换机集群来提供高性能计算服务、海量信息存储服务和不同应用系统下的网络虚拟化带宽资源服务[2]。同时云计算技术也催生了大数据时代的诞生，人们越来越重视保存历史数据来挖掘其中的潜藏价值[3]。云环境下的虚拟化资源池节点(如虚拟主机)以一定的时钟周期产生大量状态信息，通过分析其中异常状态信息历史数据，可以找出其中潜在的关联信息或模式。例如在某项分布式业务系统处理中，其采用多台虚拟机提供计算服务，而一台虚拟机的状态异常可能导致相关虚拟机状态异常；同样在某一段时间内，中端资源池虚拟主机中存在多个虚拟主机 CPU 处于低位运行，而一个虚拟主机处于高位运行，那么可以建议云计算管理者迁移此虚拟主机的业务系统至高端资源池虚拟主机或转移部分业务系统到其他中端资源池虚拟主机以减少虚拟主机负荷。

1993 年，Agrawal 等人[4]提出关联规则概念，最初是用来解决购物篮数据分析。目前关联规则已经应用到各个行业，利用关联规则算法挖掘云环境下资源池节点状态信息关联关系可以很好地扩展关联规则应用场景，对于处理资源池节点状态信息具有重要作用。

关联规则算法 Apriori[5]是基于水平数据格式来生成候选集，通过支持度统计筛选候选集形成频繁集，再通过置信度阈值产生规则。Han 等人[6]针对 Apriori 算法产生大量候选集的

收稿日期：2016-10-18；**修回日期**：2016-12-01 **基金项目**：国家自然科学基金资助项目(61602097,61272527)；四川省科技支撑计划资助项目(2016GZ0065,2016GZ0063)；中央高校基本科研业务费资助项目(ZYGX2015J072)

作者简介：高强(1993-)，男，安徽合肥人，博士研究生，主要研究方向为机器学习、深度学习；张凤荔(1963-)，女(通信作者)，教授，博导，博士，主要研究方向为信息安全、机器学习(fzhang@uestc.edu.cn)；陈学勤(1993-)，男，硕士研究生，主要研究方向为机器学习；王馨云(1993-)，女，硕士研究生，主要研究方向为机器学习；耿贞伟(1973-)，男，工程师，硕士，主要研究方向为电力信息系统；周帆(1981-)，男，副教授，博士，主要研究方向为机器学习、深度学习.

问题提出了 FP-Growth 算法,同样采用水平数据格式。ZaKi[7]提出的基于垂直数据格式的关联规则挖掘算法即 Eclat 算法,其计算性能一般优于 Apriori 等水平格式数据,但算法也存在消耗大量存储空间的问题,文献[8 ~ 10]对 Eclat 算法提出了相应的改进策略。

目前已有学者将关联规则用于异常模式的挖掘。例如,文献[11]采用一种基于模糊理论的关联规则算法,根据故障预测中聚类区域边缘数据,通过基于规则的阈值迭代算法求解日志数据预处理修正系数,从而建立主机故障预测模型,并且提高了算法效率,但是其提出的关联模型只是针对单个虚拟主机参数状态进行关联。文献[12]提出了一种结合遗传网络编程(GNP)的模糊类关联规则挖掘方法,用于检测网络入侵。文献[13]将负关联规则和独立性的卡方检验应用于数据,通过数据之间的独立性来检测 XML 文档中的异常。

对于云计算平台的异常诊断,传统的方法是通过收集日志数据,建立其在正常情况下的轮廓,然后通过比较现在的状态和正常状态来分析系统情况。文献[14]采用贝叶斯算法对云平台 IaaS(infrastructure as a service)层节点状态分类从而进行异常检测。文献[15]用无监督的 SOM 神经网络对虚拟云环境中的节点进行聚类分析并通过各指标的贡献值进行异常定位。文献[16,17]针对云计算环境的在线异常检测,提出了一种基于熵的异常测试方法 EbAT,并利用熵表示系统异常度量的分布。

