

基于 WMI 的计算机监控系统的设计与实现

雷涛^{1,2}, 井鹏程³

(1. 山东省光纤传感技术重点实验室, 山东 济南 250014;

2. 山东省科学院 激光研究所, 山东 济南 250014;

3. 河南黄河河务局信息中心, 河南 郑州 450004)

摘要: 无人值守监控系统在各种行业得到了广泛的应用, 保证无人值守计算机的稳定运行对于监控系统具有重要的意义。WMI 是 Windows 平台性能管理的事实标准, 将 WMI 技术应用到无人值守计算机监控系统, 设计并实现一个对无人值守计算机软硬件信息以及运行信息进行监视, 对监视信息进行分析预警并对无人值守计算机实现远程控制的监控系统。系统无需安装客户端, 相对于传统 C/S 模式的监控系统具有配置灵活方便、稳定性高、节约计算机资源等明显优势。

关键词: WMI; 无人值守; 监测系统; .NET

中图分类号: TP302.1

文献标识码: A

文章编号: 1673-629X(2010)12-0232-04

Design and Implementation of Computer Monitoring System Based on WMI

LEI Tao^{1,2}, JING Peng-cheng³

(1. Shandong Province Key Laboratory of Optical Fiber Sensing Technology, Jinan 250014, China;

2. Institute of Laser, Shandong Academy of Science, Jinan 250014, China;

3. Information Center of Yellow River Bureau, Zhengzhou 450004, China)

Abstract: The unattended computer monitoring system has been widely applied, and ensuring the unattended computer to run steadily has important significance. WMI is a fact standard of performance management of Windows. Aims to apply WMI to unattended computer monitoring system for the purpose of designing a long-range control monitoring system which can monitor the hardware and software information of unattended computer, as well as the operating information, and accordingly analyze and early warn the monitoring information. The system which does not need to install client is more flexible, convenient and steady, and using fewer computer resources than the traditional C/S monitoring system.

Key words: WMI; unattended computer; monitoring system; .NET

0 引言

随着信息技术、传感技术以及自动控制等技术的发展, 无人值守监控系统逐渐成熟, 并在电力行业(如无人值守变电站)、电信行业(如无人值守机房、远程基站), 矿产业(如供电设备机房、生产作业面)、环境监控行业(如排污点设备机房)等行业中的一些地理位置比较偏僻而又要求设备或者系统具有长期高度稳定性的场得到广泛的应用。无人值守监控利用传感器、控制器以及视频监控设备对无人看管、自动运行的前端

设备的运行状态和安全情况进行实时数据查看和设备控制, 从而实现远程监控与管理目的^[1]。无人值守监控系统中一般都具有现场监控计算机, 现场监控计算机主要负责监测数据的处理、汇总和存储, 对监测数据分析处理, 对设备进行控制, 同时通过网络与监控中心进行通信, 向监控中心发送监测数据, 接收监测中心的控制命令。现场无人值守监控计算机是无人值守监控系统中极其重要的组成部分, 因此对现场监控计算机进行监控, 保证现场监控计算机的正常稳定运行具有重要的意义。

无人值守计算机的监控一般采用 C/S 模式, 监控中心通过客户端软件来监测和控制现场无人值守计算机^[2]。因此, 每台无人值守计算机都需要安装客户端软件, 工作比较繁琐。而且, 客户端软件必须实时运行, 如果因为某种原因退出客户端软件, 那么监控中心

收稿日期: 2010-04-16; 修回日期: 2010-07-18

基金项目: 国家中小企业创新基金资助项目(09C26213704520); 山东省重大专项基金资助项目(2008ZHZX1A0308)

作者简介: 雷涛(1982-), 男, 硕士研究生, 研究方向为软件工程、信息管理系统。

就无法管理该计算机。这种方法不仅增加了计算机的系统开销,而且对各种故障难以检测,其管理效率很难得到较大改善。文中采用 WMI^[3,4](Windows 管理规范:Windows Management Instrumentation)构建一个监控系统,无需客户端软件即可对无人值守计算机的各种资源和运行情况进行监控,确保计算机的正常平稳运行。

