

使用混合模型预测 Web 服务器中的资源消耗

闫永权 郭 平

(北京理工大学计算机学院 北京 100081)

摘 要 软件老化是一种在长期运行的软件系统中观察到的软件异常状态,如性能下降、暂停服务,甚至失效等现象。软件抗衰技术被用来处理软件老化带来的问题:停止软件应用、移除错误的因素、重新启动应用。对于软件老化和抗衰来说,如何准确地预测老化系统中的资源消耗并且找到一个合适的时机执行抗衰是一个关键的问题。针对该问题,提出一种混合模型方法用于资源消耗预测,并且提出多门限值时间段抗衰算法用于抗衰时机的选择。实验结果表明,混合模型方法在资源消耗预测上要优于其他模型,并且提出的抗衰算法要优于单一的门限值算法。

关键词 软件老化,软件抗衰,失效,混合模型

中图法分类号 TP301

文献标识码 A

DOI 10.11896/j.issn.1002-137X.2016.10.008

Predicting Resource Consumption in Web Server Using Hybrid Model

YAN Yong-quan GUO Ping

(School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

Abstract Software aging is a phenomenon that the state of software degrades and leads to performance degradation, hang/crash failures or both in a long running software application. Software rejuvenation, which involves occasionally stopping the software application, removing the cumulative error factors and then rebooting the application in a clean environment, is used to counteract software aging problems. For software aging and rejuvenation problems, it is a key problem that how to accurately forecast the resource consumption of aging system and find a proper timing to execute rejuvenation. In this paper, a methodology of hybrid model was proposed for resource consumption forecast and a rejuvenation algorithm of time slot in multi-threshold values was given. The experiment results show that the hybrid model is superior to other models in resource consumption forecasting of an IIS Web server and the proposed rejuvenation algorithm is better than the single threshold algorithm.

Keywords Software aging, Software rejuvenation, Failure, Hybrid model

1 引言

一些研究表明^[1,2]软件系统的性能下降和突然失效等现象是由软件老化问题引起的。软件老化现象指长期运行的软件系统中出现的状态异常、性能下降、系统失效等现象。软件老化通常表现为:资源泄露,如内存泄露;文件句柄和套接字未释放;进程、线程未结束;碎片问题;数值累计错误;数据损坏、文件系统和数据库文件的损坏等。为了处理由于软件老化引起的一系列问题,Huang 等人^[1]提出了软件抗衰技术:通过偶尔或者周期性地清除老化状态,使得软件运行环境恢复到鲁棒状态。

尽管通过抗衰操作可以消除软件老化的影响,然而对一个正运行的软件系统执行抗衰操作会导致抗衰操作期间直接和间接的损失。因此为了减少由于抗衰操作引起的老化问题,需要提前找到一个合适的时机执行抗衰操作,而这往往取决于对系统状态的准确预测。

Magalhaes 等人^[3]使用 Holt-Winters 模型预测资源消耗。Li 等人^[4]使用一个神经网络的变体检测运行的软件系

统的老化现象。Hoffmann 等人^[5]使用线性回归、支持向量回归等方法来预测一个电信系统中的资源消耗。Araujo 等人^[6]使用 4 个时间序列模型预测一个云平台的资源消耗。尽管很多的学者使用时间序列方法或机器学习方法预测遭受老化影响系统中的资源消耗情况,然而其所使用的方法要么是线性方法要么是非线性方法。

自回归累积移动平均模型(Autoregressive Integrated Moving Average Model, ARIMA)在过去几十年是最流行的线性模型之一,常常被用于预测外汇交易、社会和经济领域中的问题。ARIMA 模型在使用时有如下的假设:时间序列是平稳的并且没有数据丢失,时间序列的将来值是现在值和过去值的线性关系,残差错误是一个白噪声。因而如果时间序列中的数据是非线性关系,则使用 ARIM 对序列进行拟合是不合适的。

对于非线性问题来说,人工神经网络(Artificial Neural Network, ANN)被认为是一个很好的用于处理时间序列的非线性问题的建模方法。在过去的几十年里,人工神经网络模型被应用于时间序列问题。当一个时间序列中过去、现在值

到稿日期:2015-08-16 返修日期:2015-12-07 本文受国家自然科学基金项目(61375045)资助。

闫永权(1980—),男,博士生,讲师,CCF 会员,主要研究方向为可信计算,E-mail: yongquanyan@aliyun.com;郭 平(1957—),男,博士,教授,主要研究方向为计算智能及其在模式识别、图像处理、天文大数据、软件可靠性工程等方面的应用。

