A Monitoring System Based on Nagios for Data Grid Environments*

Hsiu-Lien Yeh[1], Yan-Fu Chen[1], Tsung-Tai Yeh[2], Pei-Chi Huang[2], Shin-Hao Liu[2], Hsin-Wen Wei[2a] and Tsan-sheng Hsu[2]

[1] Institute of Information System and Applications, National Tsing Hua University, Taiwan
[2] Institute of Information Science, Academia Sinica Taipei, Taiwan

Abstract - *The amount of digital data in today's society is already enormous and it will continue to grow exponentially. Therefore, it is necessary to devise new ways to preserve and manage the data effectively and efficiently. SRB (Storage Resource Broker), and its extension iRODS (the Integrated Rule-Oriented Data System), are data grid technologies for managing colossal amounts of data. In a distributed environment, monitoring systems oversee the operation of computing systems. The monitoring service is crucial because it must ensure a high-quality computing environment and provide reliable services. In this paper, we introduce a monitoring system called SIAM, which is based on Nagios. SIAM supports full monitoring services for SRB/iRODS-based systems, including fault-tolerance and notification functions. This study focuses on extending existing components and notification functions to satisfy clients' needs and improve our system's failover scheme. The results of experiments show that the proposed system is feasible for cloud storage services, and it is adaptable robust, and responsive in the face of system failures. Overall, SIAM enhances the reliability of SRB/iRODS based systems significantly.*

Keywords: Monitoring; Distributed system; Fault- tolerance

1 Introduction

As the digital era continues to evolve, the volume of digital information will grow exponentially. According to the International Data Corporation (IDC), on a global level, the amount of digital information doubles each year [1]. As a result, a great deal of research effort has focused on data preservation and management issues. How to manage, store, maintain and understand the collected data are critical issues. In recent years, data grid systems have been used extensively to handle huge amounts of data, and the grid concept may provide solutions to the above-mentioned issues. According to a number of studies [2][3][4], SRB and iRODS are two of the most widely used data grid systems. The architecture of iRODS is modeled on that of SRB, and combined SRB/iRODS systems are used in many academic projects, as well as in national and international research institutions. Such systems facilitate improved information collaboration and management of mass data storage [5]. Additionally, a monitoring system is usually designed to address certain stability and availability issues in a data system; hence, many research institutes and companies devote considerable resources to designing highly efficient monitoring systems [6][7][8]. The purpose of a monitoring system is to detect system faults immediately. In cases where the system has failed or is about to fail due to excess information or the computational load, e.g., disk corruption, overloading, or communication hot spots, a monitoring system sends out a warning signal, and thereby prevents a system failure or at least reduces the system's downtime. The monitoring system also ensures that grid computing services are not interrupted; therefore, the risk of losing data after a breakdown is minimized, service quality is improved, and clients are satisfied. Consequently, such systems play a major role in helping service providers to ensure the reliability and usability of their services. In addition, a failover handler is an essential mechanism that ensures the operation of monitoring systems is not interrupted.

Several monitoring systems have been designed for various computing environments. Generally, the systems can be classified as built-in systems or distributed systems. The first type is an application that runs in an operating system, such as a Unix-like system [9], which normally contains a monitoring subsystem that analyses a computer's current operational status. The second type is a monitoring host that is capable of observing various machines, such as Nagios, Ganglia, Cacti, and Hawkeye [10][11][12][13], simultaneously. In both cases, a monitoring system requires an active response mechanism for different devices, so that it can quickly identify potential failures and transmit warning messages. Monitoring systems must deal with a variety of dynamic resources available in a distributed environment; thus, special mechanisms must be designed and incorporated

*This research was supported in part by the National Science Council,Taiwan, R.O.C., under Grant NSC 95-3114-P-001-007-MY3 and NSC99-2631-H-001-024.

[a] Corresponding author: hwwei@iis.sinica.edu.tw;
Institute of Information Science, Academia Sinica, No 128, Section 2, Academia Road, Nankang, Taipei, Taiwan, R.O.C;
Phone: 886-2-2788-3799 ext.2471; Fax: 886-2-2782-4814

into the system to satisfy the specific requirements of the environment.

Some researchers, such as Zanikolas [6] and Massie [14], have observed that designers typically need to consider several features before implementing a monitoring system. According to our survey, the key design features of monitoring systems include scalability, extensibility, portability, robustness, manageability, reasonable overhead, and security. After conducting a thorough survey, we chose the Nagios open source program for our implementation. Nagios is utilized by several monitoring systems, such as GridICE and EGEE [15][16][17], and is supported by a strong open source community, e.g. NRPE, NDOUtils, and PNP [17][18][19]. However, Nagios does not support the SRB/iRODS system or the graphical visualization of the Nagios front-end website; instead it just provides regular notifications via E-mail. Besides, the Nagios monitoring system causes malfunctions and stops entirely if an error occurs in one of the monitoring hosts. This problem could be avoided if the system contained an effective backup mechanism, because a backup monitoring host would take over immediately and the service would not be interrupted.

Nagios is integrated with the monitoring architecture, which oversees the SRB/iRODS system and servers in the distributed environment. The SIAM system, which works independently of the computing system, tracks the activities of servers and the computing system, and uses several real-time notification services to inform system administrators when faults occur. Since SIAM utilizes Nagios open source software, it can be extended and maintained based on the requirements of the system components and services. As well as being extendable, it is readily available, efficient, and easy to integrate. SIAM provides an appropriate infrastructure for monitoring a data system environment; hence, it can easily detect the real-time status of the servers and systems. It also contains a fully tested fault-tolerance mechanism and only incurs a small additional overhead. When a system malfunctions suddenly or shuts down, SIAM prevents a total failure by enabling the system to reboot and continue operations immediately. In this paper, we implement a monitoring system based on Nagios for a grid environment, and test its ability to extend the monitoring function.

The remainder of this paper is organized as follows. In Section 2, we review existing monitoring system architectures. In Section 3, we describe the proposed extension of the Sinica SRB/iRODS Monitoring System; and in Section 4, we evaluate the system's performance. Section 5 contains some concluding remarks.

2 Survey of related software

In this section, we compare four widely used monitoring systems, and discuss their limitations. We discuss Nagios in more detail because it provides the basis for our system. Three known extensions of Nagios are also considered. Monitoring systems like Ganglia, Cacti, Hawkeye, and Nagios provide basic functionality for monitoring hosts, services, and resources. We describe those systems below.

Table 1. The comparison of different monitoring system.

Monitoring System	**Advantages**
Ganglia	Scalable architecture (clusters in particular) Graphic support Basic historical data analysis
Cacti	Excellent graphic displays Web management interface
Hawkeye	Notification mechanism Multiplatform Possible custom-made sensors
Nagios	Excellent extensibility Notification mechanism Low overload
Monitoring System	**Disadvantages**
Ganglia	No web management interface Complicated system settings
Cacti	Poor extensibility No notification mechanism
Hawkeye	Poor front-end The system is under-developed
Nagios	No graphic display support No web management interface
Monitoring System	**Key Design Features**
Ganglia	Scalability Robustness Reasonable Overhead Portability
Cacti	Manageability Robustness Overhead
Hawkeye	Scalability Extensibility Overhead
Nagios	Scalability Extensibility Robustness Security Overhead

Ganglia, an open source distributed monitoring system developed by the UC Berkeley Millennium Project [11][14], has a hierarchical architecture and relies on a multicast-based announce protocol to monitor the states of systems. It also uses technologies like XML for data representation, PHP for web development, and RRDTool (Round Robin Database) for data visualization. RRDTool is a popular application for storing time series data in graphic form [9][20].

Cacti is a network monitoring system that presents the system performance in graphic form [12]. It utilizes the SNMP protocol to collect information from various monitoring machines, RRDTool for the graphical presentation of monitoring information on web homepages, and a MySQL database for data storage.

