AN EXTENDABLE WEB-BASED SYSTEM OF MANAGING DISTRIBUTED SERVERS USING IPMI AND WMI TECHNIQUES

Ming-Hsien Yeh¹, Yuan-Cheng Lai¹ and Jian-Wei Lin²

¹ National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C
Email: D9109106@mail.ntust.edu.tw; laiyc@cs.ntust.edu.tw
² Ching Yun University, Taoyuan, Taiwan, R.O.C
Email: jwlin@cyu.edu.tw

ABSTRACT

The rapid development of many Internet services and applications has resulted in increased demand for servers. Accordingly, a helpful tool to manage those servers is essential for educational institutions or enterprises. In this paper, a Web-based Server Management (WSM) system is proposed to ease managing distributed servers scattered in different geographic areas. The WSM system is characterized by the three features: (1) monitoring the server’s hardware health via Intelligent Platform Management Interface (IPMI) commands; (2) managing the server’s system resources via Windows Management Instrumentation (WMI) commands; (3) providing the functionalities of server groupings and task scheduling. Finally, the system performance and the investigation of WSM questionnaires have been conducted for verifying its effectiveness and validation. The software architecture used to develop the WSM is devised in a systematic and extendable way, which will also be elaborated herein.

KEYWORDS

Web-based application; distributed server management; IPMI; WMI.

1. INTRODUCTION

The popularity of the Internet and the wide diffusion of Web services and applications, have resulted in increasingly diverse, distributed, and complicated server environments. Server administrators not only must provide stable services to users, but also simultaneously face the much tougher challenge of managing the various server environments. Therefore, an effective management tool is required to lessen the complexity of managing such diverse server environments and the administrator’s load. By unifying the diverse and scattered servers within educational institutions and enterprises, reduction of costs, improvement of service and lowering crash risks, among other valuable goals can be accomplished. In our conception, an ideal server management system should realize the following main characteristics: (1) Remote Management: The ability to manage servers at any time and from any location. (2) Task Scheduling: A task can be executed automatically at a pre-designated time. (3) Group Management: The ability to divide servers into different groups to facilitate management. (4) Out-of-band Management: The ability to access the current status of servers, even when they are

© 2012 Pakistan Journal of Statistics 551
powered off. (5) *in-band Management*: The ability to obtain system information while servers are working.

Management methods such as SNMP (Simple Network Management Protocol), IPMI, and WMI have been commonly used to manage servers. SNMP is an application layer protocol for allowing network administrators to manage network performance and find network problem according to RFC 1157 (1990). IPMI is a specification for providing system management capability in hardware (IPMI home), and it includes a rich set of predefined commands and interfaces organized by types of operations, such as reading temperature, voltage, fan speed, and chassis intrusion. WMI is a component of the Microsoft Operating System (OS) for allowing users to virtually monitor each piece of a system either locally or remotely (WMI Homepage). In this paper, a total solution for server management is proposed. Using Web, IPMI, WMI, and database based techniques; a novel server management system is developed, called a Web-based Server Management (WSM) system. By browsing, administrators are able to manage a distributed server. IPMI is used to monitor and control servers regardless whether the server is power on or power off. WMI provides a common way to manage servers using Windows OS when they are power on. Lastly, by combining database with AT techniques (a Windows schedule service), group management and task scheduling are achieved.

The organization of this paper is as follows. Backgrounds concerning IPMI, WMI, and related works are presented in Section 2. Section 3 elaborates the design architecture and functionalities of our WSM system. Section 4 outlines the implementation issues of our WSM system. Section 5 presents the evaluation results while conclusions are given in final section.

2. BACKGROUND

The WSM system fully takes advantage of IPMI and WMI techniques for out-of-band and in-band management to integrate all distributed servers.

