基于用户体验的计算系统多元性能评价模型

游 静 董小龙 苏 兵 孙玉强
(常州大学信息科学与工程学院 常州 213164)

摘 要 传统的性能监控系统通过监控系统资源的使用情况来间接地反映和评价系统的运行状态,但是该方法与用户感知的系统性能有较大出入。对于终端用户,其最直接的体验就是请求的响应时间,它受系统运行过程中多种因素的综合影响。基于基因表达式编程理论,提出了一种对计算系统性能进行多元评价的基因表达式编程算法,其通过对多种系统资源数据的分析,建立响应时间对多种系统资源的数学模型,以此预测系统性能的变化。最后,针对具体的仿真环境和采样数据,采用该算法获得了响应时间的多元非线性模型。结果表明,该模型能较好地预测系统的性能变化情况。

关键词 性能评价,性能监控,基因表达式编程,软件抗衰
中图法分类号 TP302.7 **文献标识码** A

Multi-factor Performance Evaluation Model of Computing System Based on User Feeling

YOU Jing DONG Xiao-long SU Bing SUN Yu-qiang
(Faculty of Information Science & Engineering, Changzhou University, Changzhou 213164, China)

Abstract Traditional performance monitoring system monitors system resources to reflect and evaluate the system running state indirectly. But this method has greater difference to the system performance perceived by user. For end-users, the direct experience to the system performance is the response time of the request, which is affected by many factors. Based on gene expression programming theory, a gene expression programming (GEP) algorithm was proposed in the paper. Using GEP algorithm, a mathematical model (GEP model) between response time and a variety of system resources was established to evaluate the computing system performance and predict the performance changes. Finally, for specific simulation environment and sampling data, the algorithm was used to obtain the multi-nonlinear model about response time and the results show that the model can better predict the system performance.

Keywords Performance evaluation, Performance monitoring, Gene expression programming, Software rejuvenation

1 前言

计算系统的性能受到多种系统资源使用及损耗情况的综合影响。研究表明,计算系统性能衰退的同时往往伴以系统资源的损耗,如内存泄漏和溢出、未释放的文件锁、存储空间的碎片、不足的交换空间、网络带宽不足、错误的累积和数据的破坏等[1,2]。因此,研究人员针对各类计算系统,如Web服务系统[3]、群集系统[5,6]、分布式企业应用[4,7]、虚拟化计算系统[8,9]等,通过分析其性能变化规律,预测系统可能发生的失效。前期对抗衰技术的研究集中于两方面,一是假定失效率,制定抗衰策略[1,5-7],该方法只适用于失效率已知且运行稳定的系统,实际应用受到很大限制;另一方面是采用性能监控工具,监控系统资源变化并建模,预测系统衰退情况[2-4],该方法的核心是建模方法的选择和应用,这也是本文研究的主要内容。

被服务用户对服务质量的要求是多方面的,包括服务的可靠性、安全性和连续性等。而对于终端用户来说,其最希望得到的是服务的安全及时响应,这种响应可能是不完全的,但必须是快速的、有关键点的、有连续性的。所以,终端用户最直接体验到的响应时间便成为评价系统性能甚至服务质量的一个实际的重要标准。响应时间可以通过长时间监测直接分析其统计规律,但这种方法只适用于被规律访问的服务系统,且无法反映变化细节。因此,还是应该考虑性能衰退的直接诱因,以在一定程度上更真实地反映系统现状。

计算系统的实际性能可能同时受到多种系统资源使用情况的影响,前期研究中的建模方法,如线性回归[2]、ARMA[3,4]、状态空间模型[10]等,均是针对某种单一资源数据分别进行建模,以此定性地反映多种因素对性能的影响。但是,这些模型并未体现多种资源对系统性能的综合影响,更未建立终端用户实际体验到的系统性能与资源使用情况的直接关系。鉴于此,本文基于基因表达式编程理论,提出了一种对计算系统性能进行多元评价的基因表达式编程算法。搭建Web服务器,并针对具体的实验环境,采集多种系统资源的使用数据与请求响应时间,建立响应时间对多种系统资源数

