



# 基于知识库的系统安全评估方法

邢宇恒

(国家电网公司信息通信分公司, 北京 100761)

**摘要:**在系统非正常状态和正常状态这两种运行状况下,使用 nmon 等系统监测工具采集了代表系统运行状态的指标数据,对获取的指标数据进行了向量化操作,构造了可以反映系统运行状态的多维行为特征。采用类内、类间评价距离准则对指标数据进行了有效特征提取,构建了能够反映系统运行状态的行为特征知识库。然后利用行为特征知识库的数据构造了决策树,以判断系统所处状态是否安全。

**关键词:**行为特征知识库;类内、类间距离评价准则;决策树

**中图分类号:**TP399

**文献标识码:**A

**doi:** 10.11959/j.issn.1000-0801.2016115

## System security assessment based on knowledge base

XING Yuheng

State Grid Information & Telecommunication Branch, Beijing 100761, China

**Abstract:** Under the two kinds of system operation conditions, abnormal state and normal state, indicated data on behalf of the system running status was collected by system monitoring tools such as nmon, and the obtained index data was quantitatively operated. The features of multidimensional behavior was constructed which could reflect the running state of the system. The index data has been carried on by effective feature extraction by means of the between-class and within-class distance measurement criterion, and knowledge base of behavior features was constructed which reflects the system running state. The decision tree was constructed to judge whether the system of state was secure through data features of behavior knowledge base.

**Key words:** knowledge base of behavior feature, between-class and within-class distance measurement criterion, decision tree

### 1 引言

操作系统作为计算机系统资源的管理者,需要为软硬件提供相应的服务和支持,因此需要确保操作系统在正常的状态下运行。但是当系统资源消耗到一定程度时,就会导致系统运行效率下降甚至因为某项系统资源瓶颈而导致系统不可用。系统在发生某类安全问题之前,可能已经处于一种非正常的状态了。而行为作为状态变化的外

在表现,从信息数据角度看,当对输入的数据进行处理并将其转化成输出数据后,其状态会发生一定的变化,故可以通过监测到的系统状态指标数据对当前系统状态进行反推。

本文主要使用 nmon 监测工具来获取系统运行状态指标数据。在系统产生安全问题(宕机或者进入死锁状态等)之前,利用反映系统非正常和正常状态的历史数据对系统当前所处的状态进行判断,从而有效地判断出系统状态并

及时通知系统管理员采取相应的解决措施。

本文采用决策树策略对系统所处状态是否正常进行判定,构建决策树需要以下步骤。

(1)利用内存泄露堆积可能引发系统状态不正常这一特点,设计相关测试用例模拟系统的非正常状态。

(2)利用 nmon 监测工具获取系统模拟情况下的非正常状态以及正常状态的指标数据,采用第 2 节的方法对获取的指标数据进行向量化操作,构造可以反映系统运行状态的多维行为特征;并对多维行为特征进行基于类内、类间距离评价准则的有效提取,进而选取若干指标数据以构建行为特征知识库。

(3)利用行为特征知识库中的信息作为训练样本集构建决策树,从而判断系统所处状态是否正常及安全。

## 2 基于类内、类间距离评价准则构建行为特征知识库

在众多操作系统的监测工具中,nmon 是被广泛使用的监测工具。相对于其他工具来说,nmon 工具所记录的信息是比较全面的。nmon 工具监控的项目主要有:CPU 占用率、内存使用情况、磁盘 I/O 速度、传输和读写比率、文件系统的使用率、网络 I/O 速度、传输和读写比率、错误统计率与传输分组的大小、消耗资源最多的进程、计算机详细信息和资源、页面空间和页面 I/O 速度、用户自定义的磁盘组和网络文件系统等。虽然通过工具监测可获取到代表系统运行状况的指标数据,但通过 nmon 工具获取的指标数据有多个,这些指标中包含大量对系统状态分析价值不高的元素,为了排除无关信息的干扰,避免数据冗余,需要对其特征值进行有效提取后再进行知识库的构建。

