基于 BP 神经网络的软件老化趋势预测

赵靖[1]，陈小勇[1]，高振国[2]
（1．哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001；
2．哈尔滨工程大学自动化学院，哈尔滨 150001）

摘要：为了避免软件老化对计算机系统造成的损失，可以利用基于神经网络的方法对软件老化进行预测。以 Apache 软件为研究对象，首先提出了一种实验方案；然后建立了 BP 神经网络模型，输入训练数据对 BP 神经网络进行训练，再输入检验数据并输出预测结果；最后对和趋势估计对实验数据处理的结果进行了对比。实验结果表明神经网络具有较好的预测效果。

关键词：软件老化；神经网络；趋势估计

中图分类号：TP302.7

Software aging prediction based on BP neural networks

Zhao Jing[1], Chen Xiaoyong[1], Gao Zhenguo[2]
(1. Department of Computer Science and Technology, Harbin Engineering University, Harbin 150001;
2. Department of Automation and Control college, Harbin 150001)

Abstract: In order to avoid the loss of computer systems resulted from software aging, neural network was used to predict the performance aging. Firstly, proposed an experimental program with studying Apache software; Secondly, established neural network model, inputted train data, trained on the network , inputted test data and outputted the prediction results; Finally, manipulated data with the neural network method and the slope estimation method respectively and compared the results. The experimental results indicate that the performance of the neural network is more effective.

Keywords: Software aging; Neural networks; Trends estimation

0 引言

随着计算机系统的飞速发展，计算机的软件性能得到不断提高，计算机软件的复杂性也大大提高。计算机软件系统在长期不断的运行过程中，会产生内存泄漏，未释放的文件锁，舍入误差的积累，大量的存储空间碎片等现象，由于这些原因会导致软件性能衰退，这种现象称为软件老化[1,2]。为了对抗这种软件性能衰退，人们提出了种种策略，如基于时间的，基于检测的以及基于时间与负载的软件抗衰策略。研究的最终目的是能够预测软件老化情况，并应用一定的抗衰策略，来消除或者延缓软件老化现象。

当前开展的主要是基于时间的软件抗衰策略[3,4]和基于检测的软件抗衰策略的研究[5,6]。对于具有稳定衰退规律的系统，适宜采用相对简单的基于时间的抗衰策略[5,6]。对于基于时间的抗衰策略，Y.Huang[1]首先提出计算系统的两步失效与恢复模型的马尔科夫模型，Dhoi T[4,7]提出了一种非参数统计方法确定最优重启间隔。如果计算系统的衰退规律不易把握，则应采用相对比较准确的基于检测的抗衰策略，尽管其可能引发较高的成本。

基于检测的抗衰策略关心的是计算机系统运行过程中的内部参数，它具体测量和估算可

基金项目：国家自然科学基金资助项目（60873036），（60703090）；博士点青年基金（20070217051）；中央高校基本科研业务费专项资金（HEUCF100601） 资助

作者简介：赵 靖，女，教授，硕士生导师，主要研究领域为软件可靠性评估、容错计算、软件测试和智能计算. E-mail: zhaoj@hrbeu.edu.cn

能引起软件老化的各种因素及其变化趋势，并由此决定是否与何时采取抗衰措施[8]。Boboio[9]研究了确定某种系统资源的报警阈值的方法，系统监测的采样间隔会影响阈值的确定。蒋乐天[10]等人运用局部强加权算法对响应时间的拐点进行了研究，贾云飞[11]等人研究了影响软件衰退的相关参数。Trivdi K.S[12]总结了以前的工作，还证明了负载也是影响系统资源损耗程度的重要因素，并给出了基于负载的确定最优重启时间的方法。

由于软件老化的复杂性，影响软件老化的因素很多，只研究软件老化对某一参数的影响，不可避免的具有局限性，因此实验中需要采集计算机系统多个参数，综合进行研究。本文实验首先收集多个系统参数和服务器响应时间参数，然后建立 BP 神经网络，利用 BP 神经网络强大的学习能力，对某一参数未来某一时间段的值进行预测。

