基于自律计算的故障监视机制研究与设计

刘文洁[1] 李战怀[1] 周云涛[2]

(西北工业大学计算机学院 西安 710072)[1] (西北工业大学实验室与设备处 西安 710072)[2]

摘 要 自律计算是分布式异构环境下进行资源自动化管理的有效技术。其目的是通过系统的自我监视,主动发现硬件故障和软件故障,并采用策略技术加以修复,完成系统的自我管理。因此,故障监视是自律计算中较为重要的一个研究方向,但目前尚欠缺有效可行的方法来完成自律系统的故障监视。提出了一种分布式异构环境下基于事件分类的方法来设计自律计算系统故障监视机制,以统一监视管理异构资源的故障,并与自律系统互联通报,激活相应的策略来修复故障,为自律系统的自我修复提供依据。

关键词 自律计算,故障监视,分布式计算,自我修复,事件分类

中图法分类号 TP311 **文献标识码** A

Research and Design of Fault Monitoring Mechanism Based on Autonomic Computing

LIU Wen-jie[1] LI Zhan-huai[1] ZHOU Yun-tao[2]

(Institute of Computer, Northwestern Polytechnical University, Xi'an 710072, China)[1]

(Institute of Lab and Equipment, Northwestern Polytechnical University, Xi'an 710072, China)[2]

Abstract Autonomic Computing is an effective technique to achieve system self-management in distributed heterogonous computing environment. Its aim is to automatically discover hardware fault and software fault and recover the fault with policy technology, which then realizes the system self management. Therefore, fault monitoring is an important research direction. This paper proposed an event classification method to design the fault monitoring of autonomic computing system. This method can monitor all the resources in heterogonous environment and report the fault to AC system, and then activate the policy to recover system fault, which provides the basis for system self recovery.

Keywords Autonomic computing, Fault monitoring, Distributed computing, Self-recovery, Event classification

1 引言

自律计算指的是系统像生物一样能够自行控制自身状况,使信息系统本身具备维持整个系统最佳工作状态的功能。系统可以根据变化的和不可预测的条件,完成自身的设置或重新设置[1]。该概念由IBM副总裁阿兰·卡耐克于2001年10月15日提出,其中指出,通过自律计算,机器可以自我监控、自我配置、自我优化和自我恢复[2]。

国际国内有越来越多的大学对与自主计算相关的领域进行了研究。但是研究的焦点大多是采用何种方法来使得IT系统具有自我管理特征,即达到自我配置、自我优化、自我修复和自我保护,而对于自律系统的自我监视方面的研究非常少。比较典型的代表就是IBM在2003年发布的IBM Tivoli Monitoring系统,用于自律系统的性能监视,但是没有提供异构系统的性能分析和比较,因此在异构环境下应用受到限制。其次较有代表性的就是OGSA(open grid services architecture)提出的Globus Heartbeat Monitor (HBM)[3],用于组件的Heartbeat来自我发现故障和错误并进行报告,具有一定的应用价值,但不能统一处理异构资源故障。其他的研究包括ROY STERRITT在文献[4]中提出的采用事件管理机制来实现故障检测;Yoshihiro Tohma在文献[5]中提出的在自律计算中加入基于服务的容错机制等,它们对该领域的研究都有一定的启发,但是仅限于理论,还欠缺工程化的应用方法。

目前,从知识模型的角度建立的实际自律计算系统主要基于下列技术:Agent技术、Web服务和语义Web技术。其中,Agent又称为智能主体或智能代理,具有反应性、自治性和社会性等特点,能够感知环境,做出反应,已被普遍认为是支持大规模、开放和分布的信息系统实现动态服务集成和协同工作的关键技术[6]。自律计算系统中,故障的检测和通知也主要依靠Agent来解决。故障发生时,被管资源的Agent首先自发地检测出故障发生点以及故障类型,然后向自律计算系统发出通知,自律计算系统的自主管理器AM通过策略库的决策挑出适合处理该故障的策略并执行,从而自动修复故障,达到整个系统的自我管理。但是,由于Agent主要集成于被管理资源上,而且不同资源的Agent的实现主要依赖于

