Abstract. Information systems (IS) today vary widely in purpose, size, architecture, and criticality. If a team has to monitor and maintain multiple IS, it should divert to centralized management system. That kind of system, based on Java Management Extensions (JMX) technology, is presented here. This paper shows how the monitoring of the third-party components is achieved. It presents main monitoring module together with its operating modes, and it gives figures representing usage of the system. Also, the future work is discussed.

Keywords. IS monitoring and management, IS maintenance, IS administration, JMX.

1. Introduction

Information systems today vary widely in purpose, size, architecture, and criticality. There is very little that they have in common across all application types, and application architecture trends are increasing this diversity. Adding to the complexity, applications are executing across multiple hosts, operating systems and business environments.

There are many reasons for investing in centralized management systems [5]:
- too many resources to be managed,
- difficulty to control resources from a single point,
- resources are of diverse types, requiring multiple interfaces,
- most resources require high availability.

During development of featured information systems little attention was given to maintenance and manageability.

The responsibility of the system monitoring was put in the hands of operators, while each operator was specialized for her own part of the system.

As the system grew, the need for centralized management became evident as infrequent incidents of system failure occurred. It should be possible to manage it through centralized management system, even for the operators without profound knowledge of certain parts.

Since most of the modules of these systems are written in Java, Java Management Extensions (JMX) was a logical management technology of choice for building the management system presented in this paper.

2. The monitoring challenge

Monitoring and maintaining multiple IS are primary duties of our team. Typical IS is based on centralized relational database management system (DBMS) on a dedicated database server. Web users connect to enterprise applications on application server through web server. Application server authenticates users and communicates with DBMS in secure network.

Desktop Java applications connect to DBMS through Firewall and Encryption Tunnel (FET) Java application for security and data encryption. Users are then authenticated on DBMS.
Support system for reports and analysis is based on parallel DBMS. It can be used only by web users through its own application server using the same web server mentioned previously. Various service applications connect to DBMS-s in the secure part of the network for integration and interoperability. All users are authenticated using LDAP service.

In order for system to be available for users, various different modules and systems have to work properly. Monitoring of the whole system is the only way to assure its availability and performance.

3. JMX

JMX is an isolation level between the applications and management systems. It offers standard interface to customer management systems for Java application developers, just as Java Database Connectivity (JDBC, [4]) does for databases. It also sets the stage for more sophisticated, intelligent and complete management systems.

Main influencing forces for the success of JMX are [6]:
- having a simple API for development,
- ensuring enough information for management systems to manage the resource "generically",
- providing architecture to support the management of diverse, dynamic applications.

The JMX architecture is based on three component types: managed resources and MBeans that comprise the instrumentation level, agents that comprise the agent level and adapters that represent the manager level [3].

JMX managed resources are instrumented using MBean objects. JMX specifies four types of MBeans: standard, dynamic, open, and model type. Application instrumentors use or implement one or more of these types, exposing their management interface.

While developing this management system, only Standard MBeans were used, which allow registration of any Java bean with the JMX agent.

By using Standard MBeans it was easy to expose management data in our applications to the management system. However, non-Java applications and services needed development of additional service layer (Adapter for non-JMX components, shown on figure 2) that would gather data, process it, and expose it through JMX to the management system.

Agent and manager levels of the JMX architecture are comprised in a module called Supervisor. It is responsible for gathering data from exposed MBeans, presenting collected data in a standard manner, and reacting and alerting if necessary.

4. Adapter for non-JMX components

Adapter for non-JMX components (ANJC) is a Java application that collects information on states. It also collects information on availability and service load for those parts of the system that could not be monitored using Java technologies employed in module Supervisor. ANJC periodically queries monitored resource, and stores management data in MBeans. It runs on a single server and monitors multiple non-Java applications and services.

ANJC monitors assigned resources through jobs and triggers. Jobs are written in Java as a part of ANJC module, and each has the logic and knowledge implemented for monitoring and managing its respected resource. Gathered data is exposed to module Supervisor using JMX.
ANJC's configuration file defines the list of jobs and parameters for job's triggers that run on ANJC's startup. The trigger is ANJC's scheduler that starts designated job at predefined frequency. In order to change the trigger parameters or list of desired jobs, ANJC has to be restarted.

Most of ANJC jobs are monitoring of DBMS and web applications, and they use JDBC for connection. They check the availability and the state of DBMS and certain databases by querying system and user tables, and checking for system health and related business logic. Web applications are monitored by querying the websites using HTTP and HTTPS requests. Returned data is compared to the expected results defined in ANJC.

Currently module ANJC monitors multiple web servers (Orion, IIS), Active Directory and LDAP service, Informix database server and MS SQL Server.

5. Monitoring IS with JMX

Due to the fact that the information system is supervised by monitoring its modules, each of the important system components must provide monitoring features. In case of monitoring multiple information systems, all components should expose data that need to be monitored.

Implementation of the management system that monitors multiple information systems presented in this paper is related to systems built using Java technologies. All monitored components expose their standard MBeans through JMX technology.

5.1. Data gathering and storing

All MBeans are periodically fetched into the Supervisor, the central monitoring system, from the monitored system components. The basis of the Supervisor architecture is the Agent. The Agent is a logical presentation of MBean's data. The data contained in one MBean is presented as a set of attributes of one or more agents. If MBean exposes only some simple collection of the data, e.g. concerning the current state of the application, then these data will be joined in one agent and presented as its attributes. If MBean exposes data that is organized in a more complex object graph, then fetched data will be presented as attributes of more agents. In this case, each MBean is still presented through one agent, with one or more subagents attached. Each of these subagents can have subagents of their own, and so on, recursively. While fetching data for this kind of MBean-agents mapping, all data is fetched at once, and distributed to the agent and the subagents subsequently. This is the only situation throughout the monitoring system where difference is made between the agents and the subagents. Otherwise, subagents are considered to be agents, having collections of attributes and providing them to the other system facilities.

5.2. Supervisor operating modes

The Supervisor is able to operate in three modes: server application, desktop application or adjusted version of desktop application – screensaver mode.

The purpose of the server application mode is automated monitoring of the systems. The configuration parameters define which agents to fetch from services and which attributes to monitor. For each monitoring attribute, value alert thresholds are also defined. If attribute value is outside normal boundaries, the Supervisor sends various warning messages. Except for the alerts, this operating mode also provides reporting. Each report is described in a separate class, with reporting period and sending mode defined through configuration parameters.

The desktop application mode provides direct monitoring with user interaction. Each agent defined in the Supervisor has corresponding graphical user interface (GUI), integrated in the application together with other agent GUI. The agent GUI displays time of the last data retrieval and the values of all attributes. Attribute values can be shown in correlation to each other or using time series graphs. Each graph is bound to the agent that provides attributes for its creation. One agent can have multiple graphs associated with it.

The purpose of the screensaver mode is to provide visual data on systems monitored through demonstration panels or displays that are set in maintenance staff facilities. This working mode does not feature user interaction. Instead, it subsequently displays all graphs that are defined in the system, regardless of the agent they are associated with.

The implementation of a supervising system that can run in three different modes means that the architecture is somewhat specific. System components can be viewed through the mode for which they are providing (Figure 3).
Figure 3. Supervisor components

There are some components that are common to all operating modes, referred to as core components. Apart from the general implementation of the agent and subagent components, they contain data representation, basic graphs, reports and various utilities (e.g. initialization procedures). Since the desktop application mode and the screensaver mode are both GUI oriented, there are additional components that are common to these operating modes. These components include implementations of graphs and other GUI structures.

Depending on the operating mode of the system, data fetching also varies. In the server application or the screensaver mode, without user interaction, all data is fetched automatically from services, in periods of time predefined for each agent. The desktop application allows user interaction. Data is fetched automatically only once, while starting application. Every other data fetch is a result of an explicit user request.

5.2.1. Server application mode

Regardless of the operating mode, the basic action unit of the supervising system is a data fetch for the agent and a logical distribution of it to the agent attributes. Varying on the operating mode, data fetch listeners are added to each agent.

While operating in the server application mode, every agent’s data retrieval will result in the activation of the listener – SupervisingDataChecker, which will compare actual values of observed attributes with predefined thresholds and set them in corresponding categories (Figure 4). MessageManager is a component that decides which messages will be sent, and keeps a list of sent alarm messages. MessageManager's decisions on whether or not the message should be sent, and the decisions on the type of the message are based on the input from the SupervisingDataChecker and list of the sent messages. The message sending logic does not allow the same message to be sent out all over again. Instead, similar messages are grouped together during a period of time, after which summary report is sent. The period is lengthened if the same kind of messages keep emerging. If the attribute value changes from one level to another (e.g. warning to critical), periods are reset, and messages are sent according to the new level policy. Messages that are to be sent out of the system are passed to the MessageQueue, which does the actual sending. The way of sending messages is defined as a configuration parameter, and triggered by the independent scheduled job (e.g. cron job).

Figure 4. Supervisor operating in server application mode

Periodical reporting is also triggered by the system cron job. Reports are shaped as messages that are added to MessageQueue and sent the same way as the alarm messages are.

5.2.2. Desktop application mode

In this mode data retrieval is managed by the user. Listeners are bound to each agent, so when agent gets the new data, all graphs that are
associated with it are refreshed (Figure 5). This is done by the GraphFactory component, which creates graphs that are to be displayed. The type of each graph and data it represents is predefined in the corresponding Graph component. After data retrieval, the agent attributes are displayed in a data panel, while graphs are in a separated graph panels, all set up in this agent’s GUI.

5.2.3. Screensaver mode

Data retrieval is triggered periodically through a cron job. This mode uses the same listeners as the desktop application mode, so graphs associated with the agent are refreshed through the GraphFactory after data retrieval (Figure 6). These graphs are not displayed immediately, but reside in the common graph collection. The DisplayManager component takes care of the current display on the viewing screen. Triggered by another cron job, it selects one of the graphs from the collection and displays it. All cron job periods are defined as application parameters.

5.3. Expanding Supervisor’s monitoring scope

The quality of the information systems monitoring through the Supervisor is based on understanding the semantics and the meaning of the data being observed. New IS module to be monitored results in expanding the Supervisor with new agents, intended only for that module. With those agents, new graphs could be designed. Those graphs can be built only if the meaning of the attributes is known. For operating in the server application mode, configuration parameters for the agent should also be added, as well as for its attributes. New reports that utilize these agents should be made, or existing ones expanded. These actions also imply understanding data semantics.

6. Monitoring in use

The components for monitoring multiple information systems presented here are currently used to monitor two IS: Higher Education Information System (HEIS) [1, 2] and Mozvag [7]. Together, there are three DBMS, three application servers, two firewall applications,
web server, and a dozen server applications to monitor. Currently there are more than 20 agents representing data from various MBeans, with the same amount of graphs and several reports, and two instances of the Supervisor running, one in a server mode and the other in a screensaver mode. Eight team members are using the Supervisor as a desktop application to determine the operating level of the monitored modules and components.

7. Future work

The next step in monitoring and maintaining IS should be to enable module management directly through the Supervisor. While operating in desktop application mode, the user should be able to react to the agent states and control the associated module at the same place. To enable this feature, each agent should also provide the means to invoke operations exposed in MBean whose data it represents.

If operating in the server application mode, the Supervisor should provide the means of the automated management. Some actions could be predefined, e.g. bringing the service online, or restarting a service task. These actions could then be associated with the alarm states, e.g. if a service is offline, or a service task is stopped, an appropriate action should be executed.

During the execution, the Supervisor is recording a substantial amount of statistical data for each agent, which is primarily used for creating various graphs that represent a change of attribute values through time. However, this statistical data is not preserved when the Supervisor is shut down. In order to display quality graphs immediately after starting the application, it is necessary to make this statistical data persistent.

8. Conclusion

In this paper, the way of monitoring several information systems, from the point of view of the maintenance team, is presented. Main issues considering IS monitoring and maintenance are considered. It is described in more detail how it can be achieved to monitor third-party system components, such as DBMS, application servers, etc. Program module for monitoring through JMX technology is displayed, its three operating modes described. For enhancing operability, future work is presented.

9. References