Hawkeye, developed by the Condor group, is designed to monitor distributed systems [8][13]. It is implemented in two stages. First, using the Condor ClassAd Language, Hawkeye identifies problems based on the attribute values of the resources, namely, a ClassAd. Second, the manager can collect user information and deal with user inquiries when a problem occurs. Because Hawkeye uses Condor ClassAd

Language and ClassAd as building blocks, managers find the system easy to operate.

Nagios is a widely used and scalable monitoring tool developed by Ethan Galstad in 1999 [10]. It is an open source software framework under General Public License (GPL), and it provides many standard and specialized plugins for grid systems.

Table 1 compares the advantages and disadvantages of the four systems described above. Ganglia's installation procedure is quite complicated for users; Cacti has less configuration extensibility; and Hawkeye is limited because it is under-developed. In contrast, Nagios is easier implement, more extensible and easier to maintain. Thus, we selected Nagios as our model framework. For a more extensive introduction to these systems, readers may refer to the following works [8] [9] [10] [11] [12] [13] [14].

2.1 Nagios

The Nagios architecture involves a client-server layout, so it is easy to incorporate customized plugins, and system developers can try to satisfy diverse requirements by modifying or improving the system's functionality. The main functions of the Nagios system are: monitoring the utilization of several grid middleware services and infrastructures, helping web interface-enabled users to view generated reports and monitor tuning, supporting widely used notification mechanisms (E-mail), and supporting monitoring with automatic event handlers. Nagios Process Check Logic and Nagios Remote Plugin serve as add-ons. Plugins are written in C and consist of execution scripts. Based on the hardware and software characteristics, system administrators can allow other applications as add-ons to the basic plugins. Nagios provides system administrators with five mechanisms for communicating with different types of monitoring apparatus.

2.2 Extension of Nagios

NRPE (Nagios Remote Plugin Executor) is a basic secure tool set that enables monitoring of remote hosts [17], and is used to drive Nagios Plugin applications on remote hosts.

NDOUtils is an add-on application for the Nagios system. It stores profile parameters and the monitoring records of the system's status in a database [18].

The Nagios add-on PNP application "PNP-is-not-PerfParse" provides graphical functions to display monitored information [19]. PNP analyzes Nagios' monitoring services and generates performance data. It also stores the monitoring information automatically by using RRDtool, which helps in collecting data and displaying various graphics. Graphical tools can also be configured by PNP software.

2.3 Limitations of software based on Nagios

Software based on Nagios has the following limitations:

- It is not specifically designed for SRB/iRODS systems.
- It only provides a web interface to view monitored information. It is difficult for system administrators to use because the interface lacks graphical support and it cannot be modified.
- It sends a message if an error occurs, but it is difficult for administrators to distinguish between high priority error messages that need immediate attention and less urgent warnings. That is, users need to be able to customize the levels of error messages according their needs.
- It only offers one way to receive notifications, i.e., through e-mail.
- It lacks a built in mechanism for fault tolerance.

We integrate a monitoring system to overcome these limitations.

3 Sinica SRB/iRODS monitoring system

SIAM was built for the data preservation systems developed for the Digital Archives Remote-Backup (DARB) project [21][22] as part of TELDAP (Taiwan e-Learning and Digital Archives Program) [23]. The design of the data preservation system in DARB is based on SRB/iRODS middleware. To provide monitoring services for the data preservation system, we proposed using SIAM, and based the design of our system on the features of SRB/iRODS. We describe the features and components of the SIAM system in the following subsections.

3.1 Basic features

We use Nagios as the core system in the design of SIAM and, based on the features of SRB/iRODS, we offer various components as extensions of the monitoring system. Figure 1 illustrates the utilization of SIAM in a data grid environment. We extend the Nagios monitoring service to monitor the overall data preservation system, and improve the main features of the monitoring system to provide the following functions: service availability checks, system error detections, and resource management.