1 WMI 简介

WMI 是 Windows 平台对应用性能管理的一个事实标准,是对 DMTF 提出的基于 Web 的企业管理规范^[5]在 Windows 平台上的实现^[6,7]。CIM 是由 DMTF 提出的一个面向对象的抽象模型,该模型为被管理资源(如系统、设备、应用程序和网络)提供了丰富的且易于扩展的描述途径。WMI 就是使用 CIM 来描述 Windows 环境中的系统对象^[8]。通过 WMI 可以访问、配置、管理和监视几乎所有的 Windows 资源。

WMI 体系结构由托管资源、WMI 基础结构、使用者三层组成^[9],如图 1 所示。

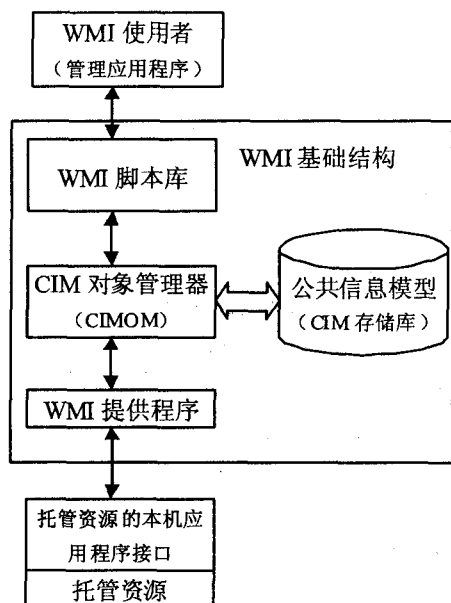


图 1 WMI 体系结构

1.1 托管资源

计算机中的逻辑或物理组件,可以使用 WMI 进行公开和管理^[10]。WMI 可以对计算机磁盘、外围设备等硬件资源,进程、事件日志、注册表设置、共享、安全性、服务等系统运行信息以及文件系统、WDM 设备驱动程序、Windows 安装程序等软件信息进行管理。WMI 与托管资源利用程序接口进行通信。

1.2 WMI 使用者

使用者是对计算机系统进行管理和控制的应用程序或者脚本。使用者仅仅需要知道计算机资源在

WMI 中对应的类就可以很方便地获取这些资源的信息并进行相应的控制。具体的信息获取过程以及控制过程被 WMI 隐藏起来,对 WMI 使用者来说是透明的。

1.3 WMI 基础结构

在 WMI 基础结构中 CIMOM 处理使用者和提供程序之间的交互^[11,12]。CIMOM 不仅仅提供使用者与 WMI 通信的接口,CIMOM 同时还向 WMI 基础结构提供注册、请求传送、安全性、远程访问、查询处理、事件处理等核心服务。CIMOM 从 CIM 存储库和 WMI 提供程序获得信息响应使用者的请求。CIMOM 利用从 CIM 存储库获得的信息,将请求传递到相应的 WMI 提供程序。上述过程是建立在面向对象的公共信息模型之上,与系统和应用程序独立,该模型还对计算机中托管资源的对象提供分层架构。“公共信息模型”是一个用于存储对象的对象数据库,包括用于访问和操作各种托管资源的静态类的定义以及实例。由 CIMOM 对 CIM 存储库进行管理。CIM 存储库相当于代理程序,CIMOM 通过 CIM 存储库检索可用的类,并确定一个 WMI 提供程序来对这些类提供实例。

“提供程序”相当于 CIMOM 与托管资源之间的一个代理程序。它的作用是利用系统提供的管理接口获取系统的管理信息。WMI 提供程序通过各种 COM 接口向 WMI 提供数据,并处理来自 WMI 使用者的请求生成事件通知。WMI 提供程序还将 CIM 存储库中定义和存储的对象类与管理信息和接口进行映射。

2 系统结构和系统功能

无人值守计算机监测系统利用 WMI 实现对各个计算机运行情况和资源利用情况等动态信息、计算机软硬件配置等静态信息的监视,根据监测信息实现对无人值守计算机的总体评价和预警。根据评价或者预警,管理人员可以对远程无人值守计算机进行相应的操作来确保计算机的正常平稳运行。同时,系统对每台无人值守计算机的相关信息存储和管理。整个系统的结构如图 2 所示。

