ASYNCHRONOUS AGENT SYSTEM FOR MONITORING COMMUNICATION AND SYSTEM STATES BASED ON THE SOA PARADIGM

Michał Niedźwiecki
Department of Computer Science
University of Science and Technology
Al. Mickiewicza 30, 30-059 Krakow, Poland
Email: nkg@agh.edu.pl

Krzysztof Rzecki
Institute of Telecomputing
Cracow University of Technology
ul. Warszawska 24, 31-155 Krakow, Poland
Email: krz@mars.iti.pk.edu.pl

Piotr Błaszczyk
Department of Computer Science
University of Science and Technology
Al. Mickiewicza 30, 30-059 Krakow, Poland
Email: blaszczy@agh.edu.pl

KEYWORDS

AGENTS, SOA, MONITORING, LOGGING, VISUALIZATION

ABSTRACT

The implementation of the concept based on the SOA paradigm appeared to be significant problem associated with the issue of monitoring and visualization of communication between the various elements of the system during the execution of business processes. The monitoring process as a collection of messages (logs) is a technique known and used. However, collate and read what the state and what messages leading up to it give you relevant information. Ways of gathering such information, especially visualization of the communication with their advantages, will be described in this article as new. This will be the first to create a universal, expandable, multiagent platform to collect information about events and history of the state changes of distributed components. This solution keeps information about the real, global sequence of events occurring on different machines (not necessarily synchronized with clocks) and connections between them.

INTRODUCTION

Computer systems monitoring tools can be classified into categories: hardware monitoring, operating system monitoring, local application moni-toring, network resources and traffic monitoring, communication and states monitoring in distributed systems. In the last category there is not too many available solutions. The most popular are central logging systems for distributed events, where no synchronization mechanisms, no time graphs and no SOA systems monitoring abilities are available.

In discussed solution greatest emphasis was put on true events sequence mapping, that took place on different machines and their mutual connections. One of the main problems to be solved was taking differences in time indicated by clocks running on different computers. Moreover, these differences must be constantly adjusted, in case of manual or automatic time change on the monitored computer.

EXISTING SOLUTIONS

Monitoring equipment depends on the producer of the equipment. Certain elements, such as CPUs load and temperature, free and used memory of the system and disk and state of the network load are common features.

Tools for monitoring the operating system are also depends on the specific operating system. In Windows NT based systems there is a tool called Task Manager (Smith and Komar, 2005). It is used to view information about running processes and occupied resources at the time, such as the CPU usage, cache used, the number of threads and opened files. In Linux systems, there are similar tools such as top and htop (Negus, 2007) programs. There are also many commercial solutions that provide more complex functionalities. One of these is Process Explorer from Sysinternals (Process Explorer, 2010). This program provides information about the processes call tree and dynamic libraries loaded into memory.

A tool to monitor a local application in the Windows NT based systems is Event Log (Smith and Komar, 2005). The Event Log consists of the Application Log, Security Log and System Log. Application Log collects information about the events that occurred in the application (eg application errors). Security Log informs about successful or unsuccessful attempts to log on and information about events associated with the resources usage (such as creating, opening, or deleting files). System Log collects information about events that occurred in the components of the system. If the system started up incorrectly, it will be stored in this log. A similar solution in Linux systems is syslog (Negus, 2007).

Tool for monitoring traffic and network resources is the Netstat program occurs both in Windows NT (Smith

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

and Komar, 2005) and Linux (Negus, 2007) system. This program provides information about opened network connections and ports on which the program listens. It can also display the Ethernet statistics for the different interfaces and protocols. In Linux based systems, there is (Negus, 2007) tcpdump program that is used to view the content of the information transmitted over the network. However, Monit (Monit, 2011) program periodically checks whether the network services, such as http server, ftp, mysql etc. are available. Such monitoring involves periodic checking, if the page displays correctly, or if the connection attempt with a specific service was successful. In case of failure it informs the administrator.

Distributed systems usually operate under the control of an appropriate middleware for sending messages to the system components. This layer is made to monitor communications and resources of distributed systems. An example of a solution providing such a layer is MPI (Message Passing Interface) (Quinn, 2003) and PVM (Parallel Virtual Machine) (Geist et al., 1994).

An example of software for logging system events within a single platform is Linux Syslog. Syslog is proper for logging events occurring on one computer. It records events with an accuracy of one second and one microsecond since the start of the operating system. Using a plug, syslog logs can be used as a source of information about events, which will be processed by the discussed project. At this moment, the elements of a distributed system communicate with the logging system using appropriate libraries.