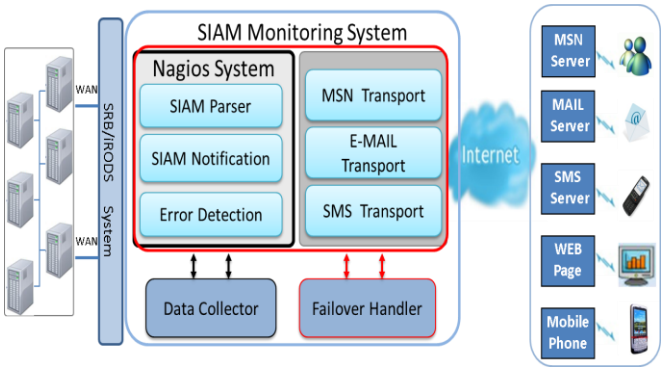

Figure 1. The configuration of SIAM in a data grid environment.

3.1.1 Service availability checks

The monitoring system periodically tests the accessibility of the data preservation server. If a login operation fails, the

monitoring application will wait for a period of time and then try to perform the login operation again. The process is repeated up to three times, after which the monitoring system immediately notifies the system administrators of the failed login attempt.

3.1.2 System error detections

The data preservation system keeps track of error messages and system processes and stores the information in log files. SIAM parses the files so that the monitoring system can identify important messages about the computing system and related components in the preservation system. When SIAM finds error messages, it identifies the relevant information and notifies the system administrators. Thus, the parser tool helps system administrators trace faults in the system and take remedial action immediately. The parser tool also allows users to define the rules to specify the tasks that should be executed if an error occurs. In summary, the parser tool is responsible for: periodically parsing the log files created by the systems and servers on the monitoring host; filtering out error messages or error strings in the log files; and recording detected errors or error strings, and sending warning messages to the system administrators.

3.1.3 Resource management

The data preservation system provides users with a logic space resource to store data. It uses a disk array as a storage space, and SIAM monitors the used capacity of all disk partitions simultaneously. The number of monitoring disks in the SIAM system depends on the size of each disk array. If the used disk space exceeds a pre-defined limit, the system will display a "Warning" or "Critical" signal. Users can customize the threshold value of each used disk space and the displayed signals according to their needs.

3.2 Advanced features of SIAM system

3.2.1 Real-time notification services

In general, Nagios does not support the SRB/iRODS system, and it only provides notifications via E-mail message. In contrast, our monitoring system utilizes various communication protocols to provide a notification service, e.g., e-mail, mobile phones, web pages, and other on-demand services. SIAM can track the status of all systems and servers in the cloud, and detect errors that occur in the data preservation system. It then alerts system administrators via real-time cloud notification services. SIAM enhances the notification mechanism by supporting other real-time cloud notification services, such as Windows Live Messenger (MSN) and Short Message Service (SMS) on mobile phones. As defined in the system configuration file, when SIAM discovers an error message, the monitoring system orders the MSN robot to transmit a message to the Microsoft MSN server via the Microsoft Notification Protocol (MSNP), and send a message to the specified MSN account. It can also use Perl script to send a message to the system administrator's mobile phone. SIAM ranks the levels of error messages by their importance, and sends corresponding notification messages to the system administrators. For example, when a system server fails or shuts down unexpectedly, SIAM dispatches a critical message to the administrator's mobile phone immediately. At the same time, the system administrator will receive an error message vial MSN or e-mail. These real-time notification services help administrators manage their systems effectively.

3.2.2 Fault-tolerance scheme

The SIAM monitoring system works independently of grid systems and provides a fault-tolerance mechanism to improve the reliability of monitoring services. Since the Nagios system does not support such a mechanism, SIAM implements a failover handler as the fault-tolerance scheme, as shown in Figure 2. The failover handler operates as follows. First, all files are backed up from the SIAM master host to the slave host, which is monitored by the SIAM monitoring system, as shown in steps 1 and 2. After installing the MySQL database between the master host and the slave host, both hosts execute their MySQL replication applications via the database. This ensures that the information backed up between the two hosts is consistent (step 3). If SIAM detects a critical fault or the master host fails, the slave host will send notification messages to the system administrators by e-mail, MSN, or SMS (step 4). The SIAM backup file is then decompressed automatically, and the slave host is substituted for the master host. Specifically, the slave host reboots and takes over as the new master host (steps 6 and 7). In this way, the failover handler ensures that 1) the monitoring service is not interrupted; 2) system administrators receive warning messages immediately so that they can take remedial action; and 3) data is not lost in the event of a serious system failure. The use of master and slave hosts results in a lower overhead and enhanced scalability in the distributed system.

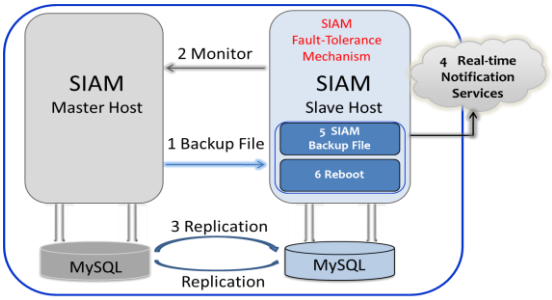

Figure 2. The SIAM fault-tolerance framework

3.3 SIAM system components

SIAM is designed for an SRB/iRODS-based data preservation system. In this subsection, we present the components of SIAM by mapping each one to a specific monitoring phase. The system contains five levels, and utilizes the Nagios Core and the standard interface to display data about various resources, services, and hosts (Figure 3).

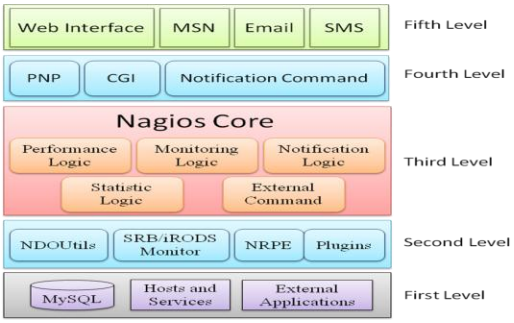

Figure 3. The components of SIAM monitoring system.

Level 1: This level stores the information gathered from hosts, services, and external applications in the database. Thus, it provides a pool of historical and periodical monitoring information, which is stored in the database. Note that the SIAM system, which can be deployed as independent server machine in a distributed environment, records the activities of the server, database, and data preservation system and notifies system administrators when errors occur.

Level 2: Level 2 contains the main applications and communication protocols that support data transfer. To ensure that users can connect to the SRB/iRODS server via a remote network and transfer data, the monitoring application must connect with the SRB/iRODS system by using remote access. Consequently, we adopt the Nagios Remote Plugin Executor (NRPE) to communicate with remotely monitored machines. The monitoring system also utilizes the NDOMOD Event Broker module and the NDO2DB Daemon in NDOUtils, which is an add-on application for the Data Collector. The NDOMOD module extracts data from Nagios and converts it to TCP socket format. The NDO2DB Daemon then stores the data in a database.

Level 3: The third level of the structure is the system kernel. The Nagios Core has five interactive components: Performance Logic, Monitoring Logic, Notification Logic, Statistic Logic, and External Command. Based on the Nagios configuration file for implementing applications, the system ensures that operations conform to the appropriate system settings.

Level 4: This level contains an application interface that communicates with the Nagios Core to control the monitoring services, display monitoring information on web pages, and send notification messages via various communication channels. Here, the monitoring system uses the graphical interface to present system statistics through the PNP application. If the monitored data, such as the CPU load and disk utilization, can be displayed and checked on the screen, system administrators can follow changes in each system via the monitoring service. SIAM enables clients to manage information without installing specific software.

Level 5: The fifth level contains an application interface that communicates with the Nagios Core to control the monitoring services, display monitoring information on web pages, and send notification messages via various communication channels. The system's web page displays the current monitoring information of the entire system. Furthermore, users can use the notification services via a Web interface and receive notifications via e-mail, MSN, or SMS.

4 Experiments

In this section, we describe the experimental environment and implementation of the SIAM system. We also demonstrate the scalability of the components in the monitoring system and evaluate the system's performance.

4.1 Experimental environment

The experimental environment contains a master host, a slave host, and other nodes that are monitored by the master host in a distributed environment. Each monitoring host is equipped with a Xeon 3.50GHz CPU and 4G memory, so its computing power is sufficient to handle a large number of nodes. In a monitoring environment, the hardware of each node includes IBM System x3650, Cisco Catalyst 3750 switch, APC Smart-UPS RT 7500 UPS, and disk array 10TB~80TB. In addition, different types of software, such as SRB, iRODS, and Oracle10g R2, are installed on each node.

4.2 SIAM monitoring system demonstration

In this section, we consider the key functions of the SIAM monitoring system, namely, service availability checks, resource allocation management, system error detection and the failover handler mechanism. As shown in Figure 4, when a provided service in the "th" node cannot be accessed, a CRITICAL message is displayed in the seventh row of the SIAM web page. The user can then click on the hyperlink "th," to obtain more detailed information, such as the service name, error timestamp, and status description (see Figure 5). Figure 6 shows an example of a CRITICAL error message generated by SIAM when monitoring the grid system installed on the "ncl" host. The message is forwarded to the e-mail and MSN accounts of the system administrators via the real-time notification system. Figure 7 displays information about the resources used by the monitored nodes, e.g., the CPU load, disk utilization, and status of the Oracle database.

The SIAM system's user-friendly web interface provides general host information, service information, and other data, as shown in Figure 8(a). It enables system administrators to configure SIAM, and provides important information for users, including the monitoring status of each category, a list of executed applications in the system, and a message and notification history record. If the master host fails to monitor distributed nodes, the slave host automatically takes over the monitoring duties and acts as the master host. To implement the failover handler, the slave host first detects the system's status through the CRITICAL error messages provided by the service availability check function. Next, the system administrators are sent a real-time cloud notification via e-mail, MSN and SMS. Then, the slave host replaces the master host and takes over the monitoring services, as shown in Figure 8(a) and (b).

Figure 4. Services availability checks.

Figure 5. SMS, MSN and E-mail notification dialogue.

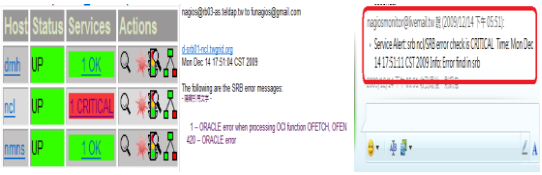

Figure 6. Real-time notification message dialogue.

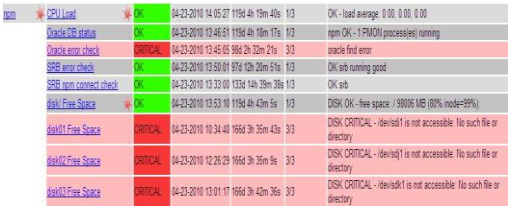

Figure 7. Resource management.

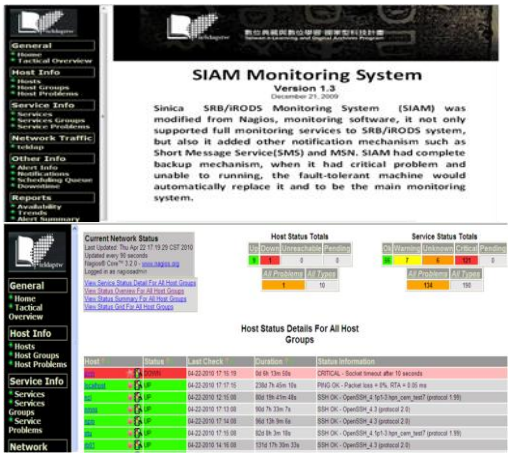

Figure 8. (a) The web page provided by the master host. (b) The web page provided by the slave host after failover.

4.3 Performance evaluation

Several important factors must be considered when evaluating the performance of SIAM, including resource utilization, error notification time, and system robustness. Resource utilization means the amount of the monitoring host's operating space that SIAM occupies while executing tasks. The notification time, which is set by the SIAM system, is the time required to dispatch notification messages when a system server fails. Robustness refers to a slave host's ability to take over from a master host in the event of a failure.

The first experiment considers the resource utilization of SIAM. In the event that the master host generates an unexpectedly large number of CRITICAL messages (such as 50,100, 200, or 300, as shown in Table 2), the operation consumes less than 1.0% of the CPU resources and 0.1% of the memory. In Figure 9, we plot the percentage average CPU usage and memory usage against the number of messages. The graph shows that, although the volume of messages increases significantly, the average resource consumption is relatively small. Furthermore, since monitored nodes use the Nagios NRPE command, SIAM occupies less system space, which in turn reduces the system overhead.

Table 2. The simulation results of resource utilization.

Monitoring Host	Resource Utilization			
	50 Critical Messages	100 Critical Messages	200 Critical Messages	300 Critical Messages
CPU Utilization	0.3%	0.3%	0.5%	0.7%
Memory Utilization	0.1%	0.1%	0.1%	0.1%

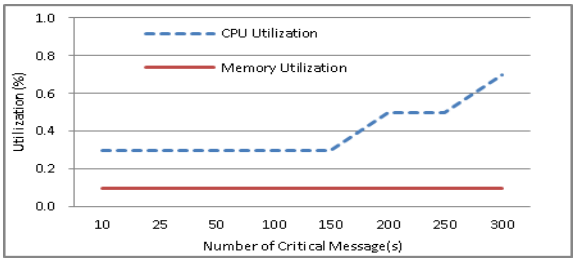

Figure 9. SIAM resource utilization.

The second experiment considers the notification time. One advantage of SIAM is that it can be customized to send out messages at different rates. In this experiment, the monitoring time interval is set at 60 minutes; that is, the system only checks for errors every 60 minutes. When the SIAM host detects a system failure in a monitored host, depending on the severity of the error, SIAM should notify the system administrators as user's required after the failure is detected. For example, if SIAM is configured to notify system administrators within 3 minutes of detecting a failure and the monitoring time interval is 60 minutes, then, in the worst case, SIAM will dispatch an alert message 62 minutes and 59 seconds (3779 seconds) after an error occurs. In the best case, where an error occurs exactly at the beginning of an interval, SIAM will respond within three minutes (180 seconds). When SIAM dispatches an alert, system administrators are notified immediately (in less than one minute) via real-time cloud services, such as MSN, SMS and e-mail. Generally, these three methods are equally fast. SMS is particularly convenient since mobile devices are portable, so administrators can receive notifications anytime, anywhere. Table 3 summarizes the notification times and notified service transfer times in the experiment.

The third experiment examines the robustness of the failover scheme in terms of the failover time (Table 3). Here, we assume that the monitoring time interval is ten minutes,

and the degree of urgency of the error message is set at level A. In SIAM, system damage is classified into three levels (A, B, and C) with A being the most critical level. If the SIAM slave host detects the failure of the SIAM master host, it will notify the system administrators within three minutes. Meanwhile, the slave host will take over the monitoring services of the master host and reorganize the monitoring system. In our experiment, the entire process took approximately twelve seconds (system failover time). Overall, the results of the three experiments show that the SIAM monitoring system is robust and it can provide real-time notifications with a low overhead.

Table 3. Simulation of the SIAM system under various execution scenarios.

Experiment	Execution Scenarios	Time (seconds)
Notification Time	Worst condition	3779
Notification Time	Best condition	180
Transfer Time of Notified Services	MSN, E-mail, SMS (average)	7
Failover Time	Level A of error message	12

5 Conclusions and future work

In this paper, we discuss how SIAM enhances the monitoring mechanisms of the SRB/iRODs system by incorporating several open source components of Nagios. We find that SIAM provides clients with more flexibility and greater control over their monitored networks because it offers both real-time notification services and a fault-tolerance mechanism. SIAM utilizes multiple communication modes when alerting administrators of system errors, all of which are customizable according to their severity. SIAM adds functionality without adding unnecessary infrastructure.