2.1 IPMI

Intelligent Platform Management (IPM) means that autonomous monitoring, controlling, and recovery features are implemented directly in platform management hardware and firmware. The key characteristic of IPM is that platform management functions, such as monitoring, logging, and recovery control are independent of the main processor, operating system, and BIOS. Thus, even when the computer is powered off, these functions can be accessed. IPMI is a standard which defines a common hardware interface to effectively support monitoring the system hardware, such as system temperatures, voltages, fans, power supplies, etc. In addition to hardware monitoring, IPMI also includes local or remote system resets and power on/off capabilities. IPMI is a type of hardware interface specification that is independent of management software. Its monitor and control functions can be utilized through standard management interfaces, such as SNMP, WMI, CIM (Common Information Model), DMI (Desktop Management Interface). Fig. 1 depicts IPMI and the software architecture stack.
The heart of IPMI architecture is a small and separate processor, namely the Baseboard Management Controller (BMC), on the motherboard. The BMC connects to the main processor, on-board sensors, and other hardware elements. The aims of BMC are to provide monitoring and to control the health of the platform’s physical hardware, and to further offer the interfaces for upper management application access.

2.2 WMI

WMI is a core management capability of Window OS, e.g., Windows XP and Windows 2003. The purpose of WMI is to provide a common way to manage nearly all Windows resources, such as BIOS information, drive partition information, OS information, and environment variables.

Most administrators use graphical administrative tools to manage Windows resources, such as disks, folders, file systems, printers, processes, registry settings, security, services, users, and groups. Prior to WMI, these Windows graphical administrative tools relied on Win32 application programming interfaces (APIs) to access Windows resources. This is also the only way that users could access Windows resources before the advent of WMI. At that time, there was no easy way to manage Windows resources using popular script languages since most script languages lack the ability to call Win32 APIs directly. WMI, which adopts the Component Object Model (COM) interface, makes system management easier and more secure. In addition, WMI offers a variety of programming interfaces, such as Visual Basic, C++, open database connectivity (ODBC), Perl, Active Server Pages (ASP), ASP.NET, and scripts within standard HTML pages.
Fig. 2: The Architecture of WMI

Fig. 2 illustrates WMI architecture. The WMI Consumer can serve as a script application, management application, Web-based application, or administrative tool; accessing and controlling management information through the WMI infrastructure. The system service, WinMgmt.exe, is the core of WMI and is known as the Computer Information Model Object Manager (CIMOM). It handles the interaction between WMI consumers and WMI providers. When the WMI consumer sends a request to obtain management information, CIMOM receives this request, identifies which provider has the information, obtains the data through this identified provider, and finally returns it to the WMI consumer. The WMI provider acts as an intermediary between the managed resource and CIMOM. The repository maintained by CIMOM contains object, class, and instance definitions that can be used to access and maintain system configuration information. The Managed Resources, such as the computer system, disks, files, folders, and applications can be managed using WMI.

2.3 Related Works

The Acer Server Manager v5.3 is one branch of Acer Technologies’ Server Management Solution (Acer Server Manager Homepage), which supports IPMI and WMI and group management. However, three additional components must be installed: “Agent” on each managed system, “Server” on the managing site, and “Console” on the client. The users use Console to connect with the managing site, and communicate with Agent through Console. Then the managing site retrieves the information and sends it to users. The RioView v1.1 is Windows-based server management software that was developed by Arima Corporation (Arima Corporation Homepage). This tool is used to manage Arima’s Scorpio server management card, which contains a BMC chip supporting IPMI v1.5. Thus, this tool provides out-of-band management. Though it can monitor hardware health, it cannot monitor the status of system resources. This tool must be installed in the client to manage servers. Kim showed the design and implementation
of a load cluster management system (LCMS) based on SNMP and Web technologies according to Kim (2002). The requirements of LCMS are examined to provide efficient and stable management operations and high availability. The LCMS follows the client-server management paradigm of SNMP, and consists of three managers having different roles, which distribute management functionality to all hosts in a cluster group. By using SNMP, this system can reduce the network bandwidth required in management operations. This system also provides automatic cluster configuration and current status monitoring of each host in a cluster group through Java and Web technologies.

3. SYSTEM DESIGN

In this section, we will expound on the system architecture, the management flow, the design issues, and the functionalities of the WSM system.