到稿日期:2009-09-28 返修日期:2009-12-14

刘文洁(1976—),女,博士生,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;周云涛(1978—),男,硕士,助理研究员,主要研究方向为控制工程。

被管资源的硬件类型，因此其灵活性较差，不能集中处理自律计算系统中的所有被管资源的故障，导致自律计算系统的故障处理效率非常低。

本文在对自律计算中的故障进行分析的基础上，提出了一种基于事件分类的故障处理机制，它将异构资源中的硬件故障统一处理和通知，简化了故障处理流程，提高了故障发现效率，为自律计算中的故障监视提供了有效的设计方法。

2 基于 Agent 的自律计算系统分析

在自律计算系统中，要完成被管理资源的自配置、自修复、自优化、自保护，必须时刻了解被管资源的状态，例如系统运转是否正常，系统负荷是否过高或过低，硬件工作是否正常。一般来说，将资源故障分为两类：硬件故障和软件故障。硬件故障包括系统宕机、CPU、网卡或磁盘损坏、OS 无法启动等等。这些属于不可修复错误，必须通过硬件替换或资源替换来完成。软件故障包括 CPU 负荷过高、磁盘容量不足、磁盘传输率过低、某个软件应用无法正常运行等，这些故障属于系统性能问题，可以实时检测，在性能指标达到某个界限值(阈值)之前就采取预防措施，从而改善系统性能问题，解决软件故障。关于自律计算的性能监视在文献[7]中提出了完整的解决方案，而硬件故障监视主要是依靠资源的 Agent 来完成。

基于 Agent 来监视和发现故障的自律计算体系结构大多采用图 1 所示的体系结构。

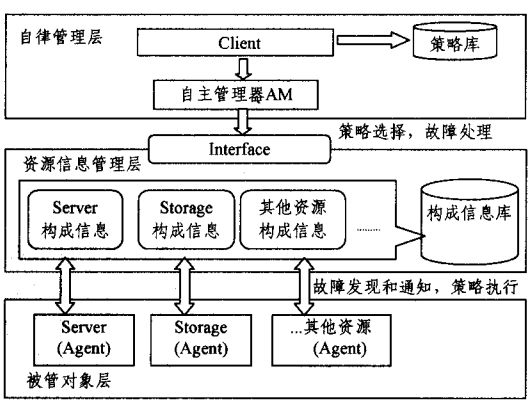

图 1 基于 Agent 的自律计算系统体系结构

从图 1 可以看出，自律计算系统主要分为 3 层，自律管理层、资源信息管理层和被管对象层。自律管理层是系统核心，由自主管理器 AM 来完成与被管资源的交互，负责处理故障，策略选择。资源信息管理层是接口层，负责收集实际的资源信息并传递给自律管理层，例如硬件类型、故障类型等。被管对象层在是实际的硬件资源，每个资源上安装 Agent，用来检测自身的故障类型并通知自律管理层。实际上自律管理层是直接管理所有的硬件资源，并针对每个硬件资源的类型来处理故障，因此决策库的数量将会相当大，AM 进行决策时会花费较长的时间，故障处理效率较低，这也是现有的大多数自律计算系统运行速度较慢。自我管理能力较低的原因。自律计算系统主要运行在分布式环境下，在管理各种异构资源设备时，若自我管理的效率低于人工管理效率，则自律计算就失去了存在的意义。因此，提高故障处理效率是提高自律系统自我管理能力的关键。

从上述分析可以看出，导致故障处理效率低的关键是故障类型没有进行有效的分类和过滤，导致不同的被管资源不能适用相同的策略。例如，资源 A 和资源 B 都发生 CPU 损坏的故障时，自律管理器需要给出两个不同的策略来处理。因此，我们可以设计一种故障分类和过滤机制，在 AM 接受所有 Agent 传来的故障通知之前，对所有的故障进行分类，减少故障类型，过滤掉可忽视的通知事件，从而减少自主决策层的负担，提高决策效率。

3 基于事件分类的故障监视机制

Roy Sterritt 总结了自律计算系统的故障发生规律指出，信息系统中的故障可以分为 3 类：Failure，Fault，Error。它们的相互影响遵循 Randell 基础链[8]：