到稿日期:2011-12-21 返修日期:2012-04-05 本文受江苏省自然科学基金项目(BK2009535),江苏省高校自然科学基金项目(07KJB520022),常州市青年人才基金项目(CQ20100007)资助。

游 静(1975—),女,博士,副教授,主要研究方向为性能监控、软件抗衰、云计算等;董小龙(1989—),男,硕士生,主要研究方向为性能监控等。

据的基因表达式编程模型，以此来分析和预测系统性能的变化情况。

2 计算系统性能评价与基因表达式编程

基因表达式编程(GEP)[11]借鉴生物遗传的基因表达式规律，采用等长线性符号作为遗传编码。GEP结合了遗传编程(GP)和遗传算法(GA)的优点，首次实现基因型与表现型的分离，进化能力得到大幅度提高。GEP可以用简单的编码解决一些复杂问题，分析多种因素对某要素的共同作用。因此，本文考虑建立基因表达式编程模型来反映多种系统资源对计算系统性能的动态综合影响。其对应关系如表1所列。

表1 计算系统性能评价与基因表达式编程算法对应关系

计算系统	生物系统
计算系统性能	生物系统健康度
各种作用规则组合	种群
规则搜索	遗传算子
一/多种作用规则	基因/染色体
数学模型	表达式树
影响因素	终结符
拟采用的运算	函数符
模型精确度	适应度

每个染色体包含一个或多个基因，允许对计算系统同时实施多种作用规则；每个基因由头部和尾部两部分组成：头部除了首位置必须为函数符号外，其他的位置可以为函数符号或者终结符号，尾部必须由终结符号组成。这样可以保证表达式树的合法性，也即保证了数据模型的可用性；同时，未表现出来的基因部分同样参与后续的遗传算子运算，并可能会最终作用于数学模型取得更好的适应度，保证了规则搜索的全局性。

3 实验环境

用户感受到的响应时间(T)包括几部分：数据的网络传输时间(T1)和请求的处理时间(T2)。T2可以由被监控端获取，而T1则需要用户端的协助。针对具体问题，决定采用T还是T2。本文拟采用T进行系统性能的评价，因此搭建仿真实验环境，在模拟系统负载的同时，监控各请求的实际响应时间，并同步获取服务系统接收请求时的系统资源使用情况。本文所获取的计算系统相关数据的仿真实验环境如图1所示。

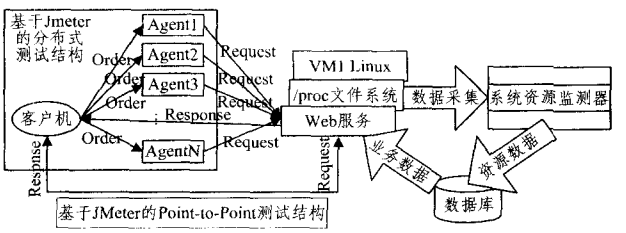

图1 仿真环境

整个仿真环境分3部分：服务系统、测试系统和资源监测系统。服务系统基于Linux平台，通过配置Apache和Tomcat来提供Web服务，Apache负责处理Web服务中的静态页面，Tomcat负责处理动态页面。测试系统产生服务系统负载，提供两种测试结构：基于JMeter的Point-to-Point测试结构和基于JMeter的分布式测试结构，可根据计算系统的实际负载需求选择不同的测试结构，具体参见课题组前期研究[12]。为了尽可能少地损耗系统资源，另行配置一台机器作为资源监测器。运行课题组在Linux环境下开发的性能监控系统，完成被监控系统在运行时资源的采集、存储和分析建模等工作。本文的目的就是进一步完善性能监控系统的功能，基于用户感觉实现对被监控系统的性能的多元评价，进而更精确地预测系统的性能变化，保障系统的可靠性。