得相互影响的关系很难被表示时,使用人工神经网络建立数学关系是合适的。当时间序列中存在多重共线性或者孤立点时,人工神经网络往往要好于线性模型。当然如果收集的数据整体上呈现线性特征,则人工神经网络^[7]的表现比线性模型的表现要差。ARIMA 和人工神经网络成功应用于各自的线性和非线性领域,然而使用人工神经网络对线性问题进行建模以及使用 ARIMA 对复杂的非线性问题进行建模都是不合适的。

本文中提出混合模型策略用于受软件老化问题影响的 IIS Web 服务器中的资源消耗预测。提出此混合模型的动机^[8]如下。首先,在实际工作环境中往往很难去判断资源消耗序列整体上是线性的还是非线性的,因此很难选择一种合适的办法(线性还是非线性方法)去预测资源消耗序列。而且由于很多因素的影响,如模型的非确定性、采样频率、参数的确定等,通常很难找到一个最优的模型。通过使用所提出的混合模型的办法,模型选择的问题可以得到很好地解决。其次,资源消耗时间序列通常不是纯粹的线性或者纯粹的非线性关系。资源消耗序列通常包含线性和非线性模式,而这意味着单独将线性或非线性模型用于资源消耗序列的建模可能都是不合适的。最后,一种被普遍接受的观点是:没有任何一种模型适合于所有的现实情形。在对资源消耗进行准确预测之后,为了更好地执行抗衰,提出了多门限值时间段检测算法用于选择抗衰时机。

本文第 2 节提出了混合模型的方法;第 3 节提出了多门限值时间段检测算法用于选择抗衰时机;第 4 节验证了本文提出方法的有效性;最后是对本文的总结。

2 混合模型方法

Denton^[9]指出,当数据中存在多重共线性或者离群点时,人工神经网络的表现要明显强于线性回归模型,Markham 等人^[10]发现人工神经网络对线性问题的处理效果依赖于样本的大小和其中存在的噪声。Balabin 等^[11]比较了 14 个不同数据集中支持向量回归和人工神经网络的预测效果,指出支持向量回归在精度上与人工神经网络相当。Taskay 等^[12]指出通过使用不同的模型可以减小模型选择的失败几率,并提高预测的准确率,这是因为数据中潜在的模式很难通过一种模型被完全发现^[13]。使用不同类型的模型组成混合模型的动机是基于以下假设:很难通过单一类型的模型确定数据的产生过程^[14],同时单一类型模型也很难刻画出数据的所有特征^[8]。由于现实数据本身的复杂性,使用线性和非线性混合模型将会是一个好的解决策略。事实上,在混合模型中使用相互之间差异很大的方法用于建模往往会产生更好的效果^[15]。

任何一个选定的资源消耗序列可以看作是由一个线性自相关部分和非线性部分构成。该混合模型可以表示如下:

$$y_t = L_t + N_t \quad (1)$$

其中, L_t 表示资源消耗序列中的线性部分, N_t 表示非线性部分,表示整个资源消耗序列函数。上式中蕴含着一个假设:一个资源消耗序列函数可以由线性部分和非线性部分通过简单叠加得到。

为了求解上式,可以分为两步进行计算:

(1)使用线性模型对序列中的线性部分进行建模。之后

该序列的残差部分将仅包含非线性的部分。这里使用 e_t 表示在时刻 t 的残差:收集的原始观测值减去线性模型的预测值。

$$e_t = y_t - \hat{L}_t \quad (2)$$

其中, $\hat{L}_t$ 表示在时刻 t 的线性部分的预测值, y_t 表示原始序列观测值。对于一个线性模型来说,残差可以用来检测线性模型是否满足要求,如果在残差序列中仍然存在线性相关结构,那么该选定的线性模型被认为不满足要求;然而即使残差序列通过了线性相关性检测,残差分析技术仍然不能检查出残差序列中的非线性关系,模型可能仍然不满足要求。

(2)残差序列中的非线性部分可以表示为:

$$e_t = f(e_{t-1}, e_{t-2}, \dots, e_{t-n}) + u_t \quad (3)$$

其中, n 表示残差序列中输入变量的个数, $e_{t-1}, e_{t-2}, \dots, e_{t-n}$ 表示残差序列中的过去值, e_t 表示在 t 时刻时残差序列的当前值。 u_t 表示随机错误;函数 f 可视作对 N_t 的估计值 $\hat{N}_t$, 因此混合模型可以表示为:

$$\hat{y}_t = \hat{L}_t + \hat{N}_t \quad (4)$$

其中, $\hat{L}_t$ 表示线性模型的输出; $\hat{N}_t$ 表示非线性模型的输出; $\hat{y}_t$ 表示对资源消耗序列进行拟合的拟合值。

2.1 资源消耗序列的线性部分拟合

使用的 ARIMA 对资源消耗序列的线性部分进行建模的算法描述如下。

输入参数:资源消耗序列

输出参数:资源消耗序列的线性部分和残差序列

步骤:

(1)数据预处理。通过对数据进行转换,如对序列值取平方根或者取对数,并进行适当的差分,一个不稳定的时间序列将成为平稳化的时间序列。

(2)模型定阶。通过自相关函数和偏自相关函数确定模型的阶。

(3)参数估计。通过一些方法,如对数似然函数、Akaike 信息准则(AIC)或者 Schwartz 信息准则(SBC),可以确定 ARIMA 模型中的相关参数。

(4)模型检查。检查模型是否遵循稳态过程准则,如果遵循,则可以使用模型;如果不遵循,则转(2)。

(5)预测。如果模型通过了模型检查,使用训练好的模型拟合资源消耗序列中的线性部分,然后用原始观测值减去 ARIMA 的预测值就可以得到残差序列。

2.2 资源消耗序列的非线性部分拟合

人工神经网络可以处理并行数据,并且可以对非线性数据进行建模。多层感知器人工神经网络尤其是单层人工神经网络是用来对非线性问题建模和预测的使用最广的模型之一。对于单层的感知器人工神经网络来说,资源消耗残差序列中的输出结果(将来值) e_t 和输入变量 $(e_{t-1}, \dots, e_{t-p})$ (历史值)可以表示为:

$$e_t = w_0 + \sum_{j=1}^p w_{j0} g(w_{j0}) + \sum_{i=1}^q w_{ij} e_{t-i} + u_t \quad (5)$$

其中, w_{ij} ($i=0, 1, 2, \dots, p, j=1, 2, \dots, q$) 和 w_j ($j=0, 1, 2, \dots, q$) 是模型的连接权值参数, p 是输入变量的个数, q 是隐含层中节点的个数, $g(e)$ 被称为激活函数。通常 Sigmoid 函数被用来作为隐含层的激活函数:

$$\text{sig}(e) = \frac{1}{1 + \exp(-e)} \quad (6)$$

感知器人工神经网络使用历史值预测将来值的函数可以表示为:

$$e_t = f(e_{t-p}, \dots, x_{t-1}, W) + u_t \quad (7)$$

其中, W 是一个参数向量, 用于表示使用到的所有参数, f 是一个决定神经网络结构和参数的函数。由上式可以看出, 人工神经网络可以简单地等价于一个非线性自回归模型。在实践中, 简单的网络结构(隐含层含有较少的节点)往往表现地更加出色一些, 这是由于过于复杂的人工神经网络在模型训练期间可能会见过拟合的现象。当人工神经网络训练迭代的次数超过其需要的次数时, 模型可能会去拟合训练集中的噪声而非数据, 而这将造成模型泛化能力的下降。

模型中隐含层的层数的选择依赖于数据的特征, 目前没有一种系统的方法用于选择该参数。同样, 对于历史值个数的选择也没有一个统一的标准。因此只能通过实验的方式选择隐含层的层数和输入节点的个数。

3 多门限值时间段抗衰算法

抗衰门限值的确定确定了抗衰的选择时机问题。基于门限值的方法首先定义老化参数的临界值, 当这些被观测的老化参数的值超过预定义的临界值时, 将会执行抗衰操作以消除老化引起的问题。Matias^[16]将虚拟内存作为软件老化标示, 指定 Apache 服务器的虚拟内存临界值, 当监视的虚拟内存值超过指定的临界值时执行抗衰操作, 他在实验中使用 3 个可控的负载参数: 页大小、页类型(动态或静态)、请求率, 来控制 Apache 服务器的内存消耗速度。Silva 等人^[17]将响应时间作为老化参数, 计算虚拟机上的平均响应时间, 在此基础上给出一个临界值, 一旦超过临界值则在虚拟机上执行抗衰。Zhao 等人^[18]使用局部加权回归算法确定响应时间的拐点, 然后定义一个临界值, 当拐点值超过临界值后, 表明老化出现, 然而响应时间对于一个 Web 服务器往往是通过客户端获得的, 很难通过服务器端获得。Araujo 等人^[6]对一个云计算平台中的老化问题执行多临界值抗衰: 首先使用 4 类时间序列模型预测每个主机节点上的虚拟内存资源消耗, 然后设定一个关键内存使用, 当虚拟内存使用超过关键内存使用的 80% 时发出警告; 当虚拟内存使用超过关键内存使用的 95% 时执行抗衰。然而他们在实验中给出关键内存使用值为 3GB, 但并没有说明为什么这样选择, 同时当虚拟内存使用超过关键内存使用的 95% 时就执行抗衰, 可能存在着误抗衰的问题。对于基于门限值的方法, 当指定的临界值过大, 可能导致系统长久处于老化状态, 过小的临界值会导致系统在正常状态下过早地执行抗衰操作。而且其采用的抗衰策略(一旦观测值超过临界值即执行抗衰)也可能造成误抗衰的问题: 资源消耗超过临界值可能是由一段时间的高负载引起的, 而当这些负载处理完毕后, 资源的占用率会降低到正常的水平, 这种超过临界值的现象并非是老化问题引起的。因此指定一个区间段, 当观测值超过临界值一段时间后判断软件出现老化现象相对合理一些。

本节提出了一种多门限值时间段抗衰算法用于抗衰操作的执行, 通过该算法可以避免基于门限值抗衰引起的问题。该算法描述如下。

输入: 资源消耗时间序列, 资源消耗数据 x_t , 资源消耗数据个数 g_w , 时间段 t_{span} , 计数 t_{count} , 多个门限值 $x_{\text{threshold}}$

输出: 系统状态时间段

步骤:

```

1. 初始化时间段  $t_{\text{span}}$ , 计数  $t_{\text{count}}$ , 多个门限值  $x_{\text{threshold}}$ 
2. for  $i=1$  to  $g_w$  do
    if ( $x_t > x_{\text{threshold}}$ ) then
        {  $t_{\text{count}}++$ 
          将资源消耗数据及相应的时刻压栈
          if ( $t_{\text{count}} > t_{\text{span}}$ )
              {将资源消耗数据及相应的时刻存入数据表清空栈}
          else
              {  $t_{\text{count}}=0$ 
                清空栈}
        }
    end for
3. 输出满足要求的资源消耗序列

```

4 实验结果与分析

实验所用到的数据是从一个商业运行的 IIS Web 服务器中采集到的。Web 服务器中的服务是由一系列的商业应用所组成的, 这些应用包括: 各个医院网站、卫生系统行政部门的网站以及一些诸如网上挂号系统等商业服务。用户能够访问的资源类型包括: html 页面、图片文件(如 jpg 和 gif 文件)、flash、视频等。由于整个 Web 应用所在的服务器并不使用双机热备技术, 因此当软件老化问题出现时, 只能通过简单的重启应用来解决老化引起的问题。

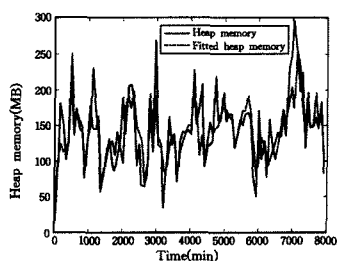
尽管有很多的工具可以用来捕获软件系统中的运行参数值, 例如 Nagios 等, 但是在这里使用 Windows 操作系统内置的计数器(通过此计数器可以得到操作系统的运行参数、Web 服务器运行参数、数据库运行参数等相关参数, 而不会影响系统的正常运行)来捕获实验所需的相关参数。整个数据的收集工作开始于 2012 年 9 月 20 日, 结束于 2013 年 3 月 5 日, 持续了约 6 个月时间, 收集的数据不仅包含资源消耗数据, 如可用内存、堆内存等, 也包含负载数据等其他相关的数据, 如虚拟内存、CPU 使用率等。收集的参数类型包括操作系统参数、IIS 参数等相关参数, 共计 101 个参数, 如 Web 服务、进程、.Net 运行信息等, 收集的数据集个数为 63 个, 除去那些由于软件更新等问题引起的服务重启数据集后, 数据集共 21 个。由于整个 Web 应用是一个商业的 Web 应用, 所有的用户访问均为真实的用户访问, 因此形成的老化数据是正常的用户访问造成的。本文为了验证所提 ARIMA 模型对资源消耗预测的准确性, 使用其中采集到的参数 .Net 堆内存^[19](简记为堆内存)对资源消耗进行预测。

4.1 混合模型预测堆内存

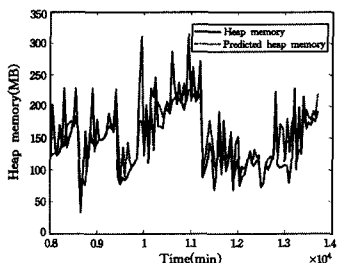
Li 等人^[20]在分析 Apache 服务器中的老化问题时, 将收集到的 7101 个数据中的 1 到 4000 的数据(约占总数据量的 56%)作为训练集训练模型; 将 4001 到 7101 的数据作为验证集。本文将收集到的老化数据集分为两个部分: 训练集和验证集。整个数据集的前 8000 个数据(大约占有所有数据的 70%)作为训练集中的数据用于训练模型; 数据集集中的剩余数据(占有所有数据的 30%)作为验证集。

通过多次训练和验证, 选择 ARIMA (3, 1, 3) 作为混合模型的线性部分; 非线性部分包含 3 个输入节点, 1 个含有 5 个节点的隐含层以及 1 个输出节点的输出层, 简称为 $N(3-5-1)$ 。

混合模型拟合的训练集和预测的验证集如图 1 所示。

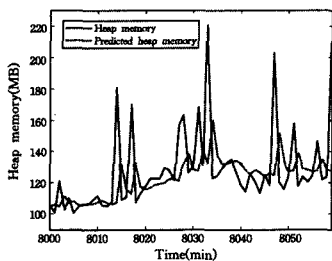


(a)训练集上拟合的堆内存混合模型

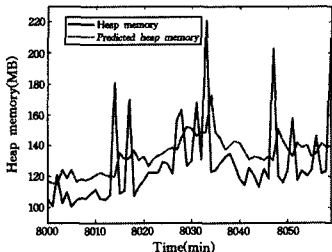


(b)验证集上的混合模型

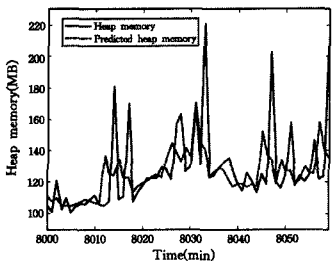
图 1



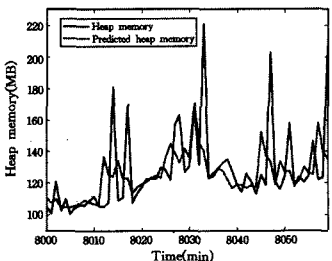
(a)ARIMA 的预测结果(前 60 个点)



(b)神经网络的预测结果(前 60 个点)



(c)支持向量回归的预测结果(前 60 个点)



(d)混合模型的预测结果(前 60 个点)

图 2

图 1(b)从整体上反映了混合模型的预测能力,但还需了解混合模型及其他模型短期的预测效果。图 2(a)至图 2(d)给出了验证集中前 60 个数据关于 ARIMA^[21]、神经网络^[22]、支持向量回归^[23]、混合模型的预测结果。

4.2 结果比较

本节将所提出的混合模型的预测能力与其他模型(ARIMA、神经网络、支持向量回归模型)进行比较。使用平均绝对误差作为模型评判的依据:

$$MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t| \quad (8)$$

式中,MAE 指平均绝对误差, y_t 指预测值, $\hat{y}_t$ 指观测值, n 为观测值的个数。

所提混合模型的预测结果与 ARIMA、神经网络、支持向量回归结果的比较见表 1。

表 1 混合模型与其他模型预测结果比较

模型	前 60 个点	最后 60 个点	验证集上的所有数据
ARIMA	15.625	19.413	25.303
人工神经网络	20.408	18.977	26.429
支持向量回归	16.739	24.908	25.741
混合模型	14.713	18.601	23.96

表 1 中支持向量回归的预测效果要好于神经网络,但是从最后 60 分钟的预测结果来看,支持向量回归在所有模型中的预测结果是最差的。与 ARIMA、神经网络和支持向量回归相比,混合模型有一个较好的预测结果。通过以上模型的比较,发现混合模型无论在长期还是短期的预测上都要好于单一模型。

4.3 多门限值时间段抗衰算法

本节使用提出的算法来执行 IIS Web 服务器的抗衰。文献[18]使用响应时间来判断老化,然而响应时间往往是通过客户端获得的,由于受各种情况的限制,如网络环境,很难获得响应时间的准确值,在本节使用堆内存选择抗衰时机。与文献[6]类似,本节中将系统的状态表示为 3 个状态:正常状态,警告状态,老化状态。由于直接将最大值作为门限值会造成较大的误差,与文献[6]不同的是,本节中取 IIS Web 服务器中最后 30 分钟堆内存的平均值作为基准,而非使用一个单一的点值,这是因为最后 30 分钟的时间是系统提供服务最后的时间,系统此时已处于老化状态,使用其平均值作为基准有助于对系统状态的判断。将平均值的 90%作为老化状态,用于表示系统在当前状态可以执行抗衰;将平均值的 80%作为警告状态,表示系统可能随时进入老化状态,用于警示。

表 2 列出了使用混合模型拟合数据时的多门限值时间段抗衰算法预测的老化状态区间:最后 30 分钟拟合堆内存的平均值为 196.6MB,取其 90%为 177MB。输入参数值为:门限值 177MB,时间段 60 分钟。

表 2 使用混合模型拟合的堆内存 IIS 老化门限值

预测状态	开始时间	结束时间
老化状态 1	10233	10393
老化状态 2	10407	11196
老化状态 3	13462	13714

老化状态 1 的持续时间是 5:42 分到 8:32 分;老化状态 2 的持续时间是 8:46 分至 21:55 分;老化状态 3 的持续时间是 11:41 分至 15:53 分。从用户的角度考虑,在老

化状态 1 上执行抗衰操作比较合理,对用户访问的影响是最小的。

如果使用单门限值算法选择抗衰区间,我们发现使用 177MB 作为门限值所预测的老化区间范围为:475 至 522, 524 至 527 等,可以看到使用单门限值算法会出现大量的误报现象。

表 3 列出了使用原始堆内存数据时的多门限值时间段抗衰算法预测的老化状态区间:最后 30 分钟拟合堆内存的平均值为 198.4MB,取其 90% 为 178MB。输入参数值为:门限值 178MB,时间段 60 分钟。

表 3 使用原始堆内存 IIS 老化门限值

预测状态	开始时间	结束时间
老化状态 1	9373	9442
老化状态 2	10429	10522
老化状态 3	10549	11202
老化状态 4	13561	13631

比较表 2 和表 3,发现使用混合模型拟合的数据直到 10292 分钟才认为 IIS 服务器出现了老化问题,而使用原始数据在 9432 分钟就认为 IIS 服务器出现了老化问题,使用混合模型拟合的数据确定老化状态的时间较原始数据要晚 660 分钟,也就是 11 个小时。

如果使用单门限值算法选择原始数据的抗衰区间,我们发现使用 178MB 作为门限值所预测的老化区间范围为:68 至 71,92 至 94 等,可以看到使用单门限值算法会出现大量的误报现象。

表 4 列出了使用混合模型进行拟合数据时的多门限值时间段抗衰算法预测的警告状态区间:最后 30 分钟拟合堆内存的平均值为 196.6MB,取其 80% 为 157MB。输入参数值为:门限值 157MB,时间段 60 分钟。

表 4 使用混合模型拟合的堆内存 IIS 警告门限值

预测状态	开始时间	结束时间
警告状态 1	6839	7304
警告状态 2	9155	9449
警告状态 3	10232	11204
警告状态 4	13392	13714

警告状态 1 的持续时间是从 21:18 分到 5:03 分;警告状态 2 的持续时间是从 11:54 分至 16:48 分;警告状态 3 的持续时间是从 5:51 分至 22:03 分;警告状态 4 的持续时间是从 10:31 分至 15:53 分。可以看到警告时间区域和老化时间区域之间存在着重叠。

如果使用单门限值算法选择抗衰区间,我们发现使用 157MB 作为门限值所预测的警告区间范围为:182 至 185, 422 至 425 等,可以看到使用单门限值算法会出现大量的误报现象。

表 5 列出了使用原始堆内存数据时的多门限值时间段抗衰算法预测的老化状态区间:最后 30 分钟拟合堆内存的平均值为 198.4MB,取其 80% 为 159MB。输入参数值为:门限值 159MB,时间段 60 分钟。

表 5 使用原始堆内存 IIS 警告门限值

预测状态	开始时间	结束时间
警告状态 1	5527	5708
警告状态 2	7202	7286
警告状态 3	9300	9442
警告状态 4	10248	11203
警告状态 5	13391	13714

比较表 4 和表 5,发现使用混合模型拟合的数据直到 6898 分钟才认为 IIS 服务器出现了老化问题,而使用原始数据在 5586 分钟就认为 IIS 服务器出现了老化问题,使用混合模型拟合的数据确定老化状态的时间较原始数据要晚 1312 分钟。

如果使用单门限值算法选择原始数据的抗衰区间,我们发现使用 159MB 作为门限值所预测的警告区间范围为:31 至 33,68 至 71 等,可以看到使用单门限值算法会出现大量的误报现象。

通过使用提出的算法来执行 IIS Web 服务器的抗衰区间的判定,发现使用混合模型拟合的数据无论是在老化状态确定还是警告状态确定上都要优于使用原始数据的判定;同时使用提出的算法在对状态区间的预测效果上也要优于单门限值算法确定的状态区间。

我们发现重启操作系统的时间为 3 分 30 秒,而重启 IIS 仅需要 20 秒的时间,如果以一年作为一个时间跨度单位,那么通过提前发现老化状态,重启 IIS 的累计时间为 12.8 分钟;而如果不能发现老化,当老化出现时,重启系统的累计时间为 134.4 分钟。也就是说通过提前发现老化状态,执行抗衰系统的不可用时间减少了 90%。

结束语 软件老化和抗衰在近几十年来已成为一个活跃的研究领域。本文首先提出了一种混合模型方法用于提高由于软件老化引起的资源消耗的预测精度问题。为了更好地执行抗衰,提出了多门限值时间段检测算法用于选择抗衰时机。通过遭受老化影响的 IIS Web 服务器中堆内存实验,我们发现混合模型在堆内存的预测结果要好于使用单一的模型。从抗衰时机的选择上来看,所提出的多门限值时间段抗衰算法的效果也要好于单门限值抗衰算法。

参 考 文 献

[1] Huang Y,Kintala C,Kolettis N,et al. Software rejuvenation: Analysis, module and applications [C]// Twenty-Fifth International Symposium on Fault-Tolerant Computing. IEEE, 1995; 381-390

[2] Grottke M, Matias R, Trivedi K S. The fundamentals of software aging [C]// IEEE International Conference on Software Reliability Engineering Workshops. IEEE, 2008; 1-6

[3] Magalhaes J P, Silva L M. Prediction of performance anomalies in web-applications based-on software aging scenarios [C]// 2010 IEEE Second Int'l. Workshop on Software Aging and Rejuvenation. IEEE, 2010; 1-7

[4] Li S, Yong Q. Software aging detection based on NARX model [C]// 2012 Ninth Web Information Systems and Applications Conference. IEEE, 2012; 105-110

[5] Hoffmann G A, Trivedi K S, Malek M. A best practice guide to resource forecasting for computing systems [J]. IEEE Transactions on Reliability, 2007, 56(4); 615-628

[6] Araujo J, Matos R, Maciel P, Vieira F. Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds [C]// 2011 IEEE Third International Workshop on Software Aging and Rejuvenation. IEEE, 2011; 38-43

- [7] Fishwick P A. Neural network models in simulation; a comparison with traditional modeling approaches [C]// Proceedings of the 21st Conference on Winter Simulation. ACM, 1989; 702-709
- [8] Zhang G P. Time series forecasting using a hybrid ARIMA and neural network model [J]. Neurocomputing, 2003, 50; 159-175
- [9] Denton J W. How good are neural networks for causal forecasting [J]. The Journal of Business Forecasting, 1995, 14(2); 17-20
- [10] Markham I S, Rakes T R. The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression [J]. Computers & Operations Research, 1998, 25(4); 251-263
- [11] Balabin R M, Lomakina E I. Support vector machine regression (SVR/LS-SVM)—an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data [J]. Analyst, 2011, 136(8); 1703-1712
- [12] Taskaya T, Casey M C. A comparative study of autoregressive neural network hybrids [J]. Neural Networks, 2005, 18(5); 781-789
- [13] Hibon M, Evgeniou T. To combine or not to combine; selecting among forecasts and their combinations [J]. International Journal of Forecasting, 2005, 21(1); 15-24
- [14] Terui N, Van DHK. Combined forecasts from linear and nonlinear time series models [J]. International Journal of Forecasting, 2002, 18(3); 421-438
- [15] Perrone M P, Cooper L N. When networks disagree: Ensemble methods for hybrid neural networks [R]. Brown Univ Providence RI Inst for Brain and Neural System, 1992
- [16] Matias R. An experimental study on software aging and rejuvenation in web servers [C]// 30th Annual International Computer Software and Applications Conference. IEEE, 2006; 189-196
- [17] Moura Silva L, Alonso J, Silva P, Torres J. Using virtualization to improve software rejuvenation [J]. IEEE Transactions on Computers, 2009, 58(11); 1525-1538
- [18] Zhao J, Trivedi K S, Wang Y B, et al. Evaluation of software performance affected by aging [C]// 2010 IEEE Second International Workshop on Software Aging and Rejuvenation. IEEE, 2010; 1-6
- [19] Matias R, Costa B E, Macedo A. Monitoring memory-related software aging: An exploratory study [C]// 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops. IEEE, 2012; 247-252
- [20] Li L, Vaidyanathan K, Trivedi K S. An approach for estimation of software aging in a web server [C]// 2002 International Symposium on Empirical Software Engineering. IEEE, 2002; 91-100
- [21] Yan Yong-quan, Guo Ping. Predicting resource consumption in a Web server using ARIMA model [J]. Journal of Beijing Institute of Technology, 2014, 4(4); 502-510
- [22] El-Shishiny H, Sobhy Deraz S, Badreddin OB. Mining software aging: A neural network approach [C]// IEEE Symposium on Computers and Communications. Marrakech, Morocco; IEEE, 2008; 182-187
- [23] Avritzer A, Weyuker E. Monitoring Smoothly Degrading Systems for Increased Dependability [J]. Empirical Software Eng. J., 1997, 2(1); 59-77