本文通过基于 VMware vSphere 虚拟化计算资源池、存储服务虚拟化资源池和网络交换机虚拟化资源池的云计算模拟平台进行状态信息数据收集,将收集到的原始数据进行数据预处理与格式转换,通过关联规则算法挖掘节点之间的关联关系。本文主要贡献如下:

a)将基于垂直格式的关联规则算法应用于云环境下资源池异常模式挖掘中。

b)改进传统的 Eclat 算法,提出了 i-Eclat 算法,建立非频繁 2-项集以减少连接次数、构建信息存储结构体限制冗余属性连接。

c)挖掘两种关联关系模式,即高位节点—高位节点模式和低位节点—高位节点模式,深层次挖掘其中的隐藏关系,对于云计算环境中资源的动态平衡具有重要作用。

1 云环境分析

1.1 云环境构建分析

在虚拟化技术支撑下,云计算更加强调资源池的概念,即将计算资源、存储资源、网络资源整合成资源池的形式,针对不同的业务需求和配置要求,按需分配。

以 VMware vSphere 为代表的虚拟化软件为资源池化和资源池划分提供了基础,通过弹性化构建低端资源池、中端资源池和高端资源池等划分方式,从而形成了计算资源池、存储资源池和网络资源池。图 1 展示的是一般资源池结构化的划分;图 2 是通过读取 VMware vSphere 控制中心 vCenter 后台数据库还原的部分资源池结构拓扑图。

1.2 数据采集

资源池异常模式挖掘需要通过多种方式的数据采集来形成原始数据集,对于计算服务器、存储服务器、网络交换机需要通过建立主动式监测探针,如通过部署了虚拟化软件主机操作系统 SDK(software development kit)平台获取性能日志信息、利用 SDN(software defined network)技术对网络流量进行采集、通过存储虚拟化控制工具采集存储性能和日志,再结合现有应用性能分析系统 API(application programming interface)开发数据源探针,实现对资源池内的虚拟资源(计算资源、存储资源、网络资源),以及承载虚拟资源的主机设备、网络设备等对象的信息远程探针,并对其进行存储,其框架如图 3 所示。