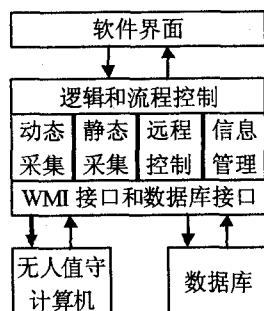


图 2 系统结构

对运行情况和资源利用情况等动态信息的监视主要是对无人值守计算机中运行的进程以及进程 CPU 占用率、CPU 总体利用率、硬盘空间剩余情况和内存利用率等信息进行监视和预警。计算机中这些动态信息是实时变化的,需要实时采集这些信息。为了保证信息采集的及时性以及程序的响应速度,需要对每台无人值守计算机分配一个线程来完成动态信息的采集。通过对 CPU、内存和硬盘的利用情况进行监视和分析使得操作人员可以了解无人值守计算机的运行情况,以便于进行相应的操作确保计算机的正常平稳运行。

对计算机软硬件配置等静态信息的监视主要是对每台无人值守计算机的操作系统、处理器和内存等硬件信息,计算机上服务的状态等不经常发生变化的信息进行监控。这些信息相对稳定,因此不需要实时地去获取这些信息数据,只有在操作人员需要查看这些信息的时候才通过 WMI 去获取这些信息。

通过对上述信息进行监测以及对监测信息的分析整理,操作人员可以对远程无人值守计算机进行相应的远程操作和控制,这些控制主要包括重启计算机、启动软件、终止进程、开停服务等。如果某个进程对 CPU 的占用率超过其允许值,系统将进行报警提示,操作人员可以终止本进程,以防止计算机出现死机等情况。

系统在运行的过程中需要使用无人值守计算机的 IP 地址、用户名和口令等相关信息,因此系统需要对这些信息进行存储和管理,提供对这些信息的添加、删除和修改等功能。当计算机的这些信息发生变化时,需要通过系统的信息管理功能对这些信息进行更改。

3 系统实现

3.1 实现平台

微软在 .NET 平台中对 WMI 进行支持,可以很方便地对系统进行开发。开发时首先引入 System. Management 命名空间,此命名空间为 WMI 提供支持,它提供的类对象为访问各种管理对象提供了面向对象的编程接口,使用该命名空间可以查询系统信息^[3]。本系统主要使用 ManagementObject、ManagementClass、ManagementObjectSearcher 类对象;ManagementObject 单个管理对象;ManagementClass 单个管理类;ManagementObjectSearcher 根据查询条件对 ManagementObject 和 ManagementClass 对象的集合进行检索;ManagementQuery 用作所有查询类的基础。因此应用程序可以使用 System. Management 中的这些类的派生类,查询所需要的信息。

3.2 软件结构和系统流程

软件中主要用到的实体类及其关系如图 3 所示。软件中主要的实体类包括:Service 类,对应计算机中服务的信息以及服务的相关操作;Process 类,对应计算机中的进程信息以及进程的启动和终止等操作;Disk 类,对应硬盘的使用信息。以及其他一些与计算机中的软硬件相对应的实体类。Computer 类与这些类之间是聚合关系,系统控制类通过对 Computer 类实例的相关的操作的调用来完成系统的各项功能。

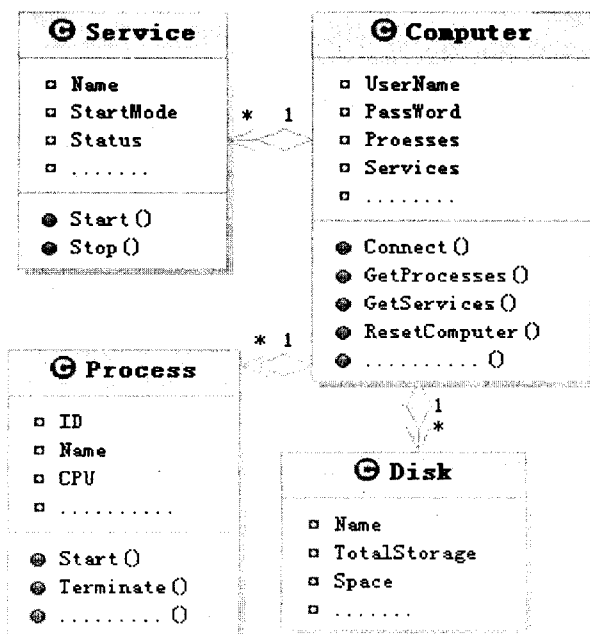


图 3 软件实体类关系图

软件启动以后首先从数据库获取各个无人值守计算机的相关信息,根据这些信息对每台无人值守计算机都实例化一次 Computer 类。对象创建完成后,为每个对象启动一个新的线程,在该线程中完成对无人值守计算机动态信息的获取以及信息分析,该线程的工作流程如图 4 所示。

3.3 关键技术及其实现

在 .NET 中,通过 WMI 获取远程无人值守计算机信息的过程非常类似,下面的示例代码为获取无人值守计算机进程信息的代码,获取其他信息可以仿照这段代码。

```
Public void getProcesses (string stringQuery)
{
    ConnectionOptions Conn = new ConnectionOptions();
    设置用户名、密码连接信息;
    Management.ManagementScope ms = new System. Management. ManagementScope("\\ \\\" + this. MachineAddress + \"\\ \\root \\ cimv2\", Conn);
    Management. ObjectQuery oq = new System. Management. ObjectQuery(stringQuery);
```

```

ManagementObjectSearcher query = new ManagementOb-
jectSearcher(ms, Conn);
ManagementObjectCollection queryCollection = query.Get();
计算每个 Process 的 CPU 占用率;
.....//将信息设置到 Process 对象
|

```

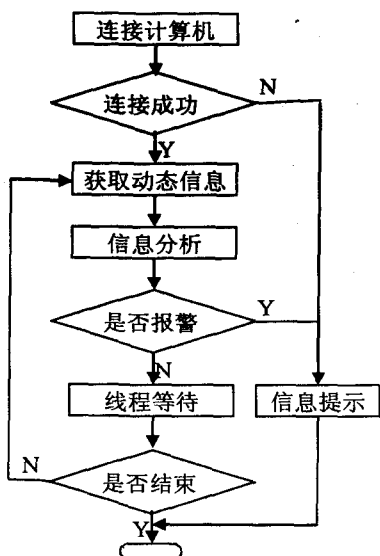


图 4 线程工程流程图

远程无人值守计算机中大部分进程信息可以比较容易地通过上述代码获得,但是每个进程的 CPU 占用率需要通过计算才能够得出,其主要的计算步骤是:

(1)通过计数器获取 idle 空闲进程 CPU 占用率 $r1$;获取各进程的 CPU 占用时间,求和,得各进程(除空闲进程 idle)CPU 时间和 $t1$;通过 $t1/(100-r1)$ 得到总 CPU 总时间 t ;主要实现代码如下:

```

private double TotalCpuTime()
{
    double d = 0; //非空闲 CPU 时间
    System.Diagnostics.PerformanceCounter PIdle = new PerformanceCounter("Process", "% Processor Time", "Idle", this.MachineAddress)
    double idlePercent = PIdle.NextValue(); //获取空闲进程 CPU 占用时间
    foreach (Process p in Processes)
    {
        if (新进程)
        {
            d += p.TotalProcessorTime.TotalMilliseconds;
        }
        else
        {
            d += p.TotalProcessorTime.TotalMilliseconds - p.told;
        }
    }
}