(上接第 42 页)

- 刘丽英. 线性自抗扰控制策略在异步电机调速系统中的应用研究[D]. 天津: 天津大学, 2010
- [6] Kang Ying, Li Dong-hai, Lao Da-zhong. Performance comparison between active disturbance rejection control and Sliding mode control in spacecraft attitude [J]. Control theory and applications, 2013, 30(12); 1623-1629 (in Chinese)
康莹, 李东海, 老大中. 航天器姿态的自抗扰控制与滑模控制的性能比较[J]. 控制理论与应用, 2013, 30(12); 1623-1629
 - [7] Liu Xiang, Li Dong-hai, Jiang Xue-zhi, et al. Application of self immunity controller in high order systems [J]. Journal of Tsinghua University (Natural Science Edition), 2001, 41(6); 95-99 (in Chinese)
刘翔, 李东海, 姜学智, 等. 自抗扰控制器在高阶系统中应用的仿真[J]. 清华大学学报(自然科学版), 2001, 41(6); 95-99
 - [8] Gao Z Q. Scaling and Bandwidth-Parameterization Based Controller Tuning [C]// Proc of the 2003 American Control Conf. Denver; IEEE, 2003; 4989-4989-4996
 - [9] Chen Xing. Parameter tuning method for the active disturbance rejection controller and its application in thermal engineering. [D]. Beijing; Tsinghua University, 2008 (in Chinese)
陈星. 自抗扰控制器参数整定方法及其在热工过程中的应用 [D]. 北京: 清华大学, 2008
 - [10] Shao Xing-ling, Wang Hong-lun. Performance analysis of linear extended state observer [J]. Control and Decision, 2015, 30(5); 815-822 (in Chinese)
绍星灵, 王宏伦. 线性扩张状态观测器及其高阶形式的性能分析 [J]. 控制与决策, 2015, 30(5); 815-822
 - [11] Zhang Jiao, Yang Xu, Liu Yuan-xiang, et al. Performance evaluation of high order linear active disturbance rejection controller [J]. Control and Decision, 2015, 30(7); 1162-1170 (in Chinese)
张皎, 杨旭, 刘源翔, 等. 高阶线性自抗扰控制器的性能评估 [J]. 控制与决策, 2015, 30(7); 1162-1170
 - [12] Duan Hui-da, Tian Yan-tao, Li Jin-song, et al. Cascade of a class of high order nonlinear systems [J]. Control and Decision, 2012, 27(2); 216-220 (in Chinese)
段慧达, 田彦涛, 李津淞, 等. 一类高阶非线性系统的级联自抗扰控制 [J]. 控制与决策, 2012, 27(2); 216-220
 - [13] Li Shu-qing, Zhang Sheng-xiu, Liu Yi-nan, et al. Parameters of the anti disturbance controller, based on system time scale [J]. Control Theory and Applications, 2012, 29(1); 125-129 (in Chinese)
李述清, 张胜修, 刘毅男, 等. 根据系统时间尺度整定自抗扰控制器参数 [J]. 控制理论与应用, 2012, 29(1); 125-129
 - [14] Shao Li-wei, Liao Xiao-zhong, Zhang Yu-he. Parameter tuning of based on time scale for induction motor [J]. Control Theory and Application, 2008, 25(2); 205-209 (in Chinese)
邵立伟, 廖晓钟, 张宇河. 基于时间尺度的感应电机自抗扰控制器的参数整定 [J]. 控制理论与应用, 2008, 25(2); 205-209