### 2.1 行为特征知识库的构建流程

行为特征知识库构建流程如图 1 所示,具体介绍如下。

(1)采集代表系统非正常状态以及正常状态下的指标数据。

(2)对获取的指标数据进行向量化操作,构造可以反映系统运行状态的多维行为特征。

(3)运用类内、类间距离评价准则进行有效行为特征提取。

### 2.2 有效特征提取

首先定义符号表示: $x_i^j=(x_{i1},x_{i2},\dots,x_{in})^T,i=1,2,\dots,m$ ,其中, $x_i^j$ 表示系统正常状态和非正常状态下采集的共  $m$  个不同时刻的指标数据集, $x_{ij},i=1,\dots,m,j=1,\dots,n$ 表示  $i$  时刻

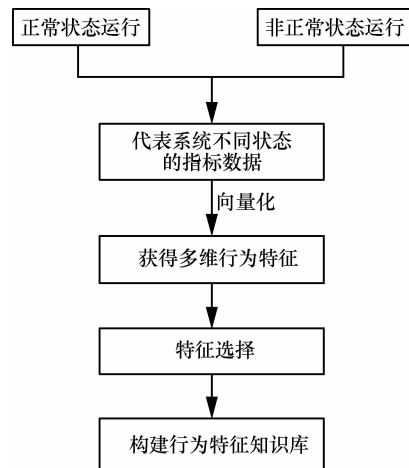


图 1 行为特征知识库构建流程

获得的第  $j$  个指标数据, $y_i \in \{0,1\}, i=1,2,\dots,m$  记录  $x_i$  在系统正常状态( $y_i=0$ )或非正常状态( $y_i=1$ )下的指标数据。 $y_i$  被称为类别标号,即将指标数据按照系统运行状态分为正常和非正常两类。

#### 2.2.1 类内距离

假设数据集  $\{x_1,\dots,x_m\}$  在某种相似性测度基础之上被划分为  $K$  类  $\{x_i^{(j)}\}, j=1,\dots,K$  为类别标号,  $i=1,\dots,n_j$  表示类内数据的序号,  $\sum_{j=1}^K n_j=m$ 。则类内距离定义为:

$$D_W = \sum_{j=1}^K \sum_{i=1}^{n_j} \|x_i^{(j)} - m_j\|^2 \quad (1)$$

其中, $m_j$  表示  $j$  类特征均值, $m_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^{(j)}$ 。 $D_W$  越小,说明特征越显著。

#### 2.2.2 类间距离

同理,类间距离定义为:

$$D_B = \sum_{j=0}^1 (m_j - m)'(m_j - m) \quad (2)$$

$$m_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^{(j)} \quad (3)$$

$$m = \frac{1}{m} \sum_{i=1}^m x_i \quad (4)$$

其中, $m_j$  表示  $j$  类特征均值, $m$  为总特征均值。 $D_B$  越大,说明特征越显著。

#### 2.2.3 类内类间距离准则

由上文的描述可知:若在类内距离越小的同时类间距离越大,则在区分特征方面效果越明显。定义可以同时反



映类内和类间距离的准则函数为:

$$D = \frac{D_B}{D_W} \rightarrow \max \quad (5)$$

类间距离  $D_B$  越大,类内距离  $D_W$  越小,即类内类间距离  $D$  越大,特征就越显著。针对本文所涉及的系统非正常与正常两类问题,设系统正常状态和非正常状态下某个指标  $T$  的分布函数如图 2 所示,并记系统正常状态下该特征的均值为  $m_0$ 、方差为  $\sigma_0$ ,系统非正常状态下该特征的均值为  $m_1$ 、方差为  $\sigma_1$ 。

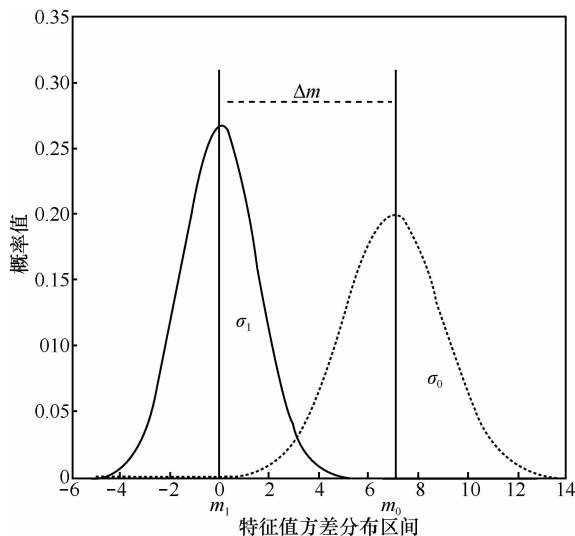


图 2 特征  $T$  的分布函数

此时,类内类间距离准则为:

$$D = \frac{D_B}{D_W} = \frac{(m_0 - m_1)^2}{\sigma_0^2 + \sigma_1^2} \quad (6)$$