…－＞ Failure－＞ Fault－＞ Error－＞ Failure－＞ Fault…

可以进一步地抽象为：

…－＞ Event－＞ Cause－＞ State－＞ Event－＞ Cause…

在系统运转过程中，Failure 的发生总是在操作过程中(Cause)遇到 Fault 而导致的，因为执行中发生了 Error(State)。要避免系统出现 Failure，则必须建立有效的容错机制(Fault Tolerance)，从而发现故障并恢复系统状态。

从上述规律可以看出，在现有的自律计算系统中，故障的描述主要通过事件(Event)来完成，自律系统管理的设备越多，接受和处理的事件数量就越多。事件处理的效率直接影响了自律系统的自我管理能力，如何快速确定故障类型，采用适当的策略来处理故障，从而使系统达到正常状态，是自律系统设计的一个非常关键的问题。现有的基于单一事件的处理方式显然不能满足这种需求。现有的事件处理机制中主要存在如下的问题。

对于 AM 而言，确定故障发生点是非常重要的功能，因此，通过事件的一些描述信息来获取故障种类及发生设备必不可少，所以在事件类型中设计了 5 个字段来完成该功能。ResourceID 确定故障设备，因为在自律系统中，每个被管理设备都有唯一的标识符保存在构成信息数据库中，EventID 区分不同的故障种类，例如 CPU 故障为 001，磁盘故障为 002，网卡故障为 003…依次类推，可以确定故障种类。由于自律系统管理的资源为网络设备，不同的设备位于不同的网段中，通过 IPAddress 来区分。Description 字段是事件的附加内容，例如事件的发生时刻，假如同一设备在 10:00 和 10:01 分发送了两次相同的事件，这可以通过该字段区分，从而过滤掉其中一条事件。Priority 的设计是为了在多个事件发生时，确定故障重要程度的标识，设定值可以从 1 到 10，AM 可以根据该字段确认先处理哪个故障事件，从而优先处理关键错误。以此结构为依据，可以设计如下的事件处理流程。

3.1 事件处理流

自律系统中的故障监视功能是通过故障监视服务来完成的，每个管理设备上都装有 Agent，因此可以实时监控到设备所发生的故障，并将故障信息以事件的形式封装之后发送给故障监视服务。故障监视服务接收到各种故障事件后，按如下的流程进行事件处理。

Step1 根据 EventID 来判断事件类型，过滤重复和无效事件；

Step2　故障监视服务接收过滤后的事件，按照 Event_Type 格式将事件存入事件接收表；

Step3　故障监视服务按照先进先出的原则从事件接收表提取数据；

Step4　做成发送事件，存储到事件发送表；

Step5　Socket 初始化，与自律计算系统通信；

Step6　按照先进先出的原则逐次从事件发送表提取事件，发送给自律计算系统；

Step7　发送成功后，返回 Step1。

故障监视服务是自律系统的一个构成部分，完成对管理设备的故障监视和通知自律系统的功能。在现有的自律计算系统的设计中，故障的监视主要依赖于安装在管理设备上的 Agent，一旦发现故障，由 Agent 将故障事件直接发送给自律计算系统的自律管理器 AM，缺乏事件的过滤和存储过程，导致管理设备众多时，AM 的应答非常缓慢。而 Agent 由于没有收到应答，重复发送事件，又加重了 AM 的负担。因此，将故障监视作为独立的服务进程来设计，加入事件过滤机制是提高自律系统工作效率的有效途径。此外，将过滤后的有效事件存入事件接收表和事件发送表，使事件处理可以有序进行，并且可以单独提取这两个表的内容，以了解设备故障的主要类型，对发生故障多的设备采用策略替换或修复，为 AM 决策提供依据。而对比事件接收表和事件发送表的内容，则可以了解是否所有接收到的事件都已经发送给 AM，防止遗漏事件的发生。图 2 显示了故障监视服务的事件处理流程。