1 实验设计

1.1 实验设置

使用四台计算机通过交换机连接组成一个网络。其中一台作为服务器（P3，933MHz，Memory 256M，Fedora8），在这台服务器上，安装了 Apache1.3.41 软件，提供 Web 服务。服务器上安装信息监控程序 systeminfo，通过读取 Linux 虚拟文件系统/proc 文件夹中的系统信息文件，收集系统信息；另外三台计算机作为客户机（P4，双核 3GHz），客户机上安装 test 程序，通过 test 周期性调用 Httperf 软件发送负载，并得到 Httperf 返回的服务器平均响应时间信息。Httperf 是一个高效的 Http 压力测试工具，使用它可以模拟出超过 1 千的并发访问，能充分测试出 web 服务器的性能[13]。长时间运行这个系统，通过收集服务器的系统信息和响应时间数据来研究服务器软件系统的老化情况。

实验平台的架构如图 1 所示：

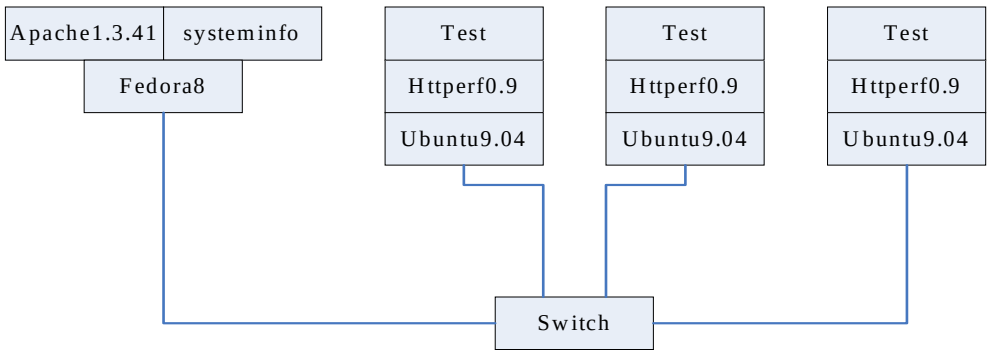

图 1 实验平台架构

Fig.1 Experimental platform framework

由 Apache 的运行机制可知，Apache 软件自身提供了一定的抗衰策略。首先，如果配置文件中的参数 MaxPerquestsPerChild 设为一个非零值 N，那么这个子进程处理完 N 个请求后，就会被父进程所销毁，并重新生成新的子进程以处理请求，这样一定程度上可以避免内存泄漏。如果这个值设置为零值，则程序在运行过程中，子进程永远不会被终止；其次，参数 MaxClients 限制了 Apache 父进程可以产生子进程的最大数目，也就是 Apache 可以同时处理的请求的数目，这就避免了 Apache 占用内存过大。由于每个子进程大约消耗 0.4MB 的内存空间，这个数目如果设定过大，必将很快的耗尽内存，导致系统宕机。如果设置的过小，服务器在还有很多资源的情况下，大量请求进入排队，等待处理，会导致响应速度很慢；最后，

参数 MaxSpareServers 设置了最大的空闲子进程数，如果空闲子进程数大于这个值，Apache 会终止一些空闲进程，以减少内存占用和泄漏。如果这个值设置小于 MinSpareServers 的设置，Apache 会自动调整为 MinSpareServers+1。为了加速软件的老化[14,15]，将 Apache 自身的抗衰策略取消，所以将 Apache 的参数设置为 MaxPerquestsPerChild=0；MaxClients=MinSpareServers=StartServer=250；MaxSpareServers=0。

1.2 测定服务器容量

服务器容量就是服务器每秒所能处理的最大请求数，在实验之前，需要测定服务器的容量，用来设定客户机向服务器发送多大的负载，可以通过 Lei Li[14,15]等人的方法来测定服务器的容量。在实验中，我们通过程序周期性的调用 Httperf，不断加大 Httperf 的发送速率，收集 Httperf 返回的各种信息，并分析这些返回信息，比如：连接率、错误率等，来判断服务器的容量。如图 2 所示,(a)为错误率，（b）为连接率。

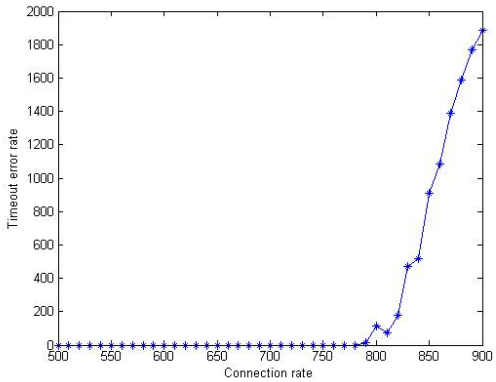

图 2 测定服务器的容量(a)
Fig.2 Testing the capacity of the server (a)