One of solutions is to use the ESB bus for SOA (Masternak et al., 2010), which is an additional intermediate layer in the multilayer information systems architecture that allows to use this concept in the corporate environment. It allows dynamic attachment and removal of services, which are the part of the corporate information system. ESB bus mediates in transferring messages between services, which gives it the opportunity to precisely monitor and log information about events. ESB bus is implemented in such application servers like GlassFish ESB (OpenESB, 2010), IBM WebSphere ESB (IBM Corporation, 2010) and Oracle ESB (Davies et al., 2008). Exception messages monitoring they support, the entire transaction monitoring.

In this case, the problem the monitoring system depends on the ESB bus. Not all systems use this bus, and using it in existing systems would cause too much cost of rebuilding the system. Moreover, what about the components outside the bus?

Nowadays, solutions of monitoring connections between services are limited to storing information about the sender and recipient and the transaction ID.

ASYNCHRONOUS MONITORING AGENT SYSTEM

Discussed solution should provide history presentation of time synchronized components states in a distributed system and messages sent between them with a recognition of their relationships. It allows for exact analysis of situations which are a “bottleneck” in the system and led to the failure or may be helpful to look for methods of system optimization. This solution is flexible and extensible. Using the appropriate plugins it allows integration with systems using different technologies.

The system consists of the following elements:

- Log Server,
- Log Agent,
- Log Visualizer.

Log Server is a central system storing information about events. It ensures exact representation of the sequence of related events. The task of the server is also calculating difference of clocks between the Log Server and monitored machine hosting the application. To determine this difference Log Server is using NTP (Mills et al., 2010) protocol, set up between the Log Server and the Log Agent server in oposite to (Cetnarowicz, 1996) that presents centralized system.

Log Server has modular structure. It has the ability to attach external plug-ins that can be used to interpret certain special events.

The Log Agent is a program placed on monitored place. Its basic functionalities are:

- time synchronization with Log Server,
- communicate to get information about events,
- communicate to download program code to extend itself functionalities.

However Log Agents are individuals, the Log Server is controlling their work.

Log Visualizer is an application which connects to Log Server and retrieves information about the events. It has the ability to view history state of each of the monitored components separately or present entire business process on one sequence diagram.

The Prototype Of The System

The first Log Server prototype was a synchronous solution (Figure 1 and 2). It caused a performance decrease of the monitored system. The reason for that was the mechanism in which monitored component - using appropriate libraries - was connecting directly to Log Server, by component blocking for the time of message sending. Additionally, Log Server could receive only one message at one time. It ensured keeping real events occurring, which was an asset solutions. The system was used for implementation and testing of a prototype of an emergency action management system. This system was developed as an EU project “New information technologies for the electronic economy and information society based on the SOA paradigm” (POIG.01.03.01-00-008/08) (Cetnarowicz et al., 2010). Log Server was found perfect for this task, because all computers were in one subnet, by that overhead of Log Server was minimal.

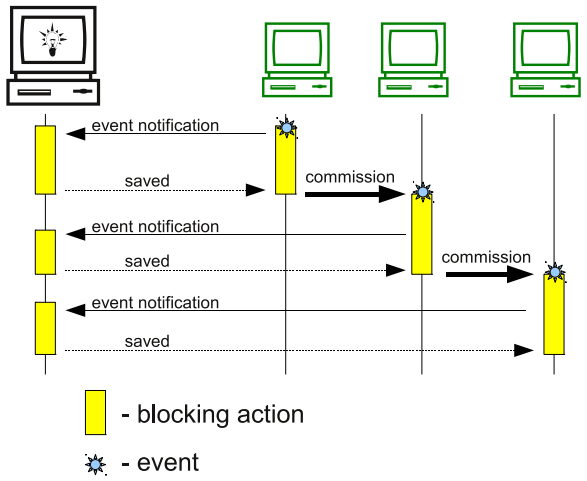

Figure 1: Prototype Log Server work synchronously.

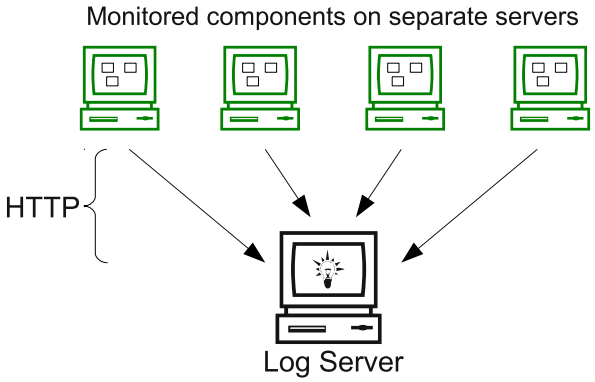

Figure 2: Log Server prototype.