由此可知,  $\Delta m = |m_0 - m_1|$  越大,对于两类的区分性就越大,即行为特征显著性就越好。这说明指标  $T$  对于区分系统正常状态和非正常状态的能力很强,应该将此指标作为有效特征加入行为特征知识库中。相反,若  $\Delta m = |m_0 - m_1|$  越小,则说明该指标将系统状态分类正确的概率比较小,可靠性比较低,这种特征可被丢弃,不再加入行为特征知识库。

### 3 决策树的构建

根据第 2 节系统监测取得的有效系统指标参数,例如 CPU 利用率、内存利用率、磁盘 I/O 速率、中断数、文件系统利用率,通过设计不同的测试用例来模拟系统资源不同的消耗状态,根据系统对一类固定操作的响应时间长短来判断系统是否处于正常状态。对监测得到的数据进行归类,例如将 CPU 的占用率分为高、中、低 3 个值。将有效系统指标归类后得到的系统状态参数见表 1。

在得到有效的特征知识库后,就可以利用这些指标数据构建决策树。目前决策树技术已经在许多数据挖掘系统中得到了应用,国内外很多公司均推出了自己的数据挖掘系统,例如 Microsoft、SGI、SAS、IBM、SAP 等在已推出的数据挖掘系统中首选的就是决策树方法。

而当前最具有影响力的决策树算法是 Quinlan 于 1993 年提出的 C4.5 算法。C4.5 选择信息增益值率最大的属性作为分裂属性,将训练实例集分裂成若干子集,在各子集上持续地递归上述过程,直到叶子节点为纯的实例集或者再无其他可供选择的分裂属性,然后使用剪枝技术修剪之前产生的树。

表 1 系统状态参数

| CPU 利用率 | 内存利用率 | 磁盘 I/O 速率 | 中断数 | 文件系统利用率 | 系统状态 |
|---------|-------|-----------|-----|---------|------|
| 高       | 高     | 高         | 少   | 高       | 不正常  |
| 高       | 高     | 高         | 多   | 中       | 不正常  |
| 中       | 高     | 高         | 少   | 中       | 正常   |
| 低       | 中     | 高         | 少   | 高       | 正常   |
| 低       | 低     | 中         | 少   | 中       | 正常   |
| 低       | 低     | 中         | 多   | 中       | 不正常  |
| 中       | 低     | 中         | 多   | 低       | 正常   |
| 高       | 中     | 高         | 少   | 低       | 不正常  |
| 高       | 低     | 中         | 少   | 低       | 正常   |
| 低       | 中     | 中         | 少   | 中       | 正常   |
| 高       | 中     | 中         | 多   | 高       | 正常   |
| 中       | 中     | 高         | 多   | 高       | 正常   |
| 中       | 高     | 中         | 少   | 中       | 正常   |
| 低       | 中     | 高         | 多   | 中       | 不正常  |

**定义 1** (信息熵) 设有一个离散随机变量  $X$ , 它有  $n$  个可能取值, 分别为  $a_1, a_2, \dots, a_n$ , 各种取值出现的概率分别为  $p_1=P(a_1), p_2=P(a_2), \dots, p_n=P(a_n)$ , 则:

$$\text{info}(p_1, p_2, \dots, p_n) = - \sum_{n=1}^N p_n \lg p_n \quad (7)$$

称为随机变量  $X$  的信息(离散)熵。

**定义 2** (信息增益) 一个属性的信息增益是指由于使用这个属性分割实例集而导致的期望熵的降低。其定义式为:

$$\text{gain}(S, A) = \text{info}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{info}(S_v) \quad (8)$$

