# 基于自律计算的多数据点阈值检测方法

刘文洁 李战怀
(西北工业大学计算机学院 西安 710072)

**摘 要** 自律计算在实施过程中通常采用阈值来标记性能故障。阈值是系统的性能计数器的边界值,系统自动检测这些边界值。若某个设备的性能指标达到这些边界值,则表示系统的某台设备出现了性能故障,自律系统的自主管理器便会主动采取策略来修复性能故障,从而保证系统一直处于稳定的工作状态。目前,自律计算领域欠缺对于系统性能故障的研究。在研究自律计算自我监视的基础上,提出了一种多数据点的阈值检测方法,即通过多次检测阈值的越界和恢复来判断系统是否出现性能故障,以保证检测的有效性,为系统采取策略修复故障提供有利的依据。

**关键词** 自律计算,性能故障,阈值检测,自我监视,策略
**中图法分类号** TP311 **文献标识码** A

## Multiple Data Threshold Detecting Method Based on Autonomic Computing

LIU Wen-jie LI Zhan-huai
(School of Computer, Northwestern Polytechnical University, Xi'an 710072, China)

**Abstract** Autonomic computing system often uses threshold to mark performance faults. Threshold is the boundary value of performance counter, which will be detected automatically by the system. When the performance counter of one of the devices reaches the threshold, it is considered that performance fault occurs on one of the devices. Then the autonomic manager will actively select policies to recover these faults, which therefore makes the whole system to maintain normal state. Nowadays, it lacks of research on system performance faults in autonomic computing field. On the basis of studying the self-monitoring of autonomic computing system, this paper proposed a multiple data threshold detecting method, by detecting the exceeding and recovering of the threshold in multiple times, system can effectively judge whether the performance fault occurs. This method assures the detecting accuracy, which provides the effective basis to recover fault for autonomic computing system.

**Keywords** Autonomic computing, Performance faulting, Threshold detecting, Self-monitoring, Policies

## 1 引言

自律计算的目的是建立类似于人的神经系统的信息系统,参照人的神经系统的自我调节机制,以现有的理论和技术(面向服务的计算、自适应控制理论、优化理论、基于策略的理论、多主体理论等)为基础,构建具有自我管理能力的IT系统,从而降低管理的复杂性[1]。

自律计算作为新兴的研究领域,自2001年提出其概念后,众多学者对其概念进行了扩展和补充,明确了研究方向。Kephart 2003年提出了MAPE(监视-分析-规划-执行)控制作为自主元素的自适应控制机制,为自律计算的研究奠定了初步的研究方向和路线。其指出,自律监视是自主元素的一个关键特征,自主元素要能够监视自身状态的变化并记录状态信息,要能够在故障发生时和自主管理器进行数据交换,从而达到自适应和自优化的目的。常用的实现方法是基于统计学的方法和基于采用样的测试方法[2]。

Roy Sterritt给出了一种自律计算的3层体系结构,采样有效的事件管理机制在自主元素之间传递信息,提出了"心跳/脉搏监视"和"自主信号通道"的概念[3]。其指出,建立自律系统的关键就是要在自律系统和自主行为之间建立一种桥梁。

Ada等在文献[4]中提出了一种基于组件的自动性能优化框架。整个框架包含一个单独的监控和诊断模块、被应用适应模块,能够根据上下文自动地选择最优的组件实现。这些模块能够根据系统关注热点优化系统开销,从而可以长时间运行。

文献[5]提出了一种自律资源管理框架来实现虚拟服务中心的自我管理行为。其采用虚拟服务器,细化资源共享方式来改善服务中心资源分配方式的灵活性,提高工作效率。本文提出了采用可能性分析模型的非线性优化技术来实现框架,以实验的方式证明了框架对于自律资源管理在性能上的提高,但该框架同样尚未应用于实际的自律资源管理中。