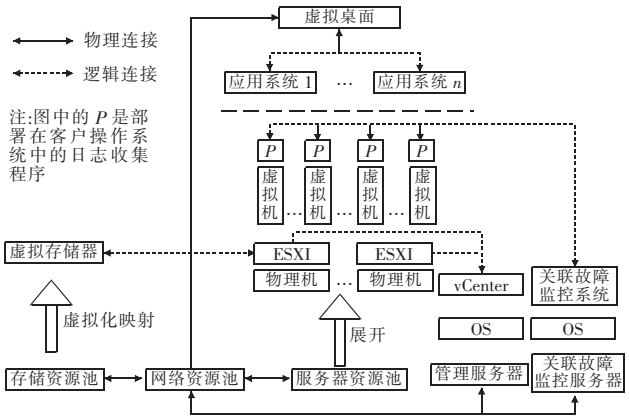


图 1 云环境资源池

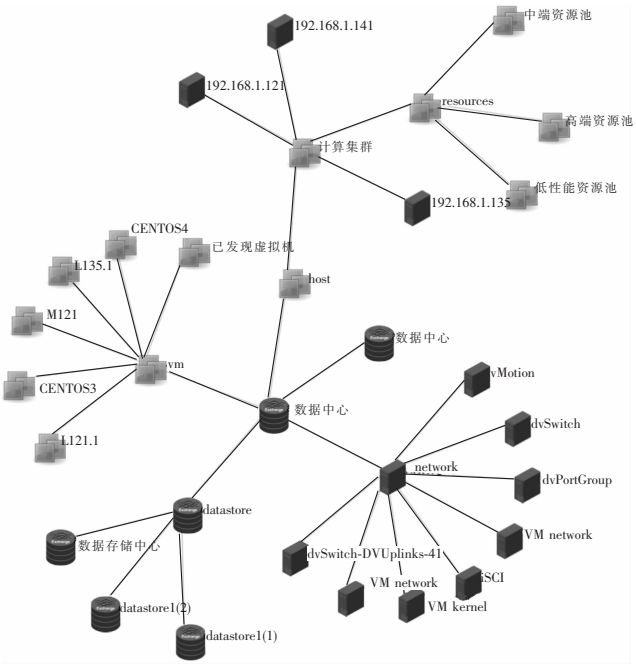


图 2 云计算中心部分拓扑结构

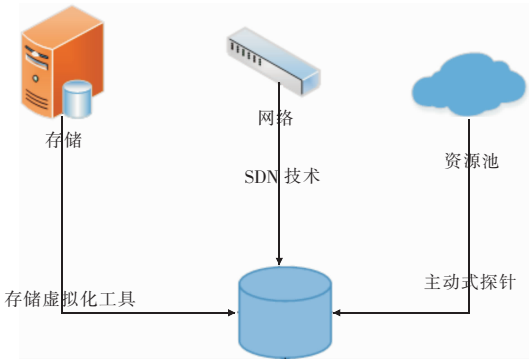


图 3 云环境资源池信息获取探针

2　基本原理及模型定义

2.1　关联规则

定义 1　关联规则是具有 $X \rightarrow Y$ 形式的蕴涵关系,X 和 Y 在关联规则中分别称之为先导和后继。$X \rightarrow Y$ 表示如果 X 成立那么 Y 在满足一定条件下也成立,其中 X 和 Y 为项目集合,一般用支持度和置信度作为条件成立的评价标准。

定义 2　支持度 $\text{support}(X \rightarrow Y)$ 表示在事务数据库中包含 $X \cup Y$ 项目集的事务占全体数据库事务的百分比,即 $\text{support}(X \cup Y) = \text{support}(X \rightarrow Y) = P(X \cup Y)$,其中 $X \cup Y$ 表示一条事务中同时包含 X 和 Y。

定义 3　置信度 $\text{confidence}(X \rightarrow Y)$ 表示事务数据库中包含 $X \cup Y$ 的项目集事务数目占包含 X 的项目集事务数目的比重。即

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

2.2　定理与模型定义

2.2.1　基本定理

定理 1　非频繁项集的超集一定是非频繁项集。

定理 2　任何频繁项集的非空子集也是频繁项集。

2.2.2　模型参数定义

不同于传统的购物篮事务数据库分析,本文事务数据库存储数据有计算服务器上的虚拟主机参数信息,包括 CPU 占用率、内存使用率、虚拟网卡端口速率等;存储服务器参数信息包括 CPU 占用率、内存使用率、控制器状态、LUN 状态等;交换机参数信息包括 CPU 占用率、内存使用率、各端口速率等。按照设备长期运行情况划分,将各项数值型数据状态信息转换为三种状态信息,即低位、正常、高位。其中,低位表示 CPU 利用率较低或者网络端口速率低等;正常表示节点状态运行一般情况下产生的占用率或端口速率等;而高位表示 CPU 占用率高或端口速率高等状态。

由于各物理设备或虚拟主机运行情况不同,并且不同的物理设备的参数也不相同,所以首先需要确定各个节点状态信息的模式评判标准。本文通过统计各个节点状态信息在历史数据中的低位运行状态、正常运行状态和高位运行状态比重进行模式划分,实现所有的状态信息转换成可处理数据模型。

定义 4　对于云计算环境 $P = \{\text{IP}_1, \text{IP}_2, \text{IP}_3, \cdots, \text{IP}_n\}$,$P$ 表示云计算环境中资源池中所有节点集合。P 中节点包括虚拟主机、网络设备、存储服务器等。本文统一用其 IP 地址作为其唯一标志符。数据集合 $D = \{P^1, P^2, P^3, \cdots, P^t\}$,$t$ 表示时间戳,表示数据集收集了 t 个时间戳节点状态信息集合,$t \in [1, T]$。

定义 5　$\text{IP}_i = \{s_1, s_2, s_3, \cdots, s_m\}$,$i \in [1, n]$,其中 s_j 表示节点中存在的参数种类,如 CPU 状态、内存状态、虚拟网卡端口状态等,其中 $j \in [1, m]$。

定义 6　定义 $s_j \in V = \{\text{value}_1, \text{value}_2, \text{value}_3\}$,其中 value_1 表示低位状态,value_2 表示正常状态,value_3 表示高位状态。

定义 7　高位—高位模式表示在云计算环境中一个处于高位运行的节点集合会导致另外一个高位运行节点集合的产生。在真实的生产环境中即一个节点的异常会导致另外一个节点的异常,如分布式业务系统,往往一台虚拟主机的异常会导致其他虚拟主机状态异常,这种模式反映了虚拟主机由于内在的关联,一台虚拟主机的异常往往会产生连锁反应,导致其他虚拟主机异常。

定义 8　低位—高位模式表示在云计算环境中一个处于低位运行的节点集合会导致另外一个高位运行节点集合的产生。在真实的生产环境中,如一台物理主机中存在多台虚拟主机,其中多台虚拟主机一直处于空闲状态,而一台虚拟主机处于高位运行,这种模式反映了需要将其虚拟主机上的业务系统转移到其他虚拟主机上以减轻虚拟主机的运行压力。

3　i-Eclat 算法描述

3.1　模型描述

本文的算法由预处理模块、模型构造模块和模式挖掘模块三部分组成。其框架如图 4 所示。

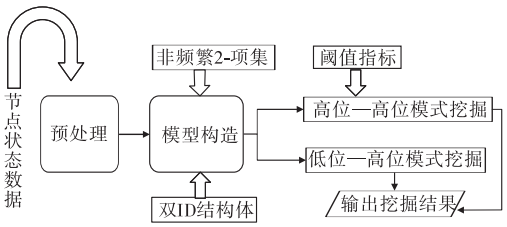


图4　算法的整体框架

a)原始数据按照时间序列存储了每一台设备的状态数据,而本文处理的数据对象是垂直格式,因此通过预处理模块,将原始水平格式状态数据转换成垂直格式数据。

b)模型构造模块主要构造模型挖掘信息存储结构体,以减少冗余连接产生,并且结构体保存数据原始信息,利于挖掘结果的产生。同时构造非频繁 2-项集,对关联规则挖掘项目集连接进行提前剪枝,以减少非频繁集合的比较运算。

c)模型挖掘模块主要分为高位—高位模式挖掘与低位—高位模式挖掘,通过不断的算法迭代,挖掘所有符合条件的具有关联关系的集合。

3.2　预处理

3.2.1　数值型数据转换

原始资源池节点状态数据为数值型数据,同时数据参数不一,例如 CPU 为占用率,而网卡端口状态参数为流量速率。

对于云计算环境 P,需要将数值型数据转换为状态位的数据 S,其中 $S = \{\text{value}_1, \text{value}_2, \text{value}_3\}$。同时不同节点之间即使状态信息标志相同,但是由于受其本身业务环境的不同,数值代表的意义也不相同。假设 IP_1、IP_2 同时表示虚拟主机,存在相同状态集合 $\{s_1, s_2, s_3, s_4\}$,虽然状态类别表示相同,但是在真实生产环境中其运行业务不同,持续运行状态信息数值也不相同。

因此在数据处理中数值型数据转换步骤如下:

a)查找在 t 个时间戳中的节点各个状态的最大值 $\max\_v_j$ 和最小值 $\min\_v_j$,其中 $j \in [1, m]$。

b)统计节点的各个状态在 t 个时间戳的数值分布。

c)根据数据分布情况,按照一定规则将数值型数据切割转换成 $V = \{\text{value}_1, \text{value}_2, \text{value}_3\}$ 三种状态集合。

3.2.2　数据格式转换

Eclat 算法不同于诸如 Apriori 算法的数据格式,其采用垂

直数据格式,减少计算资源消耗,提升规则提取速度。转换过程如图 5 所示。

时间	节点	状态类型
	IP_1	$s_1,s_2,\cdots,s_m$
	IP_2	$s_1,s_2,\cdots,s_m$
	…	…
	IP_n	$s_1,s_2,\cdots,s_m$
2	…	…
…	…	…
t	…	…

转换

IP\时间	1	2	…	t
IP_1	s_1	s_1	…	s_1
	s_2	s_2	…	s_2
	…	…	…	…
	s_m	s_m	…	s_m
IP_2	…	…	…	…
…	…	…	…	…
IP_n	…	…	…	…

图 5 数据格式转换

3.3 模型构造

3.3.1 非频繁 2-项集

i-Eclat 算法采用非频繁 2-项集对项目集连接进行剪枝,保存非频繁 2-项集有助于减少不必要的非频繁项目集的比较运算,提升算法效率。本文使用双哈希链表实现非频繁 2-项集的存储,其结构如图 6 所示。其中 key_1 存储非频繁 2-项集第一项即第一个节点 IP,value_1 值为哈希链表用于存储非频繁 2-项集的第二个节点 IP,对于每个节点中的参数 id 则用列表的形式存储在 key_2 所对应的 value_2 中;建立 subindex 结构体存储参数,first_sub_id 表示第一个节点 IP 的某一参数,sec_sub_id 表示第二个节点 IP 的某一参数。