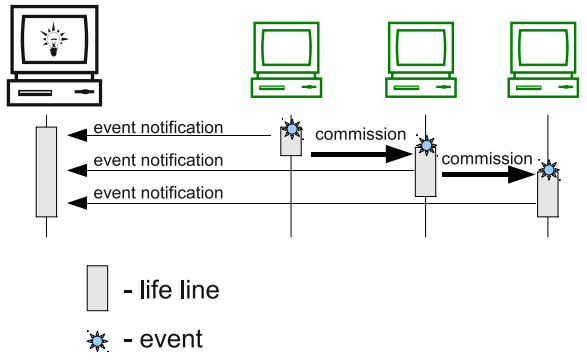

Figure 3: Asynchronous, non-blocking Log Server model.

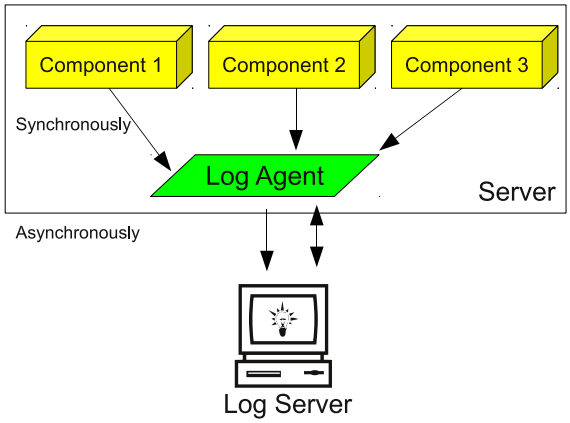

Figure 4: Log Agent.

For larger and more distributed systems it is not sufficient. There is a risk that, in case of insufficient network link bandwidth or Log Server failure, the entire system will be blocked. It was the reason to balance the efficiency and make it independent from the monitoring system. Log Agent helps to fulfill this function. Log Server desirable action outcome should be non-blocking asynchronous messaging as in Figure 3.

Log Agent

Log Agent is an application running in the background on the monitored server (Cetnarowicz, 2009). Monitored component is not connected directly to Log Server (Figure 2), but to Log Agnet (Figure 4).

Connections with Log Agent are in synchronous, as before with Log Server. However, the component blockade is set just for the time of short message sending from the container to the Log Agent and note the exact time of this event. The rest of the work the Log Agent is doing in a separate process. With this solution, there is messages registration sequence maintained within a single machine, with a minimal system slow down.

Clock Synchronization Problem

Log Agents another task is to send collected logs to Log Server. During this task occurs clocks synchronization problem.

For example (Figure 5):

1. Application on server A sends a message to server B and notify its Log Agent about sending a message.
2. After a second server B notify its Log Agent about receiving a new message from the server A.

If the clocks on both servers are synchronized, the problem will not occur. However, when the clock on the server A is late relative to the clock on server B for longer than one second in this case, the result is a record that first server B has received a message from server A, then server A sent it to the server B. So we have a distortion of events sequence.

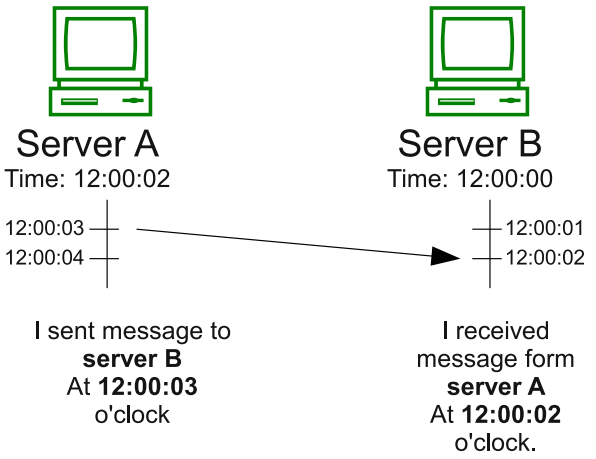

Figure 5: The problem of synchronizing clocks: Server B read the message before the server A has sent it to him.

There are various ways to manage witch such distortions. One of them is the constant clock synchronization. This is possible, only if all the servers belong to one corporation and are located relatively close to each other. If the servers are located far from each other or belong to other corporations, then technical and political barriers occurs. Technical barrier are the time lags in the exchange of information between computers caused by physical distance and number of nodes through which the packet of information has to go. Wave propagation speed in electrical cables is finite, each node also generates an additional delay. Political barrier is the fact that computers belong to various corporations for security reasons do not allow the time synchronization or they use different solutions which generate significant discrepancies.