```

```

return d / (100 - middleCpuPercent); // 返回  $t1/(100-r1)$ 
}

```

(2)对各进程,获得该进程 CPU 累积占用时间 t_{new} ,计算: $(t_{new} - t_{old})/t$,即得该进程的 CPU 占用率,其中 t_{old} 是程序中记录的该进程上一次 CPU 累积占用时间。

```

Private void GetCpuPercent ()
{
    this.mCurrentTotalCpuTime = this.CalCurrentTotalCpuTime
(); //获取 CPU 总时间 t
    for (int i=0; i < Processes.Length; i++)
    {
        if (Processes[i].Id != 0) //非空闲进程 idle
        {
            long tnew = (long) Processes[i].TotalProcessorTime.TotalMilliseconds;
            double d = (tnew - Processes[i].told) * 100 / this.mCurrentTotalCpuTime;
            Processes[i].told = tnew;
            Processes[i].CpuPercent = d; //该进程的 CPU 占用率
        }
    }
}

```

4 结束语

通过 WMI 技术的应用,文中对无人值守监控系统中的无人值守计算机构建了一个方便有效的监控系统,来实现对其软硬件信息和运行信息的监视以及远程控制。微软在 .NET 平台中对 WMI 进行支持,使得软件的开发过程更加快捷。而且采用 WMI 技术,无需在被监测计算机上安装客户端,系统配置灵活方便。因此,采用 WMI 技术的无人值守计算机监控系统相对于传统的 C/S 模式的监控系统具有明显的优势。

参考文献:

- [1] 郑泽民,张艳霞,唐哲红.视频监控系统在无人值守行业应用分析[J].电信科学,2008(10):63-66.
- [2] 鄢家志,洪志全,刘秀芬.基于 OVPM 的主机性能远程监控的实现[J].计算机技术与发展,2007,17(5):28-30.
- [3] Microsoft Corporation. Windows Management Instrumentation SDK Documentation[M]. [s.l.]:[s.n.],2005.
- [4] Matthew L, Ashleym. Windows Management Instrumentation (WMI)[M]. Indianapolis: Indianapolis New Riders Publishing, 2001.
- [5] Thompson J P. Web - Based Enterprise Management Architecture[J]. IEEE Communication Magazine, 1998, 36: 80-86.

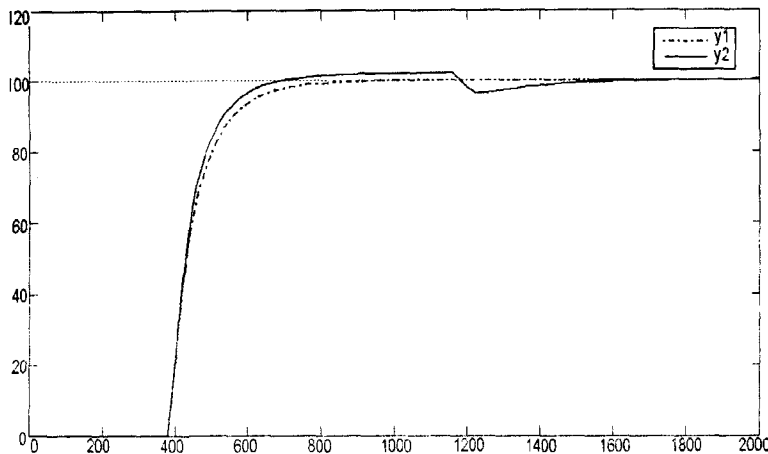


图 3 网络控制系统阶跃响应曲线
(y_1 : 仿真案例 I; y_2 : 仿真案例 II)

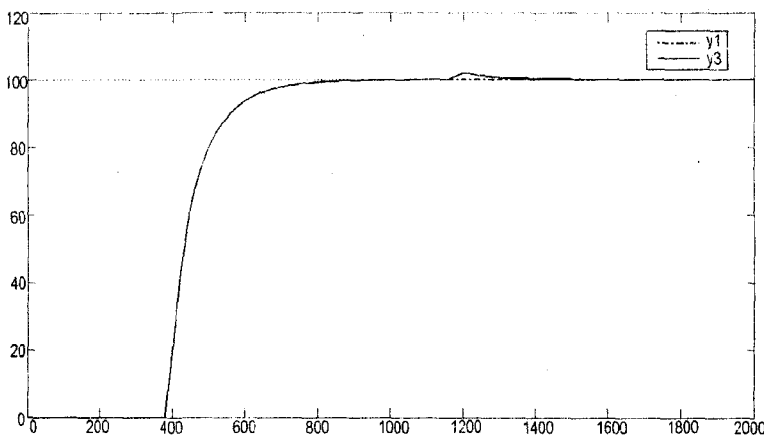


图 4 网络控制系统阶跃响应曲线
(y_1 : 仿真案例 I; y_3 : 仿真案例 III)