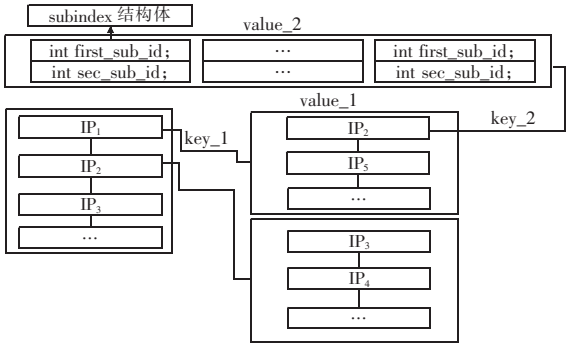


图 6 非频繁 2-项集

因此对于 3-项集及 3-项集以上的连接,则首先通过检测其是否在非频繁 2-项集中,如果存在,则不进行支持度统计,以减少数据运算次数。例如对于 A、B 集合的最后一项分别为〈IP_1,s_1〉和〈IP_2,s_2〉,判断其是否可连接的步骤如下:

a)通过第一层哈希链表定位至 key_1 = IP_1;

b)得到 value_1,并且定位到 key_2 = IP_2;

c)得到 value_2 即获得参数列表,判断 s_1 是否等于 first_sub_id,并且 s_2 是否等于 sec_sub_id,如果存在则不进行连接。

3.3.2 信息存储结构体

本文定义数据存储结构体如图 7 所示,在 ID 结构体中存储节点标志即 main_ID 和状态种类标志即 sub_ID;在 Node 结构体中定义 ID 集合,主要是为了算法迭代中存储连接的节点集合,同时 info 存储 t 时刻状态位的信息;在 CloudNode 结构体中同样定义节点标志位 BID 集合,此定义节点标志集合主要在数据连接时排除同一节点内的种类状态信息连接,以减少冗余连接信息。

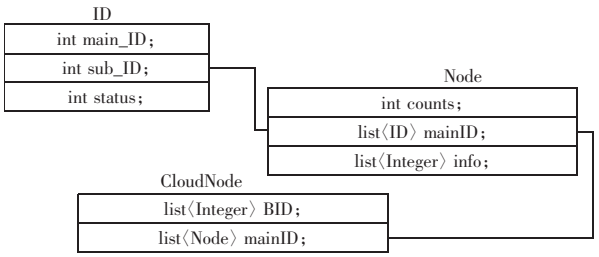


图 7 信息存储结构体

3.4 模式挖掘

3.4.1 高位—高位模式挖掘

高位—高位模式挖掘主要处理云计算环境中,节点某项参数的异常会导致其他节点某项参数的异常,在此主要比较在 t 时刻两节点某项状态是否同时为 $value_3$,如果状态相同则支持度增加。其核心步骤描述如下:

a)获取数据,创建新数据结构存储集合 C_u,其中 u 表示迭代次数。

b)获取节点集合 $A_u = \{C_{u-1}, IP_i\}$,$B_u = \{C^*_{u-1}, IP_j\}$,其中 $i < j$。

c)判断 A_u、B_u 是否具有共同前缀,即 C_{u-1}、C^*_{u-1}是否相等;如果不是则继续循环,否则进行 d)。

d)依次连接 A_u、B_u 节点内部参数,判断其是否在非频繁 2-项集内,如果在,则不进行连接和支持度统计;否则进行步骤 e)。

e)如果 A_u、B_u 参数集合中 t 时刻状态位都为 $value_3$,则支持度增加,$t \in [1,T]$。统计完后判断 A_u、B_u 参数集合连接后是否为频繁集,如果是,则同时加入存储集合 C_u,否则继续连接。

f)$u = u + 1$,返回至 a)。

3.4.2 低位—高位模式挖掘

为了更好地挖掘低位—高位模式,本文在获取 1-项频繁集时标记节点参数在 t 个时刻状态位出现次数最多的状态位代表连接的参数特征。例如 IP_1 中参数 s_1 状态位 $value_1$ 出现的次数最多,那么在下一次参数连接时,其他节点某项参数和 IP_1 中参数 s_1 连接,只需要比较参数 s_1 状态位 $value_1$。