IBM多伦多实验室的Marin Litoiu提出了一种自律计算系统的性能分析方法[6]。该方法利用性能评估以及线性和非

到稿日期:2010-06-07 返修日期:2010-08-30 本文受国家自然科学基金重大国际(地区)合作项目(60720106001)资助。
刘文洁(1976-),女,博士,主要研究方向为软件理论、自律计算、高可用性系统,E-mail:liuwenjie@nwpu.edu.cn;李战怀(1961-),男,教授,博士生导师,主要研究方向为数据库与知识库系统、存储区域网络、软件工程。

线性的编程模型，通过观察工作负载的负载度来计算分布式系统性能矩阵的边界值。系统采用队列网络模型建模(QNM)，模型数据能够实时地采集和过滤。自律管理器通过性能边界值，可以调节会话的数量，计算可用节点数。

廖备水等人在阐明了自律计算基本概念的基础上，提出了一个自律计算模型，刻画了自主元素和自律计算系统的基本工作机制和原理，提出了基于知识模型和数学模型的自律计算系统[7]。但基于模型的方法只能进行定性分析和逻辑推理，对如吞吐率、CPU 和存储器的使用率等系统性能的管理不能进行定量描述；数学模型中的问题求解知识可以构建性能模型。

自律系统所管理的资源是多种多样的，各种跨系统跨版本的操作系统都可能成为管理对象。因此，解决分布式环境中混合系统的性能监视是一个关键问题。文献[9]针对分布式环境下的异构操作系统，设计了一个单独的性能监视工具，以完成对自律系统所管理的多种异构设备的统一性能监视，并能够对统计后的性能数据进行统一的分析和比较，实现有效的自我监视。

虽然上述研究对于自律系统的自律监视提供了研究思路，但现有的方法中，均未提到如何处理性能故障的边界值问题。现有方法中主要采用的是单值检测，即当发现设备性能值超过边界值一次，则立刻认为性能故障发生，导致偶然的性能指标越界也要通知自律系统，采取策略修复，即使一次越界后性能指标立刻恢复正常的状况依然如此，这严重影响了自主管理器的工作效率。

文献[8]提出了一种自适应阈值的网络流量异常检测算法，其核心思想是建立阈值的置信区间，通过判断采样点是否落在区间内来判断异常是否发生。但该方法仅适用于短时间内异常的监测，不适合长时间性能数据分析；此外，检测对象依然是单点，对于偶然的异常点也会判断为系统异常发生，在自律系统中会发生通报事件，影响系统性能，因此也不适合于自律系统的性能检测。

本文在文献[9]的基础上，提出了一种多数据点的阈值检测方法。即在一个时间段内对设备的性能指标进行连续的阈值检测，当整个时间段内的多次检测值均超过边界值时，才认为性能故障发生，从而避免了无效决策所导致的系统开销，提高了自律系统的决策效率。

## 2 监视流程分析

在自律计算系统中，系统故障的监视通过两种途径来完成：一种是针对硬件故障(CPU、网卡、内存损坏等)的监视，主要采取在被管设备上安装 Agent 的方式来完成。Agent 作为系统的守护进程，定时向自律管理器报告系统状态，在故障发生时将故障部件及类型通过事件发送给自律系统，由自律系统选择合适的策略修复故障。另一种是对系统性能故障(CPU 超负荷、内存容量不足等)的监视，主要依靠操作系统的性能监视器来完成。但是自律系统是分布式环境下的新型计算模式，所管理的设备是异构设备，要统一监视性能故障，必须依靠单独的性能监视系统来完成。文献[7]中的监视系统针对异构系统设计，将监视器设计为独立的系统，与自律系统互相通信，具有参考意义。以此为基础，我们可以对自律系统和监视器之间的通信流程进行分析，从而得出阈值的故障通知过程。

假设自律系统为 AC，监视器为 MT，则监视流程如下。

初始状态：自律系统 AC 中包含 $n(0<n<N, N$ 为自然数)个资源{R1,R2,R3,…,R$n$}，监视工具 MT 中没有任何监视被管资源{Φ}，则：