3.1 System Architecture

In IPMI, BMC has different communication interfaces, such as IPMB (Intelligent Platform Management Bus), serial/modem, LAN (Local Area Network), ICMB (Intelligent Chassis Management Bus). In order to integrate scattered servers, a LAN interface is chosen as the best choice. Our WSM system adopts a 3-tier structure, shown in Fig. 3, composed of a managing site, the client cluster, and the server cluster. The main reason for adopting a 3-tier structure, instead of a 2-tier structure, is that the managing site will maintain each server’s IP, account, and password. A 2-tier structure is an inconvenient way to manage the distributed servers, especially when the number of servers is huge, because the users must remember each server’s IP, account, and password individually. 3-tier architecture has adopted in Chirico (2005), Lin (2008), and Yan (2006) where a managing site centrally maintains all distributed server information.

In practice, the managing site installs not only Windows OS, IIS, SQL server 2000, and .NET Framework but also the WSM application, which is programmed in HTML, JavaScript, and ASP.NET. Each managed server must be equipped with IPMI and Windows OS. This also reveals that the proposed architecture is extendable since a new server equipped with these features can be integrated immediately into the system via LAN without modifying any program code. A client side can easily manage all the distributed servers via browsing by connecting to the managing site.
3.2 Management Flow

Fig. 4 exhibits the management flow, the steps of which are summarized as follows:

- **Step 1:** After logging onto the managing site successfully by inputting user name and password, authorized users can further manage all distributed servers.
- **Step 2:** Users select a managed server or group. Then they can carry out any provided functions, such as Group Management (go to step 3), Task Scheduling (go to step 4), WMI (go to step 5), or IPMI (go to step 6).
- **Step 3:** The managing site executes an “add node/group” command or a “delete node/group” command, and also updates database information.
- **Step 4:** The managing site adds a task to “Scheduled Tasks” or deletes a task from it, and also updates database information.
- **Step 5:** The managing site connects to the managed server by using WMI messages. After successfully connecting to the managed server, WMI commands and response messages will be transferred back and forth among servers for in-band management.
- **Step 6:** This step is similar to step 5, except using IPMI message for out-of-band management.
### 3.3 Design Issues

This subsection elucidates several aspects of our design.

- **Locating IPMI-enabled servers**: The managing site uses the IPMI “ping” message to discover the IPMI-enabled servers. This message resembles the “ping” network command which is used to test if a host exists on the Internet. If a requested server supports IPMI, it returns IPMI’s “pong” message to the managing site.

- **Group Management**: The WSM system uses a tree structure to implement the functionality of Group Management. The first layer specifies group names while the second contains the list of the servers. Such a structure facilitates maintaining the configurations of each server and group.

- **Security**: The WSM system uses Secure Sockets Layer (SSL) to encrypt the transmitted data on the Internet.

- **Task Scheduling**: Normally, AT (AT Command Homepage) and Schtasks (Schtasks Command Homepage), can be chosen to schedule a task in Windows OS. In fact, Schtasks contains more options and more completeness than AT, unfortunately it does not support Windows 2000 OS. Consequently, only AT can be chosen, despite having fewer schedule options.

- **Database**: The WSM database consists of seven tables: Account, Config, Found, Root, Node, Schedule, and Schedule Log. The Account table, which stores user id and password, is used to check the login. The Config table stores Retry and Timeout values. When performing “Search BMC Server” by specifying the IP range, the system stores these located servers’ IPs into the Found table. The Root table stores each group name. The Node table stores each server’s information, such as Server IP, WMI Account, and Password; as well as Group ID. The Schedule table stores each scheduled task’s info, including Task ID, Task, and Start Time. Once a task is activated by the schedule service at the specified time,
the Schedule Log table saves the execution results, e.g., task name, execution time, target server, and the status of this task, i.e., success or failure.

- Authentication: After logging onto the system successfully, the authorized user’s information will be stored in his session and can further manipulate the desired functions until his session expires.

3.4 Functionalities
The WSM system has four major functionalities, including Configuration, Command, Monitoring, and System Information, which are plotted in Fig. 5 in detail.

![Function Lists Diagram](image_url)

**Fig. 5: The Function Lists**

### 4. IMPLEMENTATION
In this section, we will elaborate how to implement the WSM system. The software architecture, shown in Fig. 6, is referred to Antonelli (2006) and Lai (2009), where a COM component is responsible for one specific task. There are four modules in the managing site, including IPMI Module, WMI Module, Task Scheduling Module, and Group Management Module.
4.1 Development Tools

Our development tools are Visual C++ 6.0 and Dreamweaver 2004 MX additional. The former is used to develop DLL and COM objects, while the latter is a webpage editor used to develop ASP.NET applications.