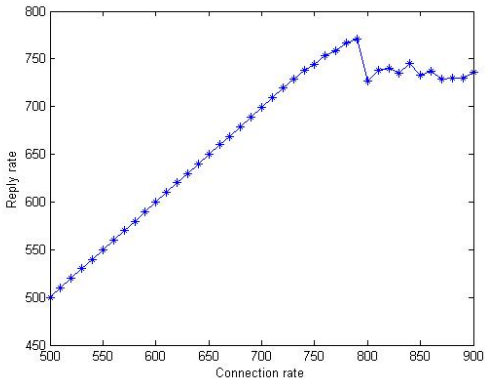

图 2 测定服务器的容量(b)
Fig.2 Testing the capacity of the server (b)

从图 2 中可以看出，当客户机的发送速率达到 770r/s 左右时，服务器的响应率急剧下降，错误率急剧上升，由此可以判断服务器的容量在 770r/s。同样是为了加速 Apache 软件的老化，需要使 Apache 软件满负荷工作，三台客户机均衡同时向服务器发送负载，因此设定每台客户机发送的请求是 255r/s。

2 实验数据采集

在实验中，客户端不断访问 Web 服务器的一个静态页面，无需大量的计算，因此 Web 服务器是一个 I/O 繁忙型的而不是一个 CPU 繁忙型的。采样时间的设置在实验中也很重要，如果采样时间设置的太大，不能够反映系统资源变化的细节趋势，如果采样时间太小，又会给服务器带来较大的负担。在实验中，设定采样时间间隔为 5 分钟。收集到的系统信息见表 1。

表 1 系统信息
Tab.1 System information

系统参数	参数意义
ResponseData	服务器的响应时间
BufferCacheUsed	系统中实际使用的内存
LoadAvg	系统的平均负载

3 神经网络模型建立

3.1 建立模型

建立的数据模型对于神经网络的性能有至关重要的影响，它影响到神经网络的学习能力和泛化能力，因此必须根据我们所研究的问题，建立合适的数据模型来进行神经网络训练。本文实验以五天内检测到的数据进行分组，一共取 1440 个数据将数据分成三组，以第一天，第二天，第三天的数据为训练数据集，以第四天数据为目标数据集，以第二天，第三天，第四天的数据为测试数据集，用来预测第五天的数据，可以用第五天测得的实际数据与之进行对比，检验神经网络的性能。为了提高神经网络的拟合能力，对输入输出数据均进行归一化处理。