Step1　MT 向 AC 获取被管资源。获取后状态

AC:{R1,R2,R3,…,R$n$}，MT:{R1,R2,R3,…,R$n$}

此时，MT 开始对资源 R1,R2,…,R$n$ 进行性能监视。

Step2　AC 进行资源调整，删除被管资源 R1，AC，MT 未同步，则

AC:{R2,R3,…,R$n$}，MT:{R1,R2,R3,…,R$n$}

此时，MT 依然监视已删除资源 R1。

Step3　AC 和 MT 进行资源定时同步，同步后状态

AC:{R2,R3,…,R$n$}，MT:{R2,R3,…,R$n$}

此时，MT 不再监视已删除资源 R1。

Step4　MT 发现资源 R3 的性能指标超过阈值，向 AC 发通知，AC 执行策略追加资源 R$m$，MT 未同步，则

AC:{R2,R3,R$m$,…,R$n$}，MT:{R2,R3,…,R$n$}

此时，MT 尚未监视资源 R$m$。

Step5　AC 和 MT 进行资源定时同步，同步后状态

AC:{R2,R3,R$m$,…,R$n$}，MT:{R2,R3,R$m$,…,R$n$}

此时，MT 开始对资源 R$m$ 进行监视。

上述 Step1 是启动过程，是标识 AC 与 MT 协作的开始。若 AC 无资源调整，则双方资源集合均无变化；若 AC 由于策略执行导致资源构成发生变化，则 MT 通过和 AC 的定时同步可以得到最新的资源列表，从而保证监视资源的一致性。MT 如果发现 AC 的任何资源性能指标超过阈值，则以通知的方式报告 AC，AC 采取预定策略调整资源。因此，Step2 到 Step5 是反复执行的过程，说明了 AC 在 MT 的协助下不断地进行自我配置、自我修复的自动化管理。

Step4 是阈值监视与通知的过程。在此流程中，当监视器发现资源阈值超过一次时，则立刻向自律系统发送性能故障通知，从而促使自律系统选择策略来修复该故障，这也正是问题所在。由于没有区分偶然的阈值超标和连续的阈值超标状况，必然会导致自律系统要花费代价来处理偶然出现的伪故障，降低了系统的工作效率，在真正故障发生时不能及时有效地加以处理。因此，在处理自律系统的性能监视问题时，对伪故障进行过滤是提高系统工作效率的一个重要因素。

## 3 多数据点阈值检测方法

性能数据是监视系统定时地从被管资源中得到的基础数据，例如每分钟从设备上获取其 CPU 使用率的数值，根据数值的大小来判断设备的工作状况。由于异构设备 OS 的不同，所监视的性能指标也有所差别，但大多数的 OS 具有一些相同的性能指标，如% CPU Usage，% Disk Transfer Rate 等。这些共同的性能指标使我们可以对异构系统的性能状况进行统一的检测和分析。由于性能数据可以是基础数据，也可以是统计数据，例如按照每分钟统计一个数据点，一个小时可以获得 60 个数据点，按照每 20 分钟的时间间隔对该小时内的数据进行平均值的统计，可以得到 3 个点，由此可以减低分析数据量，在长的时间间隔内判断设备性能状况的趋势更加容易。但是，无论是基础数据还是统计数据，判断阈值超标

的检测方法是一致的。在此，我们以基础数据作为研究对象。

### 3.1 阈值区间的定义

阈值是性能计数器的界限值，标识被管资源的性能指标的正常或异常状况。阈值具有上限边界值 $T_{uppererror}$ 和下限边界值 $T_{lowererror}$，且 $T_{uppererror} > T_{lowererror}$。