4.2 IPMI Module

We use the DLL (Dynamic Link Library) and COM (Component Object Model) objects to implement this module. IPMI DLL implements low level actions, namely IPMI packet transmission. It receives parameters from the IPMI COM object, fills these parameters into the proper packet fields, and then sends the packet on a socket. The IPMI COM object loads this IPMI DLL and implements various IPMI functions. ASP.NET creates an IPMI COM object and uses it to invoke all functions related to IPMI functions. We now demonstrate several IPMI commands. The IPMI command uses three methods in the IPMI COM object. The first method, `Connect()` method, is used to connect with a managed server. This method utilizes five arguments - ipArray, username, password, retry, and timeout. The ipArray is an array that stores an IP address, and the username and password are BSTR types of the COM object’s string type. The retry and timeout are integer types. This method returns a Boolean value to determine whether this `Connect()` method succeeds or fails. The second method consists of the IPMI commands, such as `PowerOn()` and `PowerCycle()`. The last method is `Disconnect()`, which is used to disconnect from a connected server. Many more of the remaining IPMI commands are omitted here due to space limitations.

4.3 WMI Module

WMI is based on the concept of the WMI provider, namely the Win32 provider. Many WMI classes are available and organized into groups identified by namespaces, which can be used once users create a WMI object. Subsequently, we demonstrate the manner of using the WMI Win32_Service class to display the list of all services installed on the system. First, information such as the server’s name or IP address, login account,
An extendable web-based system of managing distributed servers...

and login password must be established. The login account must be in the administrator group of the managed server. Then an object SWbemLocator can be created using a VBScript CreateObject() function call. After creating the SWbemLocator object, we can employ the SWbemLocator::connectServer() method to obtain a SWbemServices object that represents a connection to a namespace on a local or remote host. The SWbemServices::InstancesOf() method creates an enumerator which returns the instances of a selected class. After obtaining the Win32_Service instances, the information of interest can be iteratively displayed through them.

4.4 Task Scheduling Module

The Win32_ScheduledJob WMI class is used to create a task. When a scheduled task is executed, a batch file is called to run a WSH (Windows Script Host) program (WSH Homepage), which queries the database to determine all server information related to this task. Then this program creates an IPMI COM object to execute the task. WSH allows users to write scripts to automate tasks on a Windows OS.

The Win32_ScheduledJob class represents a task created with the AT commands from Control Panel/Scheduled Tasks. We can enumerate or delete any existing tasks displayed in the Control Panel/Scheduled Tasks using the Win32_ScheduledJob instances. Each scheduled task is stored persistently, implying that the scheduled task will not disappear after computer reboot. Tasks are scheduled according to UTC (Universal Coordinated Time) with bias offset from GMT (Greenwich Mean Time). The Win32_ScheduledJob class returns the local time with UTC offset when enumerating an object and converts it to local time when creating a new task. When a scheduled task is activated, it executes a batch file with a task number. Then this batch program calls the WSH program to perform this task. The WSH program first searches this task’s information, including Task, Type, and ID from database. The Type value is used to distinguish that the target is either a single server or a group. If Type is “Server”, then WSH queries this server’s information from the database according to the ID value, and connects to this managed server. According to the specified Task content (Power On, Hardware Reset, etc.) for execution of this task, and then disconnect this connection from the managed server. Finally, it records the task’s results into the database. If Type is “Group”, then it searches all server information belonging to this group, and iteratively executes each server’s task.

4.5 Group Management Module

The relationships between groups and servers are expressed in a tree structure which can be translated into the tables presented in Fig. 7. The first table is a Root that records each group’s information, and the second table is a Node that records each server’s information.

