DOI: 10.7763/IPEDR. 2012. V49. 14

Study of Realized Mehtod on a Java Web Server Monitoring System

Kun Liu[1], Hai-yan Zhao[1], Long-jiang Dong[2] and Li-juan Du[1]

[1]College of Oriental Application & Technology, Beijing Union University, Beijing, China

[2]Software Development Department, Adobe (Beijing) Corporation, Beijing, China

e-mail: wtliukun@buu.com.cn,

donglongjiang@yahoo.com.cn, zhaohaiyan@buu.edu.cn, yykjtlijuan@buu.edu.cn

Abstract—This paper introduces the JMX technology and MXBean platform firstly, then designs a monitoring architecture based on JMX technology and MXBean. Lastly the actual application of this architecture is described with Weblogic monitoring object as an example. The experiments prove that this monitoring system can monitor servers in real time and send out warning messages when servers are in an unstable state.

Keywords-Java Management Extentions; Monitoring System; Java Virtual Machine;Web Server

1. Introduction

With the development of Java Web and J2EE technology, more and more applications are developed by Java. The servers such as Jboss,Weblogic[1],Tomcat and etc need to be used to run the applications. These servers must work well for providing reliable and stable services when the systems under the condition of high load. But under special circumstances such as a sudden increase in traffic, malicious attacks by hackers and etc, these servers always collapse and services can not be achieved. In order to avoid this kind of circumstance happening, we design a Java Web server monitoring system. This monitoring system can monitor servers, judge whether servers are in an unstable state or not and inform system administrators to take appropriate measures.

2. Monitoring Architecture Design

2.1 Overview of Related Technologies

1) JVM

The Java Virtual Machine (JVM) is instrumented for monitoring and management, providing built-in ("out-of-the-box") management capabilities for for both remote and local access. The JVM include a platform MBean server (management agent) and platform MBeans that JMX management applications can use[2].

2)JMX

The JMX[3][4] technology provides the tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future[3]. Typical uses of the JMX technology include:

- Consulting and changing application configuration
- Accumulating statistics about application behavior and making them available
- Notifying of state changes and erroneous conditions

The JMX API includes remote access, so a remote management program can interact with a running application for these purposes.

3) MBean

MBeans are managed beans, Java objects that represent resource to be managed. An MBean has a management interface consisting of[2]:

- Named and typed attributes that can be read and written
- Named and typed operations that can be invoked

- Typed notifications that can be emitted by the MBean

MBeans can be standard or dynamic. Standard MBeans are Java objects that conform to design patterns derived from the JavaBeans component model. Dynamic MBeans define their management interface at runtime[2].

In JMX technology framework, monitoring resources are designed to managed Beans, also known as MBean objects. These Mbeans are registered into MBean server which is embedded in the Java Virtual Machine. Monitoring managed programs connect to MBean server using JMX technology and read datas from the server in real time.

4) MXBean

Starting with release 5.0, J2SE includes a built-in platform MBean server and provides a set of standard platform MBean. A platform MBean (also called an MXBean) is an MBean for monitoring and managing the Java Virtual Machine (JVM). Each MXBean encapsulates a part of JVM functionality such as the JVM's class loading system, JIT compilation system, garbage collector, and so on[2]. We can get the following informations from MXBean.

a) Information of garbage collection statistics, include: number of garbage collectors in Java Virtual Machine, times of garbage collection made by each garbage collector, time-consuming and etc.

b) Information of memory pool, include: number of memory pools in Java Virtual Machine, minimum capacity, maximum capacity and used capacity of each memory pool and etc.

c) Information of threads, include: number of threads in Java Virtual Machine, number of deadlock threads, details of each deadlock thread and etc.

d) Runtime information, include: start time and running time of Java Virtual Machine and etc.

e) Information of compilation system, include: cumulative time-consuming for compilation in Java virtual machine and etc.

2.2 Monitoring Architecture

Java application server consists of one or more Java application programs. Each Java application runs on a separate Java Virtual Machine. Therefore, monitoring the application server is the same to monitoring the Java virtual machine.

Monitoring architecture is composed of Collector and Analyzer.

1) Collector

Collector is responsible for obtaining related informations about Java Virtual Machine from standard MXBeans and monitoring other customize informations. These informations include the process's memory capacity of Java Virtual Machine, CPU's utilization ratio, number of Java Virtual Machine's object instances, memory size used by each object, number of J2EE applications requested by users and etc. Collector packages these informations into MBeans.

In order to obtain customize MBeans from platform MBean server, we embed an MBean loader using JVMTI[5] technology when Java Virtual Machine starts up , load these customize MBeans into system and then register on platform MBean server using the MBean loader.

2) Analyzer

Analyzer connects to platform MBean server by Collector using JMX technology and monitors running state of Java Virtual Machine in real time. We analyze the datas obtained from Collector and inform system administrators in time if Java Virtual Machine in an unstable state. When the data value exceeds the specified threshold range, we say Java Virtual Machine is in an unstable state.

Monitoring architecture figure is shown in Figure 1.

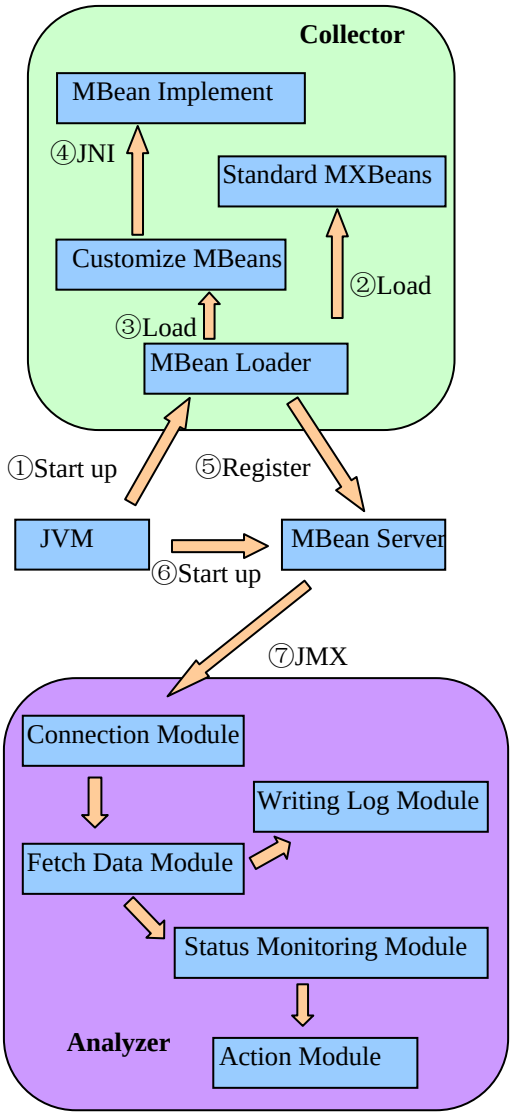

Figure 1. Monitoring Architecture

3. Implementation of Collector

We introduce the practical application of monitoring system on the Linux machine equipped with Weblogic service.

3.1 Workflow of Collector

- When Java Virtual Machine(JVM) starts up, it sets "-Xjavaagent" parameter option and starts up MBean loader(The step is marked ① in Figure 1). JVM also starts up MBean Server at meantime(The step is marked ⑥ in Figure 1).
- The loader loads standard MXBeans and customize MBeans(These steps are marked ②③ in Figure 1), and regitsters them into MBean server(The step is marked ⑤ in Figure 1).
- MBean server starts up using JMXConnectorServer(The step is marked ④ in Figure 1).

3.2 Loading related standard MXBean

We get the concerned MXBeans from java.lang.management.ManagementFactory, include:

f) *MemoryPoolMXBean:* Information of memory pool.

g) *MemoryMXBean:* Information of each memory pool's memory usage condition.

h) *GarbageCollectorMXBean:* Information of garbage collector.

i) *RuntimeMXBean:* Running information of Java Virtual Machine.

j) *ThreadMXBean:* Count of threads and information of deadlock condition.

k) CompilationMXBean: Information of compilation system.

3.3 Implementing and loading customize MBeans

We design customize MBean interface and complete the specific implementation of the following information using native method.

l) Information of virtual memory capacity: The VmSize field in /proc/<pid>/status system file.

m) Information of CPU utilization ratio: The related time field in /proc/<pid>/stat system file.

n) Information about the virtual machine's object instance: The usage state informations of all object instances are obtained using JVMTI technology.

o) Count of pending users' requests: WorkManagerRuntimeMBean.PendingRequests[8] of Weblogic.

p) Count of rejected users' requests: MaxThreadsConstraintRuntimeMBean.DeferredRequests[8] of Weblogic.

q) Count of users' requests per second: ThreadPoolRuntimeMBean.Throughput[8] of Weblogic.

4. Implementation of Analyzer

Analyzer consists of Connection Module, Fetch Data Module, Status Monitoring Module, Writing Log Module and Action Module. Connection Module and Fetch Data Module communicate with Collector and receive data from Collector. Other modules analyze the data and take appropriate actions. The detailed steps are discussed in title "F" of this section.

4.1 Connection Module

Tasks of Connection Module include:

- Analyzer connects to MBean server on Collector using JMXConnector.
- MbeanServerConnection object is returned to Analyzer using JMXConnector.

4.2 Fetch Data Module

Tasks of Fetch Data Module include:

- Obtain standard MXBeans and customize MBeans at regular intervals through MbeanServerConnection object.
- Obtain real-time data of Java Virtual Machine from MBeans.

4.3 Status Monitoring Module

Tasks of Status Monitoring Module are analyzing the data from Fetch Data Module. If the data value is over threshold repeatedly, it means that Java Virtual Machine is in an unstable state. The threshold can be set as follows:

- Excessive memory usage(More than 80%)
- Excessive number of garbage collection(More than 20 times / min)
- Excessive virtual memory usage(More than 80%)
- Excessive number of threads(More than 100)
- Existing deadlock threads
- Excessive number of pending users' requests(More than 20)
- Excessive number of rejected users' requests(More than 10)
- Count of users' requests per second continued to increase(Continuous increase 100%)

4.4 Writing Log Module

Tasks of this module are saving the datas from Fetch Data Module to files with a certain format.

4.5 Action Module

Tasks of Action Module include:

- Send E-mail and SMS to system administrators if Java Virtual Machine is in an unstable state.
- Collect all the running information under the current state, so the system administrators can track the location of fault cause.

4.6 Workflow of Analyzer

- Connection Module connects to MBean Server using JMX technology(The step is marked ⑦ in Figure 1).
- Obtain monitoring object's state informations in real time through Fetch Data Module.
- Save and analyze the data obtained.
- Judge monitoring object's data according to the given threshold in Status Monitoring Module. If reaching the alert status, Analyzer implements the action set in Action Module and informs the administrators in time.

5. Exepriments

We use this monitoring system on Linux machine equipped with Weblogic service and set the threshold counter to 5. Data acquisition results are shown in Figure 2 and Figure 3.

As shown in Figure 2, we do experiment ten times when count of threads is excessive (count of collection points is 3 to 7). The count of threads is over than 100 at each experiment. The monitoring system identified threads' number is too much and sent warning message to system administrators.

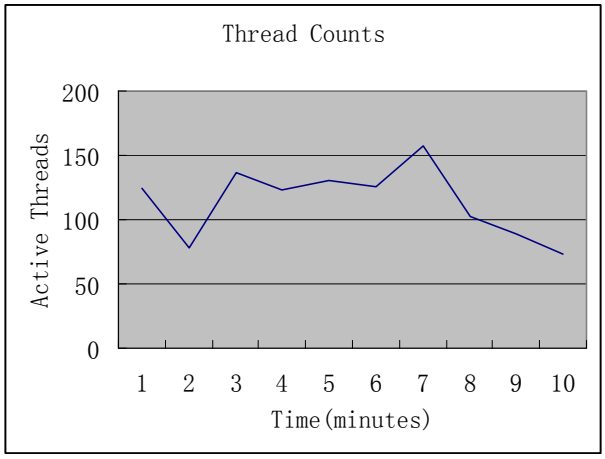

Figure 2. Excessive number of threads

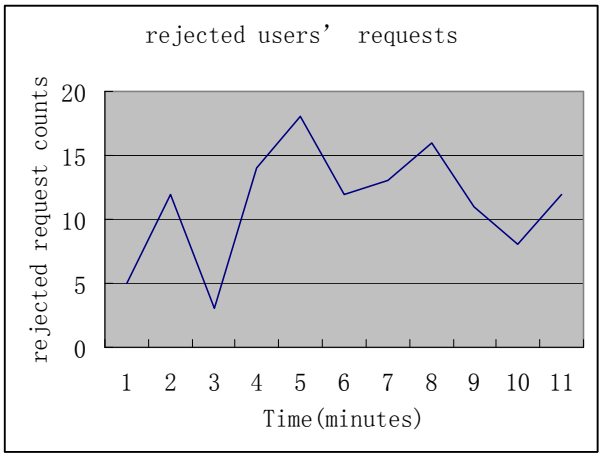

Figure 3. Excessive number of rejected users' requests

As shown in Figure 3, we do experiments eleven times under the circumstances of excessive count of rejected users' requests (count of collection points is 3 to 7). Count of rejected users' requests is over than 10. The monitoring system identified rejected load of Weblogic is too high and sent warning messages to system administrators.

6. Conclusions

We find if the object monitored by the Java Virtual Machine in the state of high load and malicious attacks by hackers and etc, the monitoring system can make the right judgments and let administrators receive alert messages in time to take appropriate measures.

7. Acknowledgment

This work is supported by a grant from the Research Program of Beijing Union University(No.ZK2009606).

8. References

[1] Qiu zhong-fan, Chