Monitoring System for Cloud Services in Distributed Architecture

Sohit Shukla, Neelendra Badal
Department of Computer Science & Engineering, KNIT, Sultanpur, India

Keywords: Cloud Services, Distributed architecture, Monitoring system

Abstract: Services over the cloud are the key emerging features of the Internet world, which will facilitate a virtual cloud service in a distributed architecture. These services are tremendously increasing with time. The increase in the service will be led to a huge number of real-time tasks in progress that will utilize many site facilities, data, and networks. To check the performance and the accuracy there is a various monitoring system which will provide the data to be analyzed for the improvement. We have presented the main aspects of these monitoring systems for cloud services in a distributed system that has been designed for the services in the virtual environment.

1 INTRODUCTION

A huge number of independent resources mutually connected through a network and virtually available in the distributed environment creates a cloud. Clouds are what when anyone sees in the sky then they observe that there is something which equally lies above us and available to us as it to others. In the same way, the cloud provides its services in the form of a platform, infrastructure, and software if categorize broadly and further extended to everything as a service. These resources which were combined in a virtual environment to create a service cloud have different features & requirements compare to the traditional resource network. Cloud service in distributed architecture has a federation of the clusters of distributed hosts these hosts may have diversity in the factors of the infrastructure.

For monitoring the cloud service architecture one has to make the consideration of the diverse factors involving the cloud infrastructure. Cloud computing refers to the platform for the distributed computing this leads to the distributed service architecture work as the platform for the cloud service architecture.

2 MONITORING SYSTEMS

2.1 Monitoring the Agents in a Large Integrated Service Architecture (MonALISA)

Newman H.B. et al had proposed a model which was based on dynamic distributed services architecture (DDSA) (Newman H. B. et.al, 2001) known to be as "Monitoring the Agents in a Large Integrated Service Architecture" (MonALISA) (Newman H. B. et.al, (2003).

WSDL/SOAP technologies & JINI/JAVA were used to implement. The DDSA inherently distributed, self-restarting, and loosely coupled, compiled them to robust & scalable. In their work to manage and optimize the workflow through the data grid composed of the various servers located at different sites having several computing & storage units, they had created a framework in which services were registered and sometimes traced by the look-up service. These events get notified whenever there was a change in state (change in the state of the distributed system). These invoke the ensemble of service to disseminate and gather the information about the configuration of the time-dependent state of the process, network, and the jobs running throughout the structure. This gathered information is transmitted for the analysis to advance level for corrective measures.

Shukla, S. and Badal, N.
Monitoring System for Cloud Services in Distributed Architecture.
DOI: 10.5220/0010563400003161
In *Proceedings of the 3rd International Conference on Advanced Computing and Software Engineering (ICACSE 2021)*, pages 115-118
ISBN: 978-989-758-544-9

The implementation prototype for the above architecture was on web-based technology which incorporates software components and devices into a single dynamic distributed system. Thus architecture formed has broad segments as follows.

The Data Collection Engine. One of the important tasks of the architecture was a collection of data independently and in parallel. Various modules were formed to collect different sets of information which were dynamically loaded and executed in atomic threads. In this manner, a pool of threads has been created and reused frequently.

Data Storage. Collected data values in form of normalized tables were stored in the relational database. As the collection of the data become older they were compressed and mean values were computed for a bigger duration.

Registration and Discovery. There has been an assignment of the attributes set for the group of services in the register of JINI Lookup Discovery Services. There will be replication from one to another when the information related to the group is found common in two Lookup Discovery Services. This will leads to the possibility of framing a reliable network for the registration of services. The registration was based on a lease mechanism that was responsible for verification services to be alive. Those services which fail to renew lease had been removed from the Lookup Discovery Services and a notification has been sent to clients & services which were subscribed for that events.

Predicates, Filters, and Alarm Agents. Resultant values of attributes description were similar to regular expressions-based predicates. Historical data was fetched with SQL queries on the request from the local databases. To serve each of the clients a dedicated thread has been created through creation request. These threads were responsible for the similarity test in the data flow as well as for the compressed serialized objects to the clients. Independent threads were fast and reliable if the communication errors were avoided.

Using the WSDL/SOAP predicates mechanism was also possible for the Monitoring data requests. WSDL has been used to describe the predicted classes for the client's objects to dynamically communicate.

Agent filters have been also used for information extraction. These modules are developed in the Java language for MonLISA services, data processing tasks, and returning periodically the processed information.

Whenever abnormal behaviors were detected, alarm agents were dynamically loaded to improve, manage the efficiency of the facilities for a Grid system.

Graphical Clients. The global graphical client has been developed for the discovery of active services from defined groups. These are application web start which can be started from any of the browsers.

A dedicated GUI has been developed containing the marshaled components for the service's attributes. Communicating back with each of the services to fetch the detailed information to plot has been created with values requested. Filters Agents, predicate mechanism provides the flexibility in the real-time and historical value in MonALISA. GUI automatically updates the values whenever it finds the matching values of the subscription that was collected. Services were started and unavailable this can be easily shown by the graphical clients by remote notification mechanism.