For GUI, the ASP TreeView component (Obout Inc Homepage) is used to draw the hierarchy between the groups and servers. Initially a TreeView object is created, and then all group information is queried from the database. Afterward, the TreeView::Add() method is iteratively employed to add each group into the tree structure. Then, the TreeView::Add() method is iteratively invoked to assign each server to its corresponding group. Finally, we use the TreeView::HTML() method to draw the tree in a Web page.
Fig. 7: The Entity Relationship (ER) Diagram of the Group Management

Table 1: The Efficiency of each Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Efficiency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Settings</td>
<td>100</td>
</tr>
<tr>
<td>Sensor Settings</td>
<td>150</td>
</tr>
<tr>
<td>Privilege Settings</td>
<td>190</td>
</tr>
<tr>
<td>Set SEL Timestamp</td>
<td>70</td>
</tr>
<tr>
<td>Finding BMC Server</td>
<td></td>
</tr>
<tr>
<td>Manual Management</td>
<td>100</td>
</tr>
<tr>
<td>Search BMC Server</td>
<td>Depend on IP range</td>
</tr>
<tr>
<td>Found BMC Management</td>
<td>15</td>
</tr>
<tr>
<td>Get Status</td>
<td>50</td>
</tr>
<tr>
<td>Power Control</td>
<td>55</td>
</tr>
<tr>
<td>Schedule</td>
<td>140</td>
</tr>
<tr>
<td>Schedule Information</td>
<td>220</td>
</tr>
<tr>
<td>Schedule Event Log</td>
<td>15</td>
</tr>
<tr>
<td>Sensor Monitoring</td>
<td>330</td>
</tr>
<tr>
<td>Device Information</td>
<td>320</td>
</tr>
<tr>
<td>System Event Log</td>
<td>515</td>
</tr>
<tr>
<td>BIOS</td>
<td>40</td>
</tr>
<tr>
<td>Physical Drives</td>
<td>110</td>
</tr>
<tr>
<td>Logical Drives</td>
<td>45</td>
</tr>
<tr>
<td>Memory</td>
<td>70</td>
</tr>
<tr>
<td>Processor</td>
<td>2,000</td>
</tr>
<tr>
<td>Network</td>
<td>85</td>
</tr>
<tr>
<td>Operating System</td>
<td>125</td>
</tr>
<tr>
<td>System Drivers</td>
<td>450</td>
</tr>
<tr>
<td>Process</td>
<td>900</td>
</tr>
<tr>
<td>Services</td>
<td>300</td>
</tr>
</tbody>
</table>
5. SYSTEM EVALUATION

Table 1 displays the efficiency of each function in the WSM system, which reveals extreme tiny response time responding to these functions.

Table 2 outlines the comparison with other server management software on system functionality. Among them, one limitation of the WSM is unable to manage the servers with heterogeneous OS (e.g. UNIX).

<table>
<thead>
<tr>
<th>Functionality</th>
<th>WSM</th>
<th>Acer Server Manager</th>
<th>Arima RioView</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Management</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPMI support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WMI support</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Server groupings</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Task scheduling</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Requiring extra software installed in managed servers</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Able to manage the servers with heterogeneous OS (e.g. UNIX)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

After WSM had been established, 30 IT engineers were invited to test the system. Afterward, they were further asked to fill a questionnaire whose questions are outlined as follows:

1. Accessibility: whether the system provides an easy access way for managing servers (e.g. using any browsers to manage at any time and at any place?)
2. Friendly GUI: whether the provided GUIs are sufficient and handy?
3. Management efficiency improvement: whether the provided functions improve your management efficient and shorten your time spending on server management?
4. Usefulness: whether WSM offer useful way to simplify the management procedure?
5. Overall judgment.

One of the answers, Totally Approval, Approval, Neutral, Disapproval, and Totally Disapproval, had to make to each question. The evaluation results are illustrated in Fig. 8, showing that most of the evaluated aspects received positive responses. The majority approved that WSM is a practical tool and indeed promotes management efficiency. Meanwhile, they also recommended several concrete suggestions, including adding more detail online helps and animation narrations.
6. CONCLUSION

The maturation of network environments requires the appropriate tools to conveniently manage pervasive servers. Web-based Server Management (WSM) system is developed to easily manage many distributed servers. Two techniques, IPMI and WMI, are adopted for out-of-band management and in-band management, respectively. In addition, the system provides the functionalities of server groupings and task scheduling. The proposed system architecture is also systematic and extendable. Some evaluations have also conducted. We are now gathering more valuable comments so that we may further fine-tune this system.

REFERENCES