3.5 伪代码实现

1)高位—高位模式伪代码

```
高位—高位模式
输入:
    原始数据集 O;
    支持度阈值 min_supp;
    循环结束标志位 flag = true。
输出:
    高位—高位模式 H_fre。
描述:
O = read_file( );//读取原始数据
D = value_convert(O);//数值型数据转换
H = convert_vertical(D);//转换为垂直格式数据
F = first_frequent(H,min_supp);//获取 1-项频繁集
  while(flag)
    for(i = 0;i < F. size( );i ++ )
      for(j = i + 1;j < data. size( );j ++ )
        if(differNum(F. get(i). BID,F. get(j). BID)
          if(F. get(i). BID. size( ) == 2)//产生非频繁 2-项集
            norelation_set = generate_norelation(F);
          end if
```

```
            foreach x ∈ F. get(i). info //第一项节点的参数
              foreach y ∈ F. get(j). info//第二项节点的参数
              new = combine(x,y); //连接参数
              if(new ∉ norelation_set &&
              ((tempnode = count_h_info(x,y)) > = min_supp))
                tempc = add(tempnode); //加到频繁参数集合
              end if
            end for
          end for
            new_F = add(tempc); //加入连接后的节点频繁集
        end if
      end for
    end for
    if(new_F == null)
      flag = false;
    F = new_F;
    H_fre = add(F);//存储每一层的频繁集
  answer = H_fre//获得所有模式
```

2)低位—高位模式伪代码

```
低位—高位模式
输入:
    原始数据集 O;
    支持度阈值 min_supp;
    循环结束标志位 flag = true。
输出:
    低位—高位模式 L_fre。
描述:
    O = read_file();//读取原始数据
    D = value_convert(O);//数值型数据转换
    H = convert_vertical(D);//转换为垂直格式数据
    //获取 1-项频繁集并且设置状态位
     for(i = 0;i < H. data. size();i ++)
      foreach x ∈ H. get(i). subinfo
       set_status(x);//设置状态位
       if((tempnode = count_first(x)) > = min_supp)
        F = add(tempnode)
      end if
     end for
  end for
  while(flag)
    for(i = 0;i < F. size();i ++)
      for(j = i + 1;j < data. size();j ++)
        if(differNum(F. get(i). BID,F. get(j). BID)
          if(F. get(i). BID. size() == 2)//产生非频繁 2-项集
            norelation_set = generate_norelation(F);
          end if
          foreach x ∈ F. get(i). subinfo
           foreach y ∈ F. get(j). subinfo
            new = combine(x,y); //连接参数
            if(new ∉ norelation_set &&
          ((tempnode = count_l_info(x,y)) > = min_supp))
            tempc = add(tempnode);
            end if
          end for
        end for
          new_F = add(tempc);
        end if
      end for
    end for
    if(new_F == null)
      flag = false;
    F = new_F;
    L_fre = add(F);
  answer = L_fre//获得所有模式
```

4　实验过程、结果与分析

4.1　实验环境

本文中所有实验均在表 1、2 所示的实验环境中完成。表 1 是数据处理实验环境,表 2 是构建云计算资源池部分设备信息。

表 1　数据处理实验环境

实验环境	环境配置
操作系统	Windows 7
CPU	AMD A8-7650K 3.30 GHz
内存	8 GB
编程语言	Java
编程环境	Eclipse

表 2　部分云计算模拟环境

实验环境	环境配置
虚拟化软件	VMware vSphere 6.0
原始数据存储数据库	PostgreSQL
可视化监控软件	Cacti + Nagios + Centreon
计算服务器	CPU 2 × Intel® Xeon® 内存 256 GB、SSD 2.4 TB

4.2　数据集预处理

实验所用的数据是从云计算资源池中取出的实际运行数据集,通过收集约六个月的资源池节点状态信息形成约 50 000 个时间戳原始数据集,其中计算服务器状态种类为 9 项,存储服务器种类状态数为 8 项,交换机状态由于端口数量不同平均状态种类为 15 项。如获取的虚拟主机节点参数分别为 CPU_usage、CPU_usagemhz、disk_usage、mem_usage、net_usage,其中部分环境数据如表 3 所示。