假设性能计数器为 $P$，其值 $value$ 满足如下关系：

$\{value > T_{uppererror}\} \vee \{value < T_{lowererror}\}$

则认为性能计数器 $P$ 发生性能异常，$value$ 处于阈值区间，如图 1 所示。

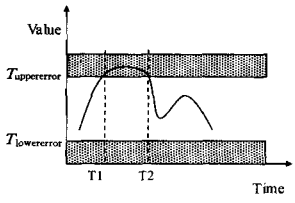


图 1 阈值区间工作示意图

在图 1 中，$T_{uppererror}$ 以上部分和 $T_{lowererror}$ 以下部分为阈值区间，若性能数据在某个时间段内的值落入该区间，则认为设备发生了性能故障。在图 1 中，时间区间[$T1$，$T2$]为故障时间段。

假如区间[$T1$，$T2$]内只有一个数据点，则阈值检测方法为单点检测法。该方法只判断一个点的越界，实现较为简单，但判断结果不准确。在实际应用中，性能数据变化较快，很可能刚越界就恢复，或者数据点在越界和恢复之间反复出现，监视器频繁通知，会导致自律管理器无法应答，而决策执行又会花费较长的时间，最终导致系统瘫痪。因此，采取更为有效的多数据点检测阈值是一个很好的解决办法。

### 3.2 阈值超过状况的判断

多数据点检测方法的依据是在发现第一个数据越界后，连续判断其后的若干数据点。如果多个数据点的值均超过阈值或者超过阈值的数据点个数大于被统计数据点个数的一半（具体个数根据实际应用而定），则认为性能故障发生。

假设采样数据点为 5 个，当发现第一个数据点超过阈值后，我们连续判断随后的 4 个数据点是否越界。如果越界数据点个数超过 3 个，则认为性能故障发生，通知自律系统。当判断的数据个数不足 5 时，不作任何通知，继续判断后续数据点。当判断个数等于 5 时，如果无性能故障发生，则计数器清零，重新开始下一组数据点计数。图 2 显示了多数据点阈值检测方法的判断流程。

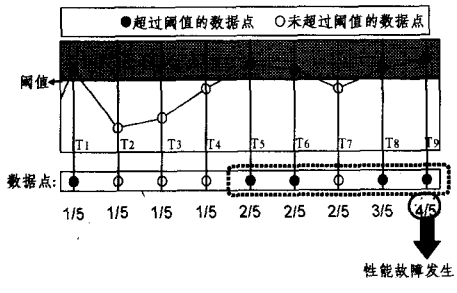


图 2 多数据点阈值检测方法示意图

在图 2 中，以[$T1$，$T9$]的时间间隔作为采样区间，当发现在 $T1$ 时刻有数据点越界时，将其作为采样的第一个数据点，数据点计数器和超阈值数据点计数器分别加 1。随后连续统计 $T2-T5$ 数据点的越界情况。在此区间内，我们发现超过阈值的数据点只有 2 个，没有超过被统计数据点的一半（2.5），因此认为该区间内性能故障未发生，系统工作正常。但是由于在 $T5$ 时刻再次发现了超过阈值的数据点，因此该时刻将会作为下一个统计区间的开始，即 $T5$ 时刻既是上一个统计区间的结束，也是下一个统计区间的起始时刻。按照同样的方法，对[$T5$，$T9$]时刻的数据点进行统计，发现该区间内有 4 个数据点超过阈值，大于被统计数据点的一半，因此认为该区间内性能故障发生，通知自律系统采取策略来修复故障。

本方法的优点是，只有在发现一个数据点超过阈值后才进行性能故障检测，并且在多次阈值超过发生的情况下才通报，因此保证了性能故障发生判断的准确性。

### 3.3 阈值恢复状况的判断

性能故障的发生会导致系统不能正常工作，而系统是否从故障中恢复的判断也影响自律系统的决策。例如自律管理器需要选择工作正常的设备来替换故障设备时，就需要获取其性能状况。如果一台设备已经从故障中恢复，则可以充当预备机器来替换故障机器。因此，能否准确判断故障恢复状况，也是影响系统工作效率的一个主要因素。