4 结束语

从网络控制系统先进控制器设计角度出发,探究通过改进网络控制器的设计方法,提高网络控制性能的方法。通过对传统 Smith 预估补偿方法进行改进,由此解决由于前向、后向网络延时和软件延时引入的新问题。然后通过相同网络条件下进行仿真实验,验证了改进的 PID-Smith 预估器能够更好地改善网络控制系统的控制性能,提高整个网络控制系统的稳

健性。

参考文献:

- [1] Yang T C. Networked control system: a brief survey[J]. IEE Proceedings - Control Theory and Applications, 2006, 153(4): 403 - 412.
- [2] Yang T C, Yu H, Fei M R, et al. Networked control systems: a historical review and current research topics[J]. Journal of Institute of Measurement and Control, 2005, 38(1): 12 - 16.
- [3] Yu H, Harding C, Hasan M S. Overview of networks and control[C]//Presented at the International Conference on Instrumentation, Control and Information Technology. Japan: [s. n.], 2005: 205 - 210.
- [4] 朱张青, 周 川, 胡维礼. 短时延网络控制系统的鲁棒 H_2/H_∞ 状态观测器设计[J]. 控制与决策, 2005, 20(3): 280 - 284.
- [5] 郭晓军, 袁 阳. 不确定时延网络控制系统的建模与稳定性研究[J]. 湖南工业大学学报, 2008, 22(5): 65 - 69.
- [6] Walsh G, Ye H, Bushnell L. Stability analysis of networked control systems[J]. IEEE Transactions Control Systems Technology, 2002(10): 438 - 446.
- [7] 张庆灵, 邱占芝. 网络控制系统[M]. 北京: 科学出版社, 2007.
- [8] 郑小宁. 基于因特网的远程控制技术[D]. 西安: 西北工业大学, 2002.
- [9] 程 彬, 李立伟, 何文雪. NET 环境下工控组件的设计与制作[J]. 计算机技术与发展, 2009, 19(11): 15 - 19.
- [10] 张光新. 啤酒生产过程的全自动化控制及其应用研究[D]. 杭州: 浙江大学, 2002.
- [11] Tipsuwan Y, Chow M Y. Fuzzy logic microcontroller implementation for DC motor speed control[C]//The 25th Annual Conf of the IEEE Industrial Electronics Society. San Jose, CA, USA: [s. n.], 1999: 1271 - 1276.
- [12] 刘金琨. 先进 PID 控制及其 Matlab 仿真[M]. 北京: 电子工业出版社, 2003.

(上接第 235 页)

- [6] 张施展, 高景昌. 基于 WMI 技术的计算机自动化管理[J]. 吉林大学学报(信息科学版), 2006(4): 451 - 454.
- [7] 戴大蒙, 李虎雄, 陈赛婷. 综合性能监控管理模型的设计与实现[J]. 计算机应用, 2007(1): 255 - 256.
- [8] 姜 边. 一种基于 .NET 元数据的对象持久化框架[J]. 计算机应用, 2005(4): 833 - 836.
- [9] 宋 昕, 盛 晨, 王新华. 基于 WMI 的计算机管理技术的研究与实现[J]. 浙江科技学院学报, 2007(1): 23 - 26.
- [10] 唐 忠, 何慧敏, 苏 飞. 基于 WMI 技术服务器网管软件设计和实现[J]. 桂林电子科技大学学报, 2008(6): 462 - 465.
- [11] 姜劲松, 吴礼发, 张 萍. 基于 WMI 的系统管理的设计与实现[J]. 计算机应用, 2004(3): 16 - 17.
- [12] 孙 涛, 姜良华, 席加铭. 基于 WMI 的公共机房学生操作行为限制系统[J]. 哈尔滨师范大学自然科学学报, 2009(3): 57 - 60.