4 计算系统多元性能评价的GEP建模

4.1 GEP基本概念

在描述算法之前，首先介绍基因构成和适应度公式。

(1)基因构成：每条染色体包含一个或多个基因，每个基因由头部和尾部两部分组成。头部除了首位置必须为函数符号外，其他的位置可以为函数符号或者终结符号，尾部必须由终结符号组成。其中函数符号是建模过程中拟采用的运算，终结符号包括所有影响系统性能的各类资源数据。头部长度h必须事先确定，尾部长度定义为t，尾部长度和头部长度的关系如下：

$$t=h(m-1)+1 \tag{1}$$

式中，m表示所使用的函数集中需要变量最多的函数的参数个数。

(2)适应度：适应度值决定算法是否可以终止，因此算法在求解过程中需要经常计算每一个染色体的适应度值fv_i。对每一条染色体的适应性进行评估，一般采用如下两种方式：

$$fv_i=\sum_{j=0}^{l}(M-|C_{ij}-T_j|) \tag{2}$$

$$fv_i=\sum_{j=0}^{l}(M-|\frac{C_{ij}-T_j}{T_j}|) \tag{3}$$

式中，fv_i是第i个染色体的适应度，M是一个常量，C_{ij}是第i个染色体对应的基因表达利用第j个样本求得的函数的值，T_j是第j个样本实际对应的函数值，l是样本的个数。

4.2 计算系统多元性能评价的GEP算法

算法要经过多次迭代才能获得评价模型，因此设置变量k记录迭代次数，初值$k=0$；为保证算法可以终止，预定义适应度阈值Fv和迭代次数上限Lim。其中$Fv=\eta M$，η为模型精确度，一般取≥80%。

设置具体算法描述如下：

(1)若$k=0$，采用随机策略，利用符号集Σ和终结符号集中元素产生n个符合基因构成要求的染色体作为初始化种群。

(2)根据式(2)计算种群中每一个染色体的适应度值fv_i，若$fv_i\geq Fv$，则停止运算，将该染色体对应的基因型转化为表现型作为最终的建模结果，输出迭代次数、数学模型及适应度值。

(3)执行单点重组、双点重组、基因重组操作。

(4)执行插串算子、根插串算子、基因插串算子操作。

(5)迭代次数k加1，根据式(2)计算种群中新产生染色体的适应度值fv_i'。

(6)若$fv_i'\geq Fv$，则停止运算，将该染色体对应的基因型转化为表现型作为最终的建模结果，输出迭代次数、数学模型及适应度值。

(7)若$k\geq Lim$则停止迭代，输出迭代次数、具有max(fv_i)的染色体对应的数学模型及其适应度值。

(8)将两代染色体放到一起排序，把适应度值靠前的n个染色体作为新的种群，用于产生下一代个体，然后跳转到步骤(3)。

4.3 GEP 算法参数

本实验中采用函数符号集∑={+,-,*,s,e},其中 s 是 sin 函数,e 是指数函数。终结符号={0,1,2},0 代表 CPU 利用率的值,1 代表内存利用率的值,2 代表采集点的时间值,目标值为被监控系统的响应时间。采用式(2)对染色体的适应度进行评价,l=900,M=10,η=94%,则 Fv=8460。

基因表达式编程模型的训练过程采用的主要参数如下:

(1)种群数量为 50;

(2)每条染色体包含基因个数为 4;

(3)基因头部长度为 4;

(4)基因变异率为 0.044;

(5)插串算子概率为 0.1;

(6)根插串算子概率为 0.2;

(7)单点重组概率为 0.3;

(8)两点重组概率为 0.3;

(9)基因重组概率为 0.1;

(10)最大迭代次数为 1000。

4.4 实验环境配置

实验采用图 1 所示的仿真环境增加负载、模拟衰退、采集数据并建模。实验中测试系统采用分布式测试结构,1 台控制器,3 台 Agent;被监控系统即 Web 服务系统运行在 Linux 环境下,配置 Apache 和 Tomcat 服务器;为了尽可能少地损耗系统资源,另行配置一台机器作为资源监测器。系统硬件配置如表 2 所列。