与阈值超过判断相对应，阈值恢复状况的判断同样采用多数据点检测方法，以避免性能故障的假恢复，导致系统决策错误。阈值恢复的判断流程如图 3 所示。

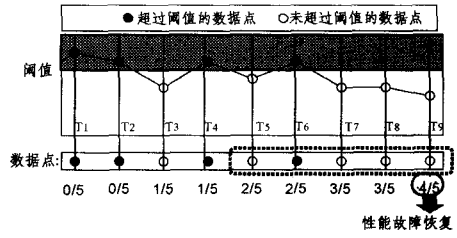


图 3 多数据点阈值恢复示意图

在图 3 中，依旧采用[$T1$，$T9$]作为采样区间，采样数据点也是 5 个。当 5 个数据点中有 3 个以上的点从阈值状态恢复时，则认为设备性能故障已经恢复正常。恢复数据点恢复的判断依然采用两个计数器，一个统计数据点个数，一个统计已恢复的数据点个数。在 $T1$ 和 $T2$ 时刻，由于系统性能还处于超阈值状态，因此恢复数据点的个数为 0；在 $T3$ 时刻，第一个恢复点出现，因此阈值恢复计数器加 1。$T5$ 时刻，第二个恢复点出现，同时第一个时间段数据点统计结束，在整个区间内，有 2 个数据点恢复，没有超过总数据点个数的一半，因此认为设备依然处于故障状态。$T5$ 时刻是数据点的恢复状况，因此作为下一个统计区间的开始，两个计数器均加 1。在区间[$T5$，$T9$]内，共有 4 个数据点恢复到正常状态，因此认为设备性能故障已经恢复。

多数据点恢复的检测方法同样提高了故障恢复状况判读的准确性，因此在实际应用中，能够提高自律系统决策的准确度。

## 4 实验与分析

针对 ESX2.5 服务器，我们对 CPU 利用率按照每分钟 1 次来采集 10 个数据点，并利用多数据点阈值检测方法进行判断，得到了实际通报给自律计算系统的阈值个数，如表 1 所列。

（下转第 168 页）

[2] Misra J, Gries D. Finding repeated elements[J]. Science of Computer Programming, 1982, 2: 143-152
[3] 王伟平，李建中，张冬冬，等. 一种有效的挖掘数据流近似频繁项算法[J]. 软件学报，2007，8(4)
[4] Manku G S, Motwani R. Approximate frequency counts over data streams[C]// International Conference on Knowledge Discovery and Data Mining. 2006
[5] Cormode G, Muthukrishnan S. What's hot and what's not: Tracking most frequent items dynamically[C]// ACM Symposium on Principles of Database Systems. 2003
[6] Yu J X, Chong Z, Lu H, et al. False positive or false negative: Mining frequent itemsets from high speed transactional data streams[C]// International Conference of Very Large Databases. 2004
[7] Teng W, Chen M, Yu P S. A regression-based temporal pattern mining scheme for data streams[C]// International conference of Very large Databases. 2003
[8] Sun X, Orlowska M E, Li X. Finding frequent itemsets in high-speed data streams[C]// SIAM International Conference on Data Mining. 2006
[9] Chang J H, Lee W S. estWin: Adaptively monitoring the recent change of frequent itemsets over online data streams[C]// International Conference on Information and Knowledge Management. 2003
[10] Koh J, Shin S. An approximate approach for mining recently frequent itemsets from data streams[C]// International Conference on Data Warehousing and Knowledge Discovery. 2006
[11] Mozafari B, Thakkar H, Zaniolo C. Verifying and mining frequent patterns from large windows over data streams[C]// International Conference on Data Engineering. 2008
[12] Chang J H, Lee W S. Finding recent frequent itemsets adaptively over online data streams [C] // International Conference on Koowledge Discovery and Data Mining. 2003
[13] Giannella C, Han J, Pei J, et al. Mining frequent patterns in data streams at multiple time granularities[M]// Kargupta H, et al., eds. Next Generation Data Mining. AAAI/MIT, 2003
[14] Li H, Lee S, Shan M. An efficient algorithm for mining frequent itemsets over the entire history of data streams[C]// International Workshop on Frequent Itemset Ming Implementations. 2004
[15] 刘学军，徐宏炳，董逸生，等. 挖掘数据流中的频繁模式[J]. 计算机研究与发展，2005，12
[16] Jin R, Agrawal G. An algorithm for in-core frequent itemset mining on streaming data[C]// IEEE International Conference on Data Mining. 2005