In situations when accurate clocks synchronization cannot be applied, sequent numbers can be used (Figure 6). By sequent numbers, sequence of occurred events can be clearly determined. Sequent numbers are implements in communication protocol layer (eg. TCP implements them). The problem is that the TCP protocol is a lower layer protocol and the higher layer protocols which are being used by other components, usually does not implement such elements.

Moreover, such mechanism does not protect against the following situation:

1. Server A sends message to server B.
2. Server B receives message from server A.
3. Server B sends message to server C.
4. Server C receives message from server B.
5. Server C sends message to server A.
6. A server receives message from the server C.

It is known that one server must send first a message to the other, so the other one could receive it. It is not known whether, eg. sending a message from the server B to C was a consequence of sending messages from server A to B, it is why events sequence distortion may appear.

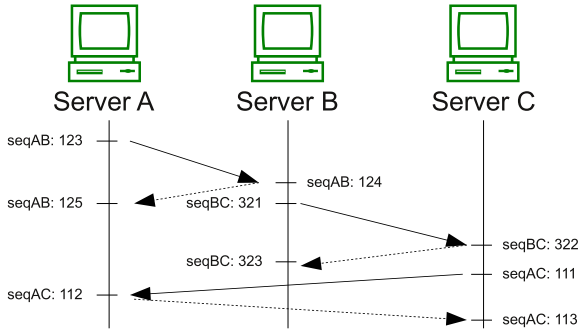

Figure 6: Sequence numbers of messages.

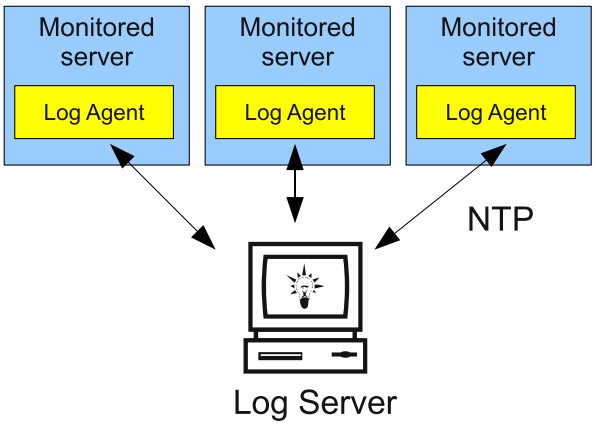

Figure 7: Diagram of time synchronization between Log Server and Log Agents acting on the monitored machines.

Applied Solution

The solution, which was used in the described system is that in the central events register (Log Server), to the time taken by the Log Agent is appended the time difference between the Log Server and the server on which Log Agent is located.

This difference is appended to each event, because the difference in times of Log Server and the server running the Log Agent may change over time.

This difference is calculated in two ways. The first way is a periodic communication with Log Agents by using the NTP with the algorithm used with this protocol to determine as precisely as possible the time difference. If the network settings do not allow to use directly NTP (Mills et al., 2010) application (Figure 7), is used a similar algorithm to designate times difference, but using TCP (Postel, 1981) protocol which may be less accurate.

The second way is correcting the previously calculated difference by errors detecting in the interpretation in the directly linked events sequence, such as the time of sending messages from server A to server B and time of receiving it by server B. After that type of incident, time difference is adjusted to such value which can simultaneously decrease these distortion and not creating new ones (eg. Between servers B and C).

Log Server

The Log Server is a central database to store gathered information about events with account of time differences detected by Log Agent.

The functionality of the Log Server is extendable by using plug-ins mechanism. The plug-ins task is to analyze information (received from Log Agent) about events and relationships detected between events that occurred in various components.

Links Detection

By detecting relationships between events, it is possible to review back events history following related messages.

The Log Server recognizes only the link between sending and receiving message. Log Server doesnt analyze, if a message sent from server B to server C is a consequence of receiving by this server message from server A, whether it is a completely separate action. Log Server provides the ability to attach an external plug-ins that analyze sending messages in purpose to extract information which connect them with a particular action.

SUMMARY

Above there was presented the concept of asynchronous agent system for monitoring, communication visualization and system states based on the SOA paradigm.

Current research suggests, that further implementation is applicable and will be performed.

It is expected to achieve a high universality solution (thanks to the plug-ins system implementation) with minimal network throughput and CPU time monitored machines at the same time (thanks to asynchronous).