表 2 实验环境硬件配置

	测试系统		Web 服务	资源监测
	控制器	Agent1/2/3	系统	系统
操作系统	Windows XP	Windows XP	Fedora5.0	Windows XP
内存	DDR2 2G	DDR2 1G	DDR2 1G	DDR2 1G
处理器	Intel (R) Celeron (R) CPU E3200 @2.4GHz	Intel (R) Celeron (R) CPU E3200 @2.4GHz	Intel Pentium 4630@ 2.80GHz	Intel Core2 Duo E4500@ 2.20GHz

4.5 计算系统多元性能评价的 GEP 模型

本文进行了 3 组实验,第一组是响应时间直接对采样时间进行建模得到模型一 $R(t)$;第二组是分别针对 CPU 和物理内存使用率,建立其一元线性回归、ARMA 和 GEP 3 种模型并进行对比;第三组是响应时间对 Cpu、内存和采样时间同时建模得到模型二 $R(Cpu,Mem,t)$。3 组实验各进行了 20 次,试验的成功率均为 100%。

第一组实验平均进化代数为 235.8 次,选择其中较好的模型作为模型一,对应的染色体为-sss1102002 * ss * 0122222-22-2222212 * / * e2001212,解析后的数学模型为 $R(t)=0.0171478+0.0174524\sin t+t/e$,模型的适应度为 8546.911,建模结果如图 2 所示。

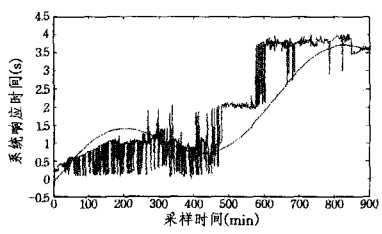

图 2 模型一:GEP 模型 $R(t)$ 的建模结果

传统的建模方法只能对单一资源的数据进行建模,因此,设计第二组实验专门针对同次实验中同步采集的 CPU 和内存数据进行分别建模,建模结果如图 3、图 4 所示。从图中可以看出,一元线性回归模型模拟了 CPU 和内存变化的整体趋势,ARMA 模型则模拟了资源变化的具体细节,而 GEP 模型更精确地关注了系统的整体趋势,并适当反映了系统资源的变化细节。另外,一元线性回归模型简单,但与实际值差距太大,实用性差;ARMA 模型相对另外两种模型要复杂得多,实际应用受到限制;GEP 模型相对简单得多,模型的数学运算和仿真也很方便,实用性强。

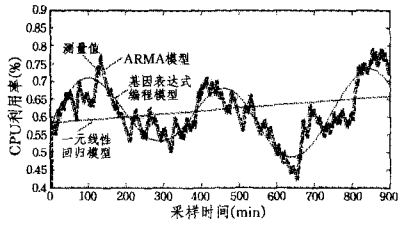

图 3 CPU 利用率模型对比图

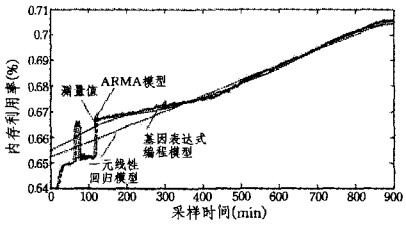

图 4 内存占用率模型对比图

第三组实验平均进化代数为 18.2 次,选择较好的模型作为模型二,对应的染色体为s * * -1122010-11 * 2020101/e * s1212222/es * 2101112,解析后的数学模型为 $R(Cpu,Mem,t)=\frac{Mem*t}{e^{\sin(Mem)}}+\frac{\sin t}{e^{Cpu*Mem}}$,模型的适应度为 8551.21。将实验二中建立的 CPU 和内存的 GEP 模型代入,可以画出考虑了资源变化因素的响应时间和采样时间的关系图,结果如图 5 所示。