(上接第 134 页)

假设阈值上限是 26，按照单点检测方法，采样点 2，5，6，7 均超过阈值，则通报次数为 4，自律管理器 AM 在 10 分钟内将收到 4 次来自同一台服务器的性能故障通报，并采取策略来修复故障，导致同样的策略执行 4 次。而第一个策略执行完毕后，可能服务器性能值已经恢复正常，因此其他策略执行属于无效操作，加大了系统开销。

表 1 ESX2.5 Server 的 CPU Usage%

| 个数 | 采样值 |
|---|---|
| 1 | 23.92613 |
| 2 | 41.1516 |
| 3 | 25.37086 |
| 4 | 22.78824 |
| 5 | 26.58733 |
| 6 | 27.66686 |
| 7 | 26.18363 |
| 8 | 23.53114 |
| 9 | 23.80006 |
| 10 | 23.17188 |

按照多数据点阈值检测方法，假设阈值上限相同，连续 3 个采样点越界才通报，则数据点 2 属于偶然越界值，不通报。只有在 5，6，7 这 3 个数据点判断完毕后才认为性能故障发生，向自律系统通报一次，策略执行一次。与单点判断相比，系统无效操作减少了 75%。

**结束语** “监视，分析，计划，执行”是自律系统的 4 个主要特征。要进行自主决策，首先要能够自己发现问题，因此“自我监视”应该是自律计算中要研究的核心问题。本文主要针对自律计算中的性能监视故障，提出了一种多数据点的阈值检测和恢复方法，其目的是改善单点检测所存在的准确度不高、影响自律系统策略执行效率的问题。多数据点检测方法能够显著改善检测的准确度，从而提高系统的决策效率。本方法已经应用到了自律监视系统的设计中，所开发的产品也已经在国外投入了使用。实践证明，采用本方法的系统具有较高的决策效率。

## 参 考 文 献

[1] IBM Company. An architectural blueprint for autonomic computing [EB/OL]. Autonomic Computing White Paper. Third Edition. http://www.ibm.com/autonomic, June 2005
[2] Kephart J, Chess D. The Vision of Autonomic Computing [M]. IEEE Computer Society, 2003: 41-59
[3] Sterritt R, Curran E P, Song H. HACKER: Human And Computer Knowledge discovered Event Rules for Telecommunications Fault Management[C]// Proc. IEEE Int. Conf. Systems Man & Cybernetics. Oct. 2002
[4] Diaconescu A, Mos A, Murphy J. Automatic Performance Management in Component Based Software Systems[C]// Proceedings of the International Conference on Autonomic Computing (ICAC'04). 2004
[5] Wang Xiao-ying, Lan Dong-jun, Fang Xing, et al. A resource management framework for multi-tier service delivery in autonomic virtualized environments[C]// Network Operations and Management Symposium(NOMS 2008). IEEE, April 2008: 310-316
[6] Marin L. A Performance Analysis Method for Autonomic Computing Systems [J]. ACM Transactions on Autonomous and Adaptive Systems, 2007, 2(1)
[7] 廖备水，李石坚，姚远，等. 自主计算概念模型和实现方法[J]. 软件学报，2008，19(4)：779-802
[8] 曹敏，等. 基于自适应阈值的网络流量异常检测算法[J]. 计算机工程，2009，35(19)：164-166
[9] 刘文洁，李战怀. 一种基于自律计算的性能监视工具[J]. 微电子学与计算机，2008，25(3)：130-133