神经网络结构如图 3。

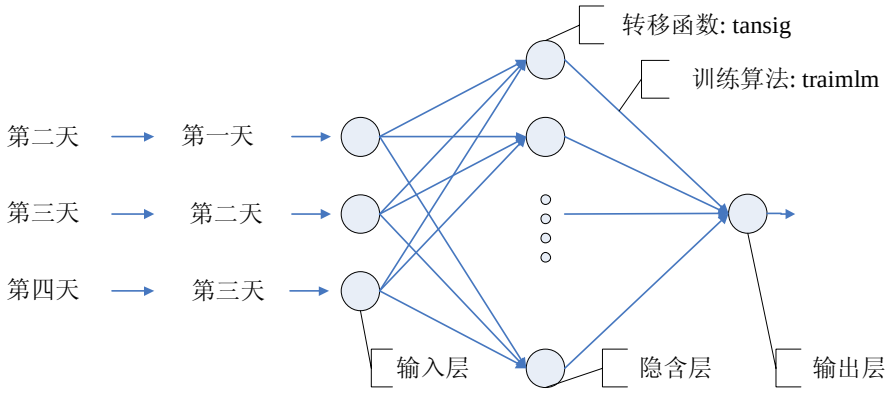

图 3 神经网络结构图

Fig.3 Neural network structure

3.2 神经网络预测及趋势估计

系统实际占用的内存的预测及趋势估计如图 4。

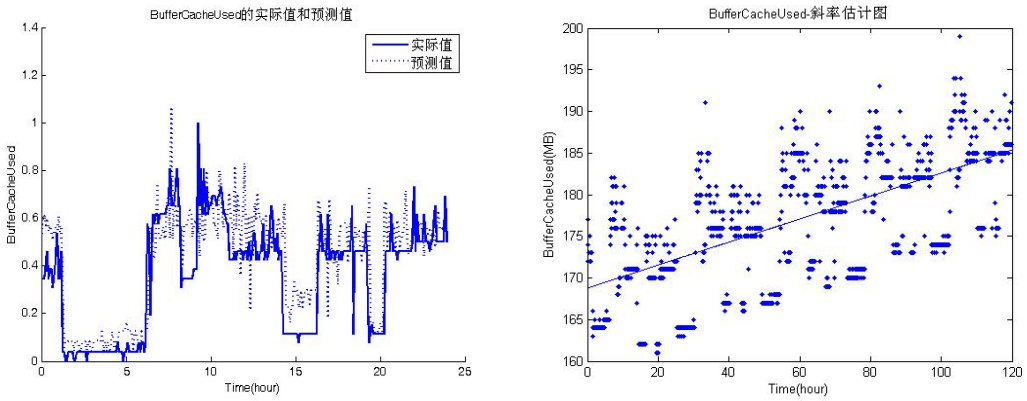

图 4 BufferCacheUsed 的预测值和趋势估计

Fig.4 Predicted value and trend estimation of BufferCacheUsed

系统平均负载的预测及趋势估计如图 5。

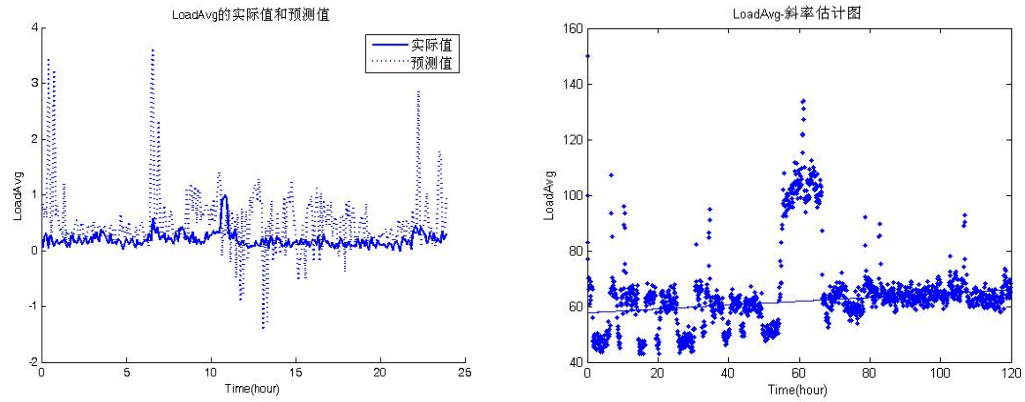

图 5 LoadAvg 的预测值和趋势估计

Fig.5 Predicted value and trend estimation of LoadAvg

响应时间的预测及趋势估计如图 6。

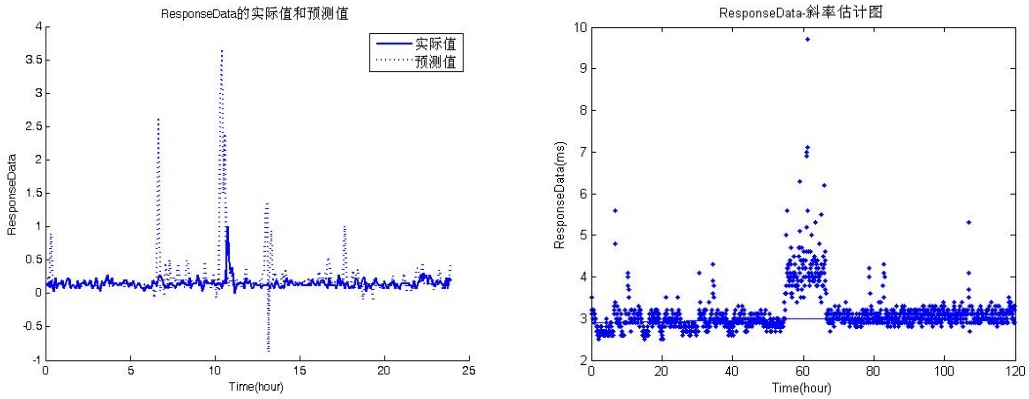

图 6 ResponseData 的预测值和趋势估计

Fig.6 Predicted value and trend estimation of ResponseData

从上面的对比可以看出，神经网络对特征明显的数据学习效果较好，能够正确的预测某一时间段的数据。趋势估计也能够正确的反映出某一参数现有数据整体上的趋势，概略地对未来的参数变化进行预测，整体上来看，预测效果不如神经网络。表 2 列出了神经网络和趋势估计的参数对比。Sen 为趋势估计时参数数据的斜率，TrainMse 为神经网络训练时的均方差，TestMse 为神经网络预测时的均方差。均方差计算公式为：$Mse=\sqrt{\frac{\sum_{i=1}^{n}(X_p-X_m)}{n-1}}$，$X_p$ 是参数的预测值，X_m 是参数的实际值，n 为样本数。