其中,  $S$  为实例集合,  $A$  为某一属性,  $v$  为属性的某一个取值。

**定义 3** (分裂信息) 假设训练实例集  $S$  的某属性  $A$  的取值个数为  $k$ ,  $S_j$  为属性  $A$  的第  $j$  个可能取值对应的实例子集, 则  $S$  关于属性  $A$  的分裂信息量定义为:

$$\text{splitinfo}(S, A) = - \sum_{j=1}^k \left[ \frac{|S_j|}{|S|} \lg \left( \frac{|S_j|}{|S|} \right) \right] \quad (9)$$

**定义 4** (信息增益率) 信息增益率 = 信息增益/分裂信息量。

图 3 为一个计算属性 CPU 的信息增益率的例子。属性 CPU 有 3 个取值, 分别为高(high)、中(middle)、低(low)。当前实例集合  $S$  中共有 14 个实例, 其中 9 个实例的类值为正常, 5 个实例的类值为不正常。因此实例集合的信息熵为:

$$\text{info}(S) = - \left( \frac{9}{14} \times \lg \frac{9}{14} + \frac{5}{14} \times \lg \frac{5}{14} \right) = 0.940 \text{ bit} \quad (10)$$

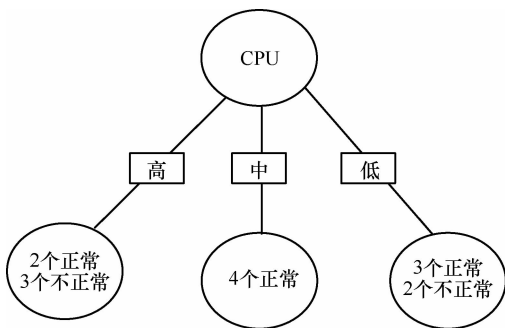


图 3 CPU 信息属性分裂

属性 CPU 为高的实例有 5 个, 其中 2 个类值为正常, 3 个类值为不正常。信息熵为:

$$\text{info}(v=\text{high}) = - \left( \frac{2}{5} \times \lg \frac{2}{5} + \frac{3}{5} \times \lg \frac{3}{5} \right) = 0.971 \text{ bit} \quad (11)$$

属性 CPU 为中的实例有 4 个, 且全部为正常。信息熵为:

$$\text{info}(v=\text{middle}) = - \left( \frac{4}{4} \times \lg \frac{4}{4} + 0 \right) = 0 \quad (12)$$

属性 CPU 为低的实例有 5 个, 其中 3 个类值为正常, 2 个类值为不正常。信息熵为:

$$\text{info}(v=\text{low}) = - \left( \frac{3}{5} \times \lg \frac{3}{5} + \frac{2}{5} \times \lg \frac{2}{5} \right) = 0.971 \text{ bit} \quad (13)$$

故属性 CPU 的信息增益为:

$$\text{gain}(S, \text{CPU}) = 0.940 - \left( \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 \right) = 0.246 \text{ bit} \quad (14)$$

选择 CPU 为分裂属性时, CPU 属性将实例集分裂成 3 个子集, 各子集中实例的个数分别为 5、4 和 5, 因此分裂信息量为:

$$\text{splitinfo}(S, \text{CPU}) = - \left( \frac{5}{14} \lg \frac{5}{14} + \frac{4}{14} \lg \frac{4}{14} + \frac{5}{14} \lg \frac{5}{14} \right) = 1.577 \quad (15)$$

因此, 选择 CPU 为分裂属性时, 其信息增益率为:

$$\text{gainRatio}(\text{CPU}) = \frac{0.246}{1.577} = 0.156 \quad (16)$$

重复以上步骤, 可以得到内存利用率、磁盘 I/O 速率、中断数等信息增益率。

使用 R 环境下 Rweka 分组提供的 J48 函数对本文第 2 节取得的有效系统状态指标来构建决策树, 得到的系统状态决策树如图 4 所示。

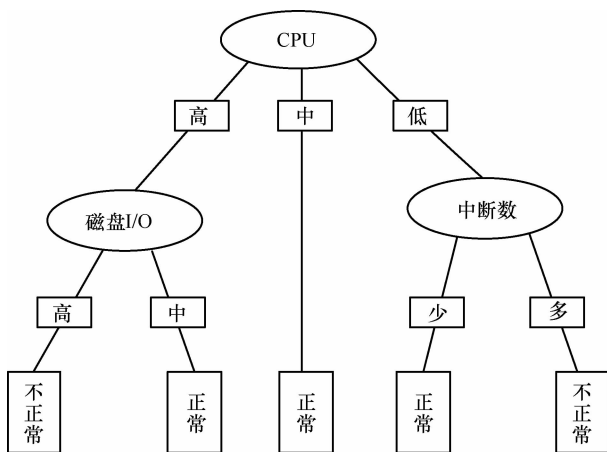


图 4 系统状态决策树

## 4 结束语

以往只是利用系统监测工具获取系统运行状态指标数据, 却没有利用这些数据对系统进行状态分析和判断。根据本文提出的思路进行评估系统状态的实验, 结果表明



本文的设计是切实可行的,具有研究价值。日常运维经验显示,信息系统在运行维护的前期,当用户压力还没有很大时,系统的性能表现还不错。当用户数越来越多或者某一个时间段内大量用户访问某一系统时,系统响应时间会变得很长甚至系统不可用。利用本文提出的方法,结合系统的压力测试,可以找出系统的资源瓶颈。在系统运行监控时,也能够对系统状态做出预判,从而避免系统出现故障。目前在电力信息系统运行维护的过程中,虽然可以做到对所有的系统进行监控,但是在发生告警通知时,有一定的误警发生,浪费了人力物力,利用本文提出的决策树方法,在运维过程中收集系统运行状态指标,构建行为特征知识库,最终构建出决策树并找出告警的根本原因(指标)如何找到发生告警通知的根本原因受到硬件和实验条件的影响,没能取得大规模的实验数据。如果能把更多的系统状态指标添加到决策树中,会使该方法对系统状态做出的判断更加精确,判断的准确率更高。

## 参考文献:

- [1] 王利. 浅谈 Linux 系统安全及应用[J]. 科技信息, 2010(10): 240-241.  
WANG L. Safety and application of Linux system[J]. Science & Technology Information, 2010(10): 240-241.
- [2] 杨善红. Linux 应用程序内存错误自动化测试研究[J]. 民营科技, 2011(10):53.  
YANG S H. Research on Linux applications automatic memory error testing[J]. Private technology, 2011(10):53.
- [3] 唐川. UNIX 系统性能监控简述及 shell 系统资源统计程序[J]. 科学咨询, 2009(7): 39-40.  
TANG C. The UNIX system performance monitoring and shell system resource statistics program[J]. Scientific Consult, 2009(7): 39-40.
- [4] 刘海燕. Linux 系统的安全检测与增强技术分析[J]. 计算机工  
程与设计, 2005, 26(1): 100-102.
- LIU H Y. Linux system safety inspection and reinforcing technology analysis[J]. Computer Engineering and Design, 2005, 26(1): 100-102.
- [5] 周超. Linux 下 C 语言程序内存泄漏的研究 [J]. 工矿自动化, 2008(4): 137-139.  
ZHOU C. Study on C language memory leak under Linux[J]. Industry and Mine Automation, 2008(4): 137-139.
- [6] HSU C W, LIN C J. A comparison of methods for multiclass support vector machines[J]. IEEE Transactions on Neural Networks, 2002, 13(2): 415-425.
- [7] The R project for statistical computing [EB/OL]. [2015-01-22]. <http://www.r-project.org/>.
- [8] 孙文华. 基于 B/S 架构的 nmon 远程监控平台[J]. 信息系统工程, 2012(6): 16-17.  
SUN W H. Based on B/S architecture nmon remote monitoring platform[J]. Information System Engineering, 2012(6):16-17.
- [9] 薛利民. 基于 Linux 的电网状态监测系统软件[J]. 电力自动化设备, 2004, 24(7): 126-127.  
XUE L M. The software of power grid monitoring system based on the Linux [J]. Electric Power Automation Equipment, 2004, 24(7): 126-127.
- [10] BENNETT A A. A behavior-based approach to adaptive feature detection and following with autonomous underwater vehicles[J]. IEEE Journal of Oceanic Engineering, 2001, 25(2): 218-220.

## [作者简介]



邢宇恒(1988-),男,国家电网公司信息通信分公司 ERP 系统运维工程师,主要研究方向为信息系统性能监测及安全。