Administration Services. To optimize the workflow and manage the distributed facilities dedicated GUI-based dynamic configuration mechanism was created. This configuration mechanism will help in creating or reconfiguring clusters, network elements, and new nodes or modules as required. It also helps module suspension, stopping, and restarting, which plays an important application performance to be understood.

Global web start also helps in the start of the administrative GUI with authentication of the trusted users. This can be proceeded by providing a private key from the administrator into the GUI and the clients will have the rights to the services.

Automatic Update for Services. As the framework deployed several locations it requires efforts to update and maintain the application. This has been achieved by developing the mechanism to automatically update the monitoring service. Threads were there check periodically for updates if any in the distribution, a restart operation was initiated when any of the events were detected. In the sequence of this, all packages were downloaded automatically to run the application while checking necessary constraints. The last published version of the application securely runs after the updating services state.

2.2 The Lattice Monitoring Framework

S. Clayman et al, built a lattice framework to optimize the workflow of the cloud services (Clayman S. et.al, 2010). Managing cloud services required the collection of relevant data effectively, so as no footprint overload occurs. In huge distributed architecture large number of probes occur so there is required relevant data collection. Designing the framework we have to first understand the producers and consumers of the data that were in the workflow. These were the elements of the network which will generate the data values while communicating with each other. In various monitoring systems probes are responsible for the data source while in lattice framework there has been a data source that will control and interact encapsulating more than one probe. There was a requirement for a fully dynamic data source. As the data is produced and collected from the various probes in the system they have a data distribution mechanism for efficient transmission of the measurement over the network.

Utilization of Lattice within RESERVOIR. As a part of the RESERVOIR project (Galis et.al, (2009) they had built a system for cloud service management using the Lattice framework. From the various probed they have gathered measurement data which were attached with virtual machines. In the sequence monitoring virtual resources and monitoring physical resources, probes have been written.

Monitoring Physical Resources. Probes have been created for Memory, CPU usage, and network usage in the underlying infrastructure. The probes for the CPU usage firstly collect data from all the cores of the processor from the various server, then the probe for memory usage is initiated following the probe responsible for the network data, these all worked in a real-time manner. This sequence runs at an equal interval of time to manage the traffic.

Monitoring Virtual Resources. Probes have been created for the data collection at running virtual machines on a particular host. Hypervisor and probe interact with each other to collect data that are under the control of the hypervisor. On the virtual machine, these probes run regularly for the collection of the data.

Monitoring Service Applications. In each virtual machine, there must be a probe deployed for the service cloud on the application environment. These probes collected data and sent it to the service manager via infrastructure from a virtual machine. Sun Grid Engine application (Sun Microsystems, 2008) used the virtual job queue that was developed by these probes.

This application has the elasticity rule to measure the queue length, so that (a) Automatically a new virtual machine has been allocated when queue length was high or (b) Automatically shut down the machine when it is low. Sun Grid Engine application-optimized the running Service Manager by adapting to the queue length of virtual machines.

3 CONCLUSION

The MonaLISA architecture simplifies the administration of the complex system, construction, and operation by interacting with the services in a robust, dynamic manner. On the same pattern, the Lattice framework provides used as a platform for various monitoring systems. Anyone can write probes of specific purpose to collect data, and consumers can access it in any way necessary. For different applications Lattice had not provided any pre-defined consumers, data source, or probe. Rather, as per the requirement, each of them can be deployed.

The above mentioned both of the discussed monitoring systems well successful monitoring system known. These systems work in three stages, which include the creation of probes for services, collection of the data values from each of the probes then sequentially updating the administrative system of the application. This shows that in today's virtual world of cloud services deployment, monitoring optimization works together in parallel for the better performance of the distributed system.

REFERENCES

H. B. Newman, J. J. Bunn, I. C. Legrand, (2001), "A Distributed Agent-based Architecture for Dynamic Services" in CHEP, Beijing,

H. B. Newman, I. C. Legrand, P. Galvez, R. Voicu, C. Cirstoiu, (2003), "MonALISA : A Distributed Monitoring Service Architecture" at Computing in High Energy and Nuclear Physics, La Jolla, California, 24-28.

Sun Microsystems, (2008), "Guide to Sun Grid Engine 6.2 Installation and Configuration," white paper.

Galis, B. Rochwerger, E. Levy, D. Breitgand, et al. (2009), "The RESERVOIR model and architecture for open federated cloud computing," IBM Journal of Research and Development, vol. 53, no. 4.

S. Clayman, G. Toffetti, A. Galis, B Rochwergere, C. Chapman, L. Rodero-Merino, L M. Vaquero, K Nagin, (2010), "Monitoring Service Clouds in the Future Internet" in Towards the Future Internet G. Tselentis et al. (Eds) IOS Press.