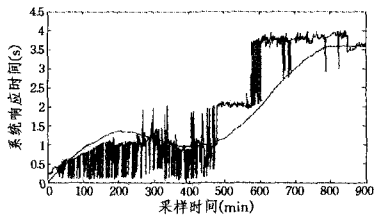

图 5 模型二:GEP 模型 $R(Cpu,Mem,t)$ 的建模结果

从以上实验过程和建模结果可以看出,获得模型一需要较长的时间,且模型的精确度较模型二差一些,这是因为响应时间本身受到多种因素的影响,如系统资源、网络资源和存储资源等。本文实验在局域网中进行,且未进行频繁的数据存取操作,所以主要受到系统资源的影响。系统资源中 CPU 和内存又是两类影响最大的资源,所以当响应时间同时对 CPU、内存和采样时间进行建模时相对速度较快,模型精确度也较高。这也说明基因表达式模型不仅适合建立单元关系,而且较适合建立多元关系,可以同时反映多种因素对性能的综合影响,弥补了传统建模方法的不足。

结束语 考虑到系统性能受到运行环境、网络环境、存储环境等多种因素的综合影响,本文采用基因表达式编程算法建立了单一资源变化模型,并与一元线性回归模型、ARMA

模型做了对比；建立了响应时间对多种资源参数的多元非线性模型，并通过具体实验数据和建模过程分析说明了该方法能较好地应用于这种多因素影响的复杂环境，能够以较快的速度挖掘出其间较复杂的多元关系，较好地预测了系统的性能变化情况。

需要说明的是，本文的实验是在局域网环境下完成的，所以网络传输的影响可以忽略不计。但在实际应用中，需要同时考虑服务器的处理时间和网络传输延迟，所建立的模型还将有助于分析服务的瓶颈并加以改进。对于运行规律多变且受时效影响较大的系统，需要持续监控并建立多阶段数学模型，后期将继续改进算法，实现系统的持续监控和动态性能评价。

参 考 文 献

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

[2] Garg S, Moorsel A V, Vaidyanathan K, et al. A Methodology for Detection and Estimation of Software Aging[C]// Proceedings of the 9th International Symposium on Software Reliability Engineering. Paderborn, Germany, 1998

[3] Grottke M, Lei Li, Vaidyanathan K, et al. An approach for estimation of software aging in a Web server[J]. IEEE Transactions on Reliability, 2006, 55(3): 411-420

[4] 徐文彬，齐勇，侯迪，等. 基于时间序列分析的应用服务器性能衰退模型[J]. 西安交通大学学报，2007，41(4)：426-429

[5] 赵旭峰，钱存华，Nakagawa T. 带有周期检测性的软件预防性再生策略[J]. 计算机科学，2009，36(8)：158-160，195

[6] Xie W, Hong Y, Trivedi K. Software Rejuvenation Policies for Cluster Systems under Varying Workload[C]// Proc. 10th Int'l Pacific Rim Dependable Computing Symp(PRDC 2004). Papeete, Tahiti, 2004: 122-129

[7] 赵天海，齐勇，沈钧毅，等. 应用服务器多态老化模型和最优再生策略研究[J]. 系统仿真学报，2007，19(8)：1705-1709

[8] Thein T, Park J S. Availability analysis of application servers using software rejuvenation and virtualization[J]. Journal of Computer Science and Technology, 2009, 24(2): 339-346

[9] Silva L M, Alonso J, Torres J, et al. Using Virtualization to Improve Software Rejuvenation[J]. IEEE Transactions on Computers, 2009, 58(11): 1525-1538

[10] You jing, Sun Yu-qiang, Wang Hong-yuan. The relationship research between usage of resource and performance of computer system[C]// 2009 WRI World Congress on Software Engineering. 2009, 3: 451-455

[11] Ferreira C. Gene expression programming: Mathematical modeling by an artificial intelligence(2nd Edition)[M]. Berlin: Springer-Verlag, 2006