表 3　部分环境数据

entity	time	CPU_usage	CPU_usagemhz	disk_usage	mem_usage	net_usage
192.168.1.141	2016-9-7 10:00	2 633	3 477	25	5 747	152
192.168.1.121	2016-9-7 10:00	527	696	22	5 728	0
192.168.1.135	2016-9-7 10:00	7 382	1 765	42	6 535	92

通过收集到的数据,根据选择的属性画出了其中五台虚拟主机部分参数状态数据预处理前各个属性的分布,如图 8 ~ 12 所示。其中 entity 表示虚拟主机节点 ID,time 为部分数据收集时间戳,z 轴为节点指标数值。

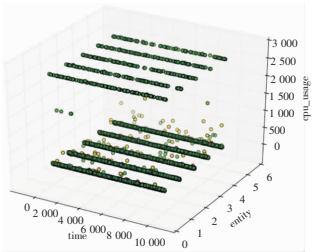


图 8　CPU_usage 分布

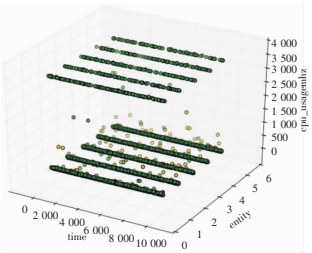


图 9　CPU_usagemhz 分布

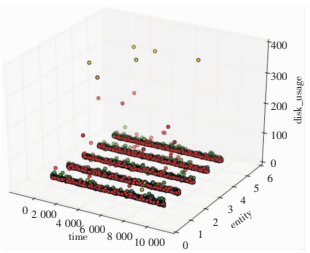


图 10　disk_usage 分布

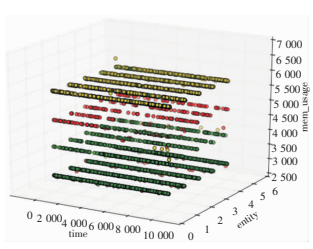


图 11　mem_usage 分布

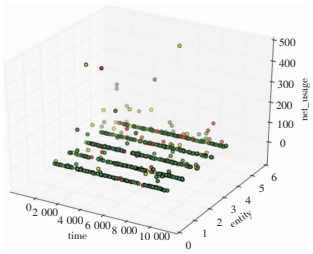
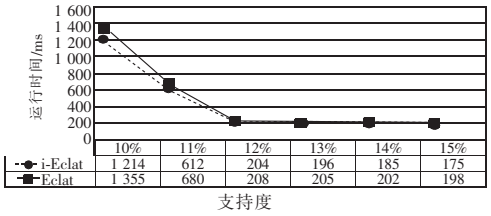


图 12 net_usage 分布

4.3 实验结果分析

4.3.1 支持度—运行时间

通过固定时间戳 T 为 10 000 个,将支持度阈值 min_supp 从 10% 上调至 15%,观察 i-Eclat 与普通 Eclat 程序运行时间(运行时间单位:ms),其对比结果如图 13 所示。通过对比发现,在相同的事务数下,i-Eclat 在支持度较小的情况下优于普通 Eclat 算法。

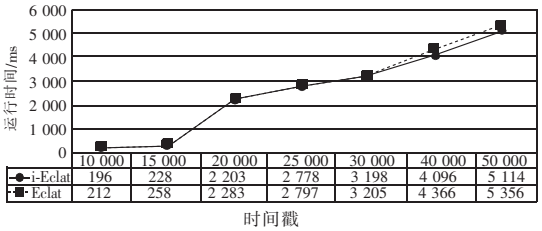


	10%	11%	12%	13%	14%	15%
i-Eclat	1 214	612	204	196	185	175
Eclat	1 355	680	208	205	202	198

图 13 支持度—运行时间

4.3.2 时间戳—运行时间

通过固定支持度 min_supp 为 20%,不断增加时间戳事务的个数,从 10 000 升至 50 000,对比 i-Eclat 与普通 Eclat 算法的运行时间,其对比结果如图 14 所示。在相同的支持度下,i-Eclat在数据集较大的情况下优于普通 Eclat 算法。



	10 000	15 000	20 000	25 000	30 000	40 000	50 000
i-Eclat	196	228	2 203	2 778	3 198	4 096	5 114
Eclat	212	258	2 283	2 797	3 205	4 366	5 356

