Journal of ELECTRICAL ENGINEERING, VOL. 56, NO. 1-2, 2005, 53–56

REVIEWS - LETTERS - REPORTS

AN ARTIFICIAL INTELLIGENCE SYSTEM PROVIDING SUPPORT TO SYSTEM ADMINISTRATORS MANAGING IN A DISTRIBUTED, HETEROGENEOUS ENVIRONMENT

Izabella Lokshina — Richard Insinga *

The paper discusses an artificial intelligence system providing a solution to the complex problem of system administration. It operates in a distributed system management environment that monitors hosts across a network, detects complex problem conditions, and takes automated, unattended actions to resolve common problems. The system was specifically developed to support system administrators managing distributed multi-vendor heterogeneous networks in financial services in Russia. The system intelligence is based on client daemon processes on all hosts of the network that independently can gather system data into a local database and analyze the data using an expert system. The system can manage problem events and actions; it distinguishes itself by its scalability and ability to be easily customized by the primary users.

K e y w o r d s: artificial intelligence methods, networks, system administration, distributed heterogeneous environment, system intelligence, built-in rule-based expert system

1 INTRODUCTION

Distributed systems dominate the computing landscape these days. Distributing the processing, databases, and communications allows companies to move more quickly and to provide the rapid growth and continual changes in the e-economy. Although the concepts behind distributed systems were conceived more than 30 years ago, the use of distributed systems moved into a new era with the advent of the Internet. Previously, the bulk of distributed systems were in-house and linked systems within a company. That has changed as the Internet turned attention outward by providing the infrastructure for advanced and global distributed systems which have to support corporate strategies and corporate success [5] [6] [7]. On the other hand, the Internet challenges the architecture of distributed systems because these need to be open and more sophisticated. System administrators have much greater difficulties than ever when managing these networks. Studies indicate that an average of approximately 40time is spent on fault management or fixing problems, as opposed to configuration management and/or capacity planning. Traditionally, end-users call a help desk or a system administrator when they have a problem. Problems include file systems filling up, printer queues not working, electronic mail not reaching its destination, or a process getting out of control on the system. The system administrators try to diagnose the problem; they look at the system and check various system statistics; and they consider the time of day and other issues that are specific to the particular company or situation. Very different actions could be necessary if a key end-user has a problem while trying to make a critical deadline or if a night shift operator has a problem at 2:00 in the morning. The system administrators often have to perform their functions in inconvenient and critical situations [1] [3] [4]. The artificial intelligence system SYMON (stands for SYSTEM MONITOR) is a distributed system management environment that monitors hosts across the network, detects complex problem conditions, and takes automated unattended actions to resolve common problems. The system was specifically developed to support system administrators managing distributed multi-vendor heterogeneous networks in financial services in Russia. Using SYMON, system administrators can discover potential problems prior to their general occurrence. The system automatically takes actions to fix the problems without any human intervention. As a result, system administrators and end-users increase their productivity and the system down time is reduced [2] [3] [4].

2 AI TECHNOLOGIES APPLIED IN THE SYSTEM

The system provides proactive monitoring and automatic actions to fix problems in distributed systems through the use of the following AI technologies: distributed relational database management system, rule-based expert system (ES), alert handler, and action manager. These technologies give the system the ability to automate system management by replicating some of the tasks of system administrators. Therefore, these technologies (as well as computer applications) reside on every host in the system.

* Division of Economics and Business, SUNY Oneonta E-mail: lokshiiv@oneonta.edu, insingrc@oneonta.edu

ISSN 1335-3632

Automated problem resolution

Like a human system administrator, each application collects data about the resource it is monitoring. Unlike a human, the application collects the data in regular intervals. The expert system provides the ability to monitor complex practical conditions. For instance, the system can monitor a particular state that results in impaired service for the end-users. The data is collected into the relational database from many locations on the system. The application uses the expert system rules to test for various complex conditions. When a set of conditions requiring attention has been met, the application generates an alert that is managed by the alert handling engine. A copy of the alert must be sent to the system administrator's console, thereby advising the system administrator of the condition. The administrator then can handle the condition manually or can configure the system to automatically fix the problem.