[12] Jing Y, Lan Z, Hongyuan W, et al. JMeter-based Aging Simulation of Computing System[C]// Proc. Computer, Mechatronics, Control and Electronic Engineering(CMCE2010). Changchun, China, 2010: 282-285

(上接第 247 页)

最后，根据记录的实验结果数据，利用软件工具统计出每种方法完成最短路径搜索所需要搜索的节点总数目，并根据式(1)得到两种方法的搜索效率，将所有的实验结果列表比较，并根据实验结果进行分析。

表 5 所列为实验结果数据对比表。

表 5 实验结果数据对比表

算法	搜索节点总数	搜索效率
传统算法	63	87.7%
改进算法	17	96.1%

由上述实验结果对比分析可知，传统的 Dijkstra 算法在搜索最短路径时需要逐一搜索所有节点，在此实验中完成最短路径的搜索需要搜索的节点总数目为 63 个，可见巨大的搜索量增加了计算压力，使得计算量很大且系统的存储压力较大。由式(1)的搜索效率计算公式可知，搜索的节点数目与搜索效率成反比，因此传统方法逐一搜索的方式直接造成最短路径的搜索效率不高，搜索效率仅为 87.7%，不能满足人们对电子导航系统的实时性要求。而考虑 GIS 空间特性，提出了方向优先的搜索策略，其通过计算连线方向夹角，并选取夹角较小的节点方式去除许多无效的搜索，缩小了搜索的范围，进行此次最短路径的搜索只需要搜索 17 个节点便可完成，大大减少了计算量，使得最终的最短路径搜索效率提高到 96.1%，取得了满意的结果，可见改进的 Dijkstra 算法能够有效提高最短路径的搜索效率，满足电子导航系统的要求。

结束语 提出了改进的 Dijkstra 算法并应用于 GIS 导航中。改进 Dijkstra 算法考虑城市交通 GIS 网络的空间分布特性，提出方向优先搜索策略来缩小节点的搜索范围，大大减少了搜索计算工作量，且占用的存储空间较少，提高了系统的工作效率，从而提高了 GIS 中最短路径的搜索效率。实验表明，这种改进的 Dijkstra 算法能够保证最短路径搜索的效率，有效满足人们对电子导航效率的要求，具有较高的使用价值。

参 考 文 献

[1] 王卫强，孙强. 求图中受顶点数限制的所有最短路径的算法[J]. 计算机工程与设计，2008，29(7)：1754-1757

[2] Han Y. A note of an $O(n^3/\log n)$ time algorithm for all pairs shortest paths[J]. Information Processing Letters, 2008, 105(08): 114-116

[3] 陈益富，卢潇，丁豪杰. 对 Dijkstra 算法的优化策略研究[J]. 计算机技术与发展，2006，16(9)：73-75

[4] 叶品勇，都洪基，沈曦. Dijkstra 算法在最佳抢修路径计算中的应用[J]. 继电器，2006，34(12)：39-41

[5] 王凌，段江涛，王保保. GIS 中最短路径的算法研究与仿真[J]. 计算机仿真，2005，22(1)：117-120

[6] Pallotlino S. Shortest path methods: comp lexity, interrelation and ne}u p ropositions[J]. Networks, 2008, 14: 257-267

[7] Deo N, Pang C Y. Shortest2path algorithms: taxonomy and annotation[J]. Networks, 2007, 14: 275-323

[8] 库向阳，史经俭，罗晓霞. 栅格数据模型中附有条件的最短路径算[J]. 计算机应用，2008，28(4)：856-859

[9] 王红梅，胡明. 基于直接/间接邻边概念的最短路径算法[J]. 计算机应用，2010，30(5)：1297-1299

[10] 李小鹏，郁滨，李亚敏. 基于 MapX 最短路径搜索算法研究[J]. 计算机工程与设计，2009，30(22)：5225-5228

[11] 魏秉铎，李林，胡峰. 普适环境下的智能地理信息服务研究[J]. 重庆邮电大学学报：自然科学版，2009，21(5)：667-671