表 2 神经网络和趋势估计的参数对比

Tab.2 Neural network and trend estimation parameters

项目 \ 参数	ResponseData	LoadAvg	BufferCacheUsed
Sen	0.0017725	0.068943	0.13816
TrainMse	0.0106	0.0052	0.0064
TestMse	0.1467	0.3938	0.0232

4 结论

在 Web 服务器长时间运行 Apache 软件，Apache 软件的性能因为软件老化而逐渐下降。

为了研究这种老化现象，本文收集系统参数和响应时间信息，建立神经网络，对未来一时间段内的数据进行了预测。实验结果表明，趋势估计只能够对当前的数据进行评估，而神经网络预测的数据可以对未来某一时间段的数据进行预测，对系统老化情况可以有更好的把握，为采取何种抗衰策略提供依据。

[参考文献] (References)

[1] Huang Y, Kintala C, Kolettis N, et al. Software Rejuvenation: Analysis, Module and Applications[A]. In Proceedings of the 25th Symposium on Fault Tolerant Computer System[C], Pasadena, CA, 1995,381-390.

[2] International Business Machines. IBM Director Software Rejuvenation-White Paper[R].2001.

[3] GargS，Puliafito A.，Trivedi K.S. Analysis of Software Rejuvenation using Markov Regenerative Stoehastic PetriNet.In:ProeeedingsoflSSRE1995, Toulouse, Farnee, 1995, 180-187.

[4] Dohi T, Goseva-Popstojanova K, Trivedi K.S. Statistical Non-Parametric Algorithms to Estimate the Optimal Software Rejuvenation Schedule[A]. In Proc 2000 Pacific Rim International Symposium on Dependable Computing (PRDC 2000)[C], Los Angeles, CA, 2000, 77-84.

[5] Vaidyanathan K.selvamuthu D, Trivedi K.S. Analysis of inspection-Based Preventive Maintenance in Operational Software Systems[A]. Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems[C], 2002, 286.

[6] Bao Y, Sun X, Trivedi K.S. Adaptive Software Rejuvenation: Degradation Model and Rejuvenation Scheme[A]. In:2003 International Conference on Dependable Systems and Networks[C], San Francisco California, 2003, 241- 248.

[7] Dohi T, Danjou T, Okamura H. Optimal Software Rejuvenation Policy with Discounting[A]. In: 2001 Pacific Rim International Symposium on Dependable Computing[C], Seoul, Korea, 2001, 87-94.

[8] Garg S, Moorsel A.V. Towards performability modeling of software rejuvenation[A]. In: Proceedings of the PMCCS-3[C], Bloomingdale. IL. 1996.

[9] Bobbio A., Sereno A., Anglano C. Fine Grained software degradation models for optimal rejuvenation policies[J]. Performance Evaluation. 2001, 45-62.

[10] Letian Jiang, Guozhi Xu. Modeling and analysis of software aging and software failure [J]. The Journal of Systems and Software vol.80, 2007. 590-595.

[11] Yun-Fei Jia, Xiu-E Chen, Lei Zhao and Kai-Yuan Cai et al. On the Relationship between Software Aging and Related Rarameters[A]. The Eighth International Conference on Quality Software[C], Oxford, UK, 2008, 241-246.

[12] Trivedi K.S Vaidynathan K, Goseva-Popstojanova K. Modeling and Analysis of Software Aging and Rejuvenation[A]. In: Proceedings of the 33rd Annual Simulation Symposium[C], Washington, DC, USA 2000, 270 .

[13] http://www.hpl.hp.com/research/linux/httperf/.

[14] Rivalino Matias Jr., Paulo J.F.Filho. An Experimental Study on Software Aging and Rejuvenation in Web Servers[A]. COMPSAC '06. 30th Annual International[C]. Chicago, IL, 17-21 Sept. 2006 189-196.

[15] Michael Grottke, Lei Li, Kalyanaraman Vaidyanathan, and Kishor S.Trivedi. Analysis of Software Aging in a Web Server[J]. IEEE Transactions on Reliability, vol. 55, no. 3, pp. 411–420, 2006.