图 14 时间戳—运行时间

4.3.3 支持度—规则数

通过固定时间戳 T 为 50 000,将支持度阈值 min_supp 从 2% 上调至 12%,程序在不同支持度下发现的规则个数分布如图 15 所示。

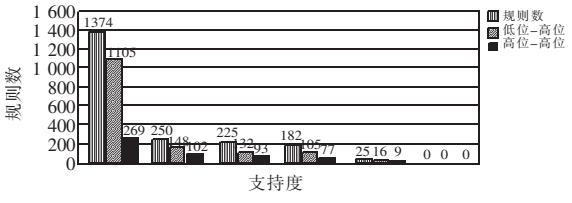


图 15 支持度—规则数

从图 15 可以发现高位—高位模式要少于低位—高位模式。通过结果分析发现,异常状态的出现与其存在的业务系统和使用方式有很大关系,在计算资源池中多个虚拟主机 CPU 的高位状态同时间段出现较为频繁,同时在交换机设备中某些端口持续出现高位状态而大多数处于低位状态,说明其端口信息交互频繁。

5 结束语

云计算资源池的节点异常模式挖掘对于快速诊断资源池异常和挖掘节点中参数之间的隐藏关系具有重要作用。本文提出了基于改进的 Eclat 关联规则算法处理资源池节点之间的关系,挖掘其中的高位—高位、低位—高位模式,并与传统 Eclat 算法进行比较。通过实验验证了算法的有效性和高效性。未来将通过资源池节点状态信息挖掘异常模式以提前预测节点异常,以及提供节点异常分类、节点定位等服务。

参考文献:

[1] 李乔,郑啸. 云计算研究现状综述[J]. 计算机科学,2011,38(4):32-37.

[2] 戴元顺. 云计算技术简述[J]. 信息通信技术,2010,4(2):29-35.

[3] Bahrami M,Singhal M. The role of cloud computing architecture in big data[M]//Information Granularity,Big Data,and Computational Intelligence. [S. l.]:Springer International Publishing,2015:275-295.

[4] Agrawal R,Imielinski T,Swami A. Mining associations between sets of items in massive databases[C]//Proc of ACM-SIGMOD International Conference on Management of Data. New York: ACM Press,1993:207-216.

[5] Agrawal R,Srikant R. Fast algorithms for mining association rules in large databases[C]//Proc of International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc,1994:487-499.

[6] Han Jiawei,Pei Jian,Yin Yiwen. Mining frequent patterns without candidate generation[J]. ACM Sigmod Record,2000,29(2):1-12.

[7] Zaki M J. Scalable algorithms for association mining[J]. IEEE Trans on Knowledge & Data Engineering,2000,12(3):372-390.

[8] 熊忠阳,陈培恩,张玉芳. 基于散列布尔矩阵的关联规则 Eclat 改进算法[J]. 计算机应用研究,2010,27(4):1323-1325.

[9] 冯培恩,刘屿,邱清盈,等. 提高 Eclat 算法效率的策略[J]. 浙江大学学报:工学版,2013,47(2):223-230.

[10] Yu Xiaomei,Wang Hong. Improvement of Eclat algorithm based on support in frequent itemset mining[J]. Journal of Computers,2014,9(9):2116-2123.

[11] 丁三军,薛宇,王朝霞,等. 基于模糊数据挖掘的虚拟环境主机故障预测[J]. 计算机工程,2015,41(11):202-206.

[12] Mabu S,Chen Ci,Lu Nannan,*et al*. An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming[J]. IEEE Trans on Systems Man & Cybernetics Part C,2011,41(1):130-139.

[13] Premalatha K,Natarajan A M. Chi-square test for anomaly detection in XML documents using negative association rules[J]. Computer & Information Science,2009,2(1):35-42.

[14] 任涛. 面向 IaaS 的虚拟机异常检测系统研究[D]. 重庆:重庆大学,2014.

[15] Dean D J,Nguyen H,Gu Xiaohui. UBL: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems[C]//Proc of International Conference on Autonomic Computing. 2012:191-200.

[16] Wang Chengwei,Talwar V,Schwan K,*et al*. Online detection of utility cloud anomalies using metric distributions[C]//Proc of IEEE Network Operations and Management Symposium. 2010:96-103.

[17] Wang Chengwei. EbAT: online methods for detecting utility cloud anomalies[C]//Proc of Middleware Doctoral Symposium. 2009:1-6.