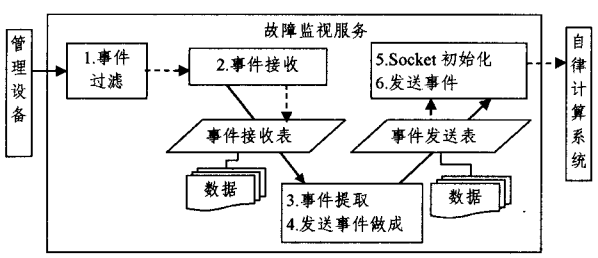

图 2　故障监视服务的事件处理流程

3.2　事件过滤处理

事件处理是故障监视服务的核心模块，能够有效提取故障事件。事件过滤的依据是故障类型和发生时刻，为了区分故障类型，为故障种类编码是一个很好的解决方法，不同类型的故障具有不同的编码，单一硬件和集群硬件可以通过编码高位的不同来区分。故障监视服务收到事件时，首先通过查故障类型的前缀码表得到故障类型，再根据事件中的时刻等消息判断事件的有效性，从而过滤出有效事件。下面列举部分故障类型的前缀码格式，如表 1 所列。

表 1　故障类型前缀码表

故障种类	前缀码
CPU 负荷故障	0x00000001
CPU 硬件故障	0x00000002
Memory 故障	0x00000003
服务器访问故障	0x00000009
Service 内部故障	0x0000000A
集群节点故障	0x20000001
集群 LAN 故障	0x20000002

在前缀码表中，不同种类的故障采用不同的前缀码来表示，前缀码是一个 8 位 16 进制数，以 0x 作为编码的开始。最高位区分硬件种类(单一设备或集群)，例如 CPU 故障最高位用 0 表示，集群节点故障用 2 表示，其他 7 位用来区分故障种类。在事件过滤时，首先判断高位得到发生故障的硬件种类，然后判断后 7 位得到故障类型，最后通过 ResoureID 来得到具体的故障设备，从而判断故障点。

如果同一故障设备在不同时刻多次发出相同故障种类的事件，则保留最初的事件，删除其他相同事件；如果设备相同而故障种类不同，则认为是有效事件，存入事件接收表。如果某一事件的故障类型在前缀码表中查不到或为空，则认为是无效事件。

通过以上的过滤流程，可以快速区分有效事件和无效事件，降低无效事件的发送，从而提高自律计算系统处理故障、修复系统的效率。

结束语　“自我监视”是自律计算系统的关键特征之一，通过自我监视，可以快速定位故障点，处理故障并修复系统，从而达到自律系统自我管理的目的。现有的研究大多是解决如何监视的问题，例如使用 Agent 技术或 OS 的监视工具等，对于如何提高自律系统的故障修复效率却没有更多的描述。本文针对自律系统设备众多、故障事件过多而导致自律系统处理效率低下的问题，提出了一种基于事件过滤的故障监视机制，通过对不同类型的故障建立前缀码来快速筛选有效事件，从而达到提高自律系统故障修复效率的目的。

本文所设计的故障监视机制已经应用到了自律系统的设计中，开发出的产品已经投入使用。实践证明，该方法可以明显地提高故障事件的处理速度和系统的工作效率。

参 考 文 献

[1] Horn P. Autonomic Computing: IBM 's Perspective on the State of Information Technology [OL]. IBM Corporation. http://www. research. ibm. com/autonomic Oct. 2001

[2] Mainsah E. Autonomic computing: the next area of computing Electronics & Communication [J]. Engineering Journal, February 2002

[3] The Globus Heartbeat Monitor Specification v1. 0[J]. http://www-fp. globus. org/ hbm/ heartbeat spec. html

[4] Sterritt R. Towards Autonomic Computing: Effective Event Management[J]. Computer Journal, IEEE Computer Society, January 2003

[5] Tohma Y. Incorporating Fault Tolerance into an Autonomic-Computing Environment[J]. IEEE Computer Society, 2004, 5(2)

[6] de Campos, de Barros G A L, de Souza A L B, et al. A Model for Designing Autonomic Components Guided by Condition-Action Policies[C]//Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE, April 2008: 343-350

[7] 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

[8] Sterritt R. Towards Autonomic Computing: Effective Event Mangagement[J]. Computer Journal, IEEE Computer Society, January 2003