Alert handler

A special ability of the system is the sophistication of the alert handler. The alert can be escalated in priority over time and different actions can be performed at each priority level. This corresponds to the different actions human administrators may perform when faced with a problem of changing significance. The application sends an alert to the system administrator's console when the problem is resolved. This way the system increases productivity of the system administrator and the end-users.

Scalable architecture

The system architecture is scalable, which means that it can be deployed across a large number of machines across multiple networks. The design considerations for scalability are divided into three main areas.

The first, there is no dependence on central servers or object brokers. The advantage is that the primary users can manage a large number of systems across multiple networks, even with wide-area connections, and can minimize the impact on the overall network performance. All processing must be performed at the client level in the system architecture. Contrary to the old-style console-polling model, clients are able to operate even if no console is alive or the network is down. Consoles are only used to display alerts generated by the clients. Old-style console polling is the standard polling of data agents from one or more monitoring consoles. The consoles periodically send data request messages to the agents and the agents gather the requested data and send it back in data messages across the network. The console then processes the data and initiates the simple action requests on the local agent systems (if that capability exists). The disadvantage of this model is that traffic is repeatedly going across the network and that network traffic grows linearly as workstations are added. If 15% of the network capacity is used to monitor 200 workstations, then at least 30% of the network will be used to monitor 400 workstations. Consoles can become bottlenecks themselves, limiting the total number of systems that can be managed from any one console. If no one console is available or alive, no monitoring occurs. Likewise, if the network is down, no monitoring occurs [1] [3]. Client-centric processing is a step forward in performance optimization and memory utilization of relational databases and expert systems. It is based on placing a fully capable relational database and a standard rule-based expert system onto each client system with minimal performance impact. All the data analysis tasks that used to require powerful server systems to manage the data and to perform problem analysis are located on each of the clients. Therefore no raw data has to go across the network to be processed. The only messages sent from clients to consoles are qualified alert notifications and responses to deliberate requests by a system administrator for data to be used for historical reporting and real-time network status displays.This architecture scales across large and wide-area networks, and alleviates the requirement for dedicated console server systems.

The second, the primary users can have different groups of system administrators monitor the same client hosts without causing undue network traffic or system load on each of the clients. This is important for companies that have corporate-wide operations groups or after-hour support centres in addition to the local day-time system administrators. The system can adjust to the organizational structure when more that one individual or group is responsible for managing a particular group of systems. There are two key aspects to this class of scalability: Peer Consoles and Super Consoles. Workstations need to be managed by multiple peer administrators who can work cooperatively. This is usually the case. When hundreds of workstations are managed by two or three system administrators, each of them may have a different focus or specialization, yet each system administrator can manage any system in that group. This capability is supported by Console application which directly manages client workstations. To assist help desk operations, the system supports the Problem Dispatch Queues and Individual Work Queues concepts, where alerts can come into a central dispatch desk to be assigned to individual specialists on the team. Often companies at the corporate level provide back-up support or after-hour support which is not local to the installation site and is connected to the local network by slower wide area connections. It would be cost prohibitive in network traffic if all individual client workstations were to report directly to consoles at a central support group. Also it would require too much administrative overhead to inform the central support of any additions and removal of client systems on the local networks. Therefore super consoles only talk directly to one or more standard consoles. They do not know about specific client hosts and never talk to clients directly. Super consoles consolidate all the alerts from the attached consoles, and system administrators at the super consoles can still take the same actions that someone at a regular console can.

The third, the primary users can create configuration templates for different machine classes, operating systems, hardware configurations, and end-user applications. Each individual client system determines which classes it belongs to and constructs its own database. The configuration templates can be arranged hierarchically to match a company's organizational structure and to provide corporate default and mandatory configurations. The system supports its ability to configure system clients from a central configuration database. Network-wide defaults apply to most configuration categories and can be overridden for individual hosts. This capability allows for configuration of a fairly homogenous group of clients on a centrally managed mid-sized network. On the other hand, large companies with complex organizational structures and several thousands of workstations often have multiple autonomous business units that manage their own networks and applications. Usually a central corporate group providing support to the business units sets certain company standards. Each of the business units may have subgroups responsible for certain areas. Large companies usually have a technical hierarchy for managing systems across their networks and frequently have systems from multiple vendors and are running different versions of the operating system. The construction of a configuration database for such a variety of system combinations in a corporate management hierarchy with a fairly independent organization structure should be done at the local level. The system allows the primary users to perform self-configuration or create configuration templates for each system type, operating system, off-the-shelf application, and in-house proprietary application. Each group can set default and mandatory configuration templates for its sub-groups, and each sub-group has the ability to override individual non-mandatory configuration parameters in each of the templates. When any of the templates are changed, each client host determines on its own what type of system it is, what application it has installed, and to which group it belongs, and then constructs its own configuration database from the templates. The system uses object-oriented technologies to allow distributed self-configuration. The system architecture consists of three layers of technology: core, open problem management design, and applications. The core layer contains the distributed database management system, the rule-based expert system, the communication capabilities, and C programming language environment (since the core layer is implemented in C). Above the core layer is the open problem management design, which uses the core layer to implement a set of services including configuration management, alert handling, action management, report generating, rule execution, and communication between client, consoles, and super console processes to pass data and execute remote procedures. Individual applications rely on services provided by open problem management design and add functionality specifically to applications such as defining databases, generating and analyzing data, generating alerts, and defining application specific actions. Everything else is handling by the open problem management design. The primary users can add their own specific site applications at run-time or compile them into an execution system using a rapid application development environment.

Expert system

The forward chaining rule-based expert system uses a particular algorithm to automatically prioritize its rules. Similar to an integrated database, the ES contains new and unique features that are useful in automating system administration tasks. Rules are grouped by functionality. System monitoring consists of three main tasks: data gathering, data analysis, and problem resolution. The integrated ES associates each rule with one of the tasks. This allows the expert system to gather all the data necessary to make decisions, analyze and process the data, and resolve the problems by posting alerts to the alert manager which causes actions to be performed automatically. Rules are automatically ordered by complexity. Proper resolution of a problem may be dependent on several external factors. For instance, if a system is running out of swap space, it is generally desirable to allocate additional swap space on the local disk. However, it may not be desirable if the local disk space is also a critical resource. In this case, additional swap space could be allocated from the remote file server. However, if network resources are limited as well, the optimal solution might be to shut down less critical applications. Because the expert system automatically uses the most specific rule for a given situation, there is no need to be concerned with complex rule sequencing when developing new applications. There is a need to write only one rule for each case. The expert system will choose the proper rule automatically. Each rule can operate on a separate schedule. Some system administration tasks are performed more frequently than others. Then the rules which automate them need to be executed more frequently, too. The expert system understands this and allows each rule to specify its own schedule and even dynamically change its schedule as needed. By limiting execution of rules in this manner, the system maximizes its efficiency and minimizes system resource usage.

Problem Management Concepts

The open problem management design provides sophisticated problem management capabilities. The key problem management concepts and design requirements are given below. Most of the engines are controlled directly from the configuration database, which means that the behaviour of alerts can easily be changed through simple configuration changes. The most important requirement is one alert for one problem. Many problem management systems create a multitude of alerts for different symptoms of the same problem. Open problem management design promises alert uniqueness and does not allow the same alert to be generated twice for the same underlying resource. Problem hierarchies are used to prevent a problem from triggering several overlapping alerts.

Problem hierarchies are specified in the system configuration database and are completely customizable by the primary users. Each alert within the system has an associated priority, which is used to determine the seriousness of the detected problem. Currently, alerts are classified into one of five priority levels. The system provides a unique engine for assigning the priority of an alert over time. This engine can either promote or demote the priority. Escalation is useful since it may be necessary for an initially insignificant problem to escalate if it is not resolved within a specified period of time. The action manager allows automatic actions to be taken at each new priority level. The system allows alerts to expire automatically after a specified period of time. This feature is useful because it allows informational messages to be posted only while such messages are relevant. By expiring alerts when they are no longer relevant, the system reduces the occurrence of information overload. The system allows system administrators to assign ownership of an alert to one or more people or groups. This feature is used to implement Problem Dispatch Queues and Individual Work Queues. When an alert has at least one owner, the alert can escalate or time out. The alert remains active within the alert handler, however, and will not be removed until it is cleared. Often alerts are generated for certain special conditions that are acceptable on certain hosts. For instance, it may be all right for the end-user on the specific host to be at 98%, since the file system space on that host never increases. Instead of having to create a special configuration entry for that host to prevent an alert for that file system to be reported from the host, it is possible to simply ignore the alert at the console display. The alert, however, is not suppressed forever at this point. Each time an alert is ignored, it is suppressed for an increasing period of time, until it becomes blocked forever. The primary user can define the intervals of the ignore engine that lets the system learn the alert preferences of system administrators.

Action Management Concepts

The open problem management design integrates the action manager, which allows system administrators to perform actions from the console. In addition, actions can be initiated automatically upon creation and escalation of alerts. The system provides numerous system management actions. Predefined actions cover file system management, process management, adding temporary swap space, controlling daemon processes, or managing the system itself. These actions are often driven from defaults specified in the system configuration database. System administrators can easily define actions and register them with the system. User-defined actions commonly call programs with arguments from the configuration database, the command line, or any alerts that the actions have been triggered against. Manual actions are usually performed on remote hosts. The system administrator has the ability to specify a single host, or a named list of hosts, on which the actions could be executed. Most built-in actions are platform independent and allow the system administrators to take multi-host actions on a heterogeneous group of systems. The system administrators are informed at all times; and action results and errors are automatically displayed as action alerts on the console displays.

3 FINDINGS AND CONCLUSIONS

In this paper, an artificial intelligence system that provides a unique solution to the complex problem of system administration is considered. Specifically this system automates system administration tasks in a distributed multi-vendor environment; identifies potential problems before they affect end-user productivity; presents problem information to system administrators in a useful manner through the open problem management design; automatically acts to resolve problems while keeping system administrators informed of its progress. This system is scalable; and it uses minimal system and network resources to avoid affecting end-user performance.

References

[1] GELMAN, A. D.—HALFIN, S.: Analysis of Resource Sharing in Information Providing Services, IEEE GLOBECOM'90 (1990), 312-316.

[2] Introduction to SYMON Concepts, Hotline-Telecom, Moscow, 1998.

[3] DEEV, V. V.—LOKSHINA, I. V.: AI - Monitoring Systems, PROGRESS, Moscow, 1998.

[4] RADEV, D.—RADEVA, S.: Intelligent methods for traffic prognoses, Proceedings of the 12th conf. Transport'02 (2002), Sofia, 29-34.

[5] STEFFERUD, E.—FARBER, D.—DEMENT, R.: SUMURU - Network Configuration for the Future, Mini-Micro Systems (1982), 311-312.

[6] WEILL, P.—BROADBENT, M.: Leveraging the new Infrastructure - How Market Leaders Capitalize on Information Technology, Harvard Business School Press, Boston, MA, 1998.

[7] WETHERBE, J. C.: IS - To Centralize or to Decentralize, SIM Network, Society for Information Management, Chicago, IL, (1987).

Received 29 October 2004