Furthermore, the SIAM system's fault-tolerance mechanism helps prevent data loss in the event of a serious system failure. The use of master and slave hosts results in lower overheads and enhanced scalability of the distributed system. Combined, these features ensure not only this minimized overhead and lower costs, but also increased robustness and alert administrators. In the future, we will explore more advanced applications of the SIAM system, including security auditing and resource performance tuning. We also plan to extend the functionality of SIAM to fit with other data transfer systems or protocols and incorporate additional cloud computing services.

6 References

[1] J. Gantz, C. Chute, Al. Mafrediz, S. Minton, D. Reinsel, W. Schlichting, A. Toncheva, "The Diverse and Exploding Digital Universe: An Updated Forecast of Worldwide Information Growth through 2011," white paper, International Data Cooperation, Framingham, MA, 2008.

[2] SRB: http://www.sdsc.edu/srb/index.php

[3] iRODS: http://www.irods.org

[4] C. Baru, R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," CASCON'98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada.

[5] R.W. Moore, "Managing Large Distributed Data Sets Using the Storage Resource Broker," ITEA Journal of Test and Evaluation, 2007.

[6] S. Zanikolas, R. Sakellarios, "A taxonomy of grid monitoring systems," Future Generation Computer Systems, vol. 21, 2005, pp. 163-188.

[7] A. Cooke, A. Gray, W. Nut, A. Cooke, A. Gray, W. Nutt, R. Cordenonsi, R. Byrom, L. Cornwall, A. Djaoui, L. Field, S. Fisher, S. Hicks, J. Leakey, R. Middleton, A. Wilson, X. Zhu, N. Podhorszki, B. Coghlan, S. Kenny, D. O'Callaghan and J. Ryan, "The relational grid monitoring architecture: mediating information about the grid," Jounal of Grid Computing, vol. 2, Dec. 2004, pp.323-339.

[8] X. Zhang, J. Freschl, and J. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems," In Proceedings of the 12th IEEE International Symposium on High-Performance Distributed Computing (HPDC-12), June 2003.

[9] S. Zdenko, R. Branimir, "Monitoring systems: Concepts and tools," 2004.

[10] Nagios:http://www.nagios.org

[11] Ganglia: http://ganglia.sourceforge.net.

[12] Group, T. C. CACTI: http://www.cacti.net/what_is_cacti.php

[13] Hawkeye :http://www.cs.wisc.edu/condor/hawkeye

[14] M.L. Massie, B. N. Chun, D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," Parallel Computing, vol. 30, 2004, pp. 817-840.

[15] S. Andreozzi, N.De Bortoli, S.Fantinel, A.Ghiselli, G.Tortone, C.Vistoli, "GridICE: a monitoring service for the Grid," Future Generation Computer Systems Journal, 2005, pp. 559-571.

[16] E. Imamagic, D. Dobrenic, "Grid infrastructure monitoring system based on Nagios," High Performance Distributed Computing Proceedings of the 2007 workshop on Grid monitoring table of contents, 2007, pp. 23-28.

[17] E. Galstad, NRPE Documentation

[18] Galstad, E., NDOUTILS Documentation Version 1.4

[19] PNP4Nagios.,Retrieved from http://docs.pnp4nagios.org/pnp-0.4/start

[20] RRDtool: http://oss.oetiker.ch/rrdtool

[21] DARB : http://rempte-backup.teldap.tw

[22] Tsung-Tai Yeh, Hsin-Wen Wei, Shin-Hao Liu, Pei-Chi Huang, Tsan-sheng Hsu, Yen-Chiu Chen, "The Development of Digital Archives Management Tools for iRODS," Proceedings of iRODS User Group Meeting 2010.

[23] TELDAP: http://www.teldap.tw/en/