Solution presented in this article can be used as universal monitoring system for components state and communicates send between them in SOA systems on heterogeneous platforms. This conception can be used for remote distributed backup. The new work with it is about monitoring dirty power in industrial power distributed system.

Future work is to perform full implementation of agent platform to work on different platform. There must be also prepared extension to monitor events through ESB and SOA-ent (Cetnarowicz et al., 2010).

ACKNOWLEDGMENT

The authors would like to thank the Institute of Telecomputing at Cracow University of Technology, Cracow, Poland for providing the resources and facilities. Author Michał Niedźwiecki would like to thank EFS of POKL 4.1.1 (POKL.04.01.01-00-367/08-00), European Union programme for support.

REFERENCES

Cetnarowicz, K. (1996). M-agent architecture based method of development of multiagent systems. In *Proc. of the 8th Join EPS-APS International Conference on Physics Computing, ACC Cyfronet*.

Cetnarowicz, K. (2009). From algorithm to agent. In *ICCS (2)*, pages 825–834.

Cetnarowicz, K., Dyduch, T., Koźlak, J., Żabińska, M., Błaszczyk, P., Niedźwiecki, M., Rzecki, K., Belava, L., Wachocki, G., Bech, P., Dziedzic, J., and Ptaszek, M. (2010). *SOA Infrastructure Tools - Concepts and Methods*, chapter SOA-Based Multi-Server Agent System — Application for Integrated Rescue Action, pages 471–488. Poznań Univesity of Ecenomics Press.

Davies, J., Schorow, D., Ray, S., and Rieber, D. (2008). *The Definitive Guide to SOA: Oracle Service Bus*. Apress, Berkely, CA, USA, 2nd edition.

Geist, A., Beguelin, A., Dongorra, J., Jiang, W., Manchek, R., and Sunderman, V. (1994). *PVM: Parallel Virtual Machine—A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, MA.

IBM Corporation (2010). *Using WebSphere Enterprise Serial Bus*. IBM Corporation e-book.

Masternak, T., Psiuk, M., Radziszowski, D., Szydło, T., Szymacha, R., Zieliński, K., and Żmuda, D. (2010). *SOA Infrastructure Tools - Concepts and Methods*, chapter ESB — Modern SOA Infrastructure, pages 17–46. Poznań Univesity of Ecenomics Press.

Mills, D., Martin, J., Burbank, J., and Kasch, W. (2010). Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Proposed Standard).

Monit (2011). Monit home page. http://mmonit.com/monit/. accesed February 2011.

Negus, C. (2007). *Linux Bible 2007 Edition: Boot Up to Ubuntu, Fedora, KNOPPIX, Debian, SUSE, Ubuntu, and 11 Other Distributions (Bible)*. John Wiley & Sons, Inc., New York, NY, USA.

OpenESB (2010). Openesb home page. http://wiki.open-esb.java.net/. accesed February 2011.

Postel, J. (1981). Transmission Control Protocol. RFC 793 (Standard). Updated by RFCs 1122, 3168, 6093.

Process Explorer (2010). Process explorer home page. http://technet.microsoft.com/pl-pl/sysinternals/bb896653. accesed February 2011.

Quinn, M. (2003). *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill Science/Engineering/Math, 1 edition.

Smith, B. and Komar, B. (2005). *Microsoft windows security resource kit, second edition*. Microsoft Press, Redmond, WA, USA, second edition.

AUTHOR BIOGRAPHIES

Michał Niedźwiecki received the M.Sc. degree from Cracow University of Technology, Cracow, Poland. Currently he is a Ph.D. candidate at AGH University of Science and Technology, Cracow, Poland. His research interests are agent systems

and service oriented architecture. His email is nkg@agh.edu.pl and his personal webpage at http://galaxy.agh.edu.pl/~nkg/.

Krzysztof Rzecki received the M.Sc. degree from AGH University of Science and Technology, Cracow, Poland, and the Ph.D. degree from The Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences, Gliwice, Poland. Currently he is an adjunct at Cracow University of Technology, Cracow, Poland. His research interests are low-level network services and service oriented architecture. His email is krz@mars.iti.pk.edu.pl and his personal webpage at http://krz.iti.pk.edu.pl/.

Piotr Błaszczyk received the M.Sc. degree from Cracow University of Technology, Cracow, Poland. Currently he is a Ph.D. candidate at AGH University of Science and Technology, Cracow, Poland. His research interests are microprocessor applications, mobile and agent systems. His email is blaszczy@agh.edu.pl and his personal webpage at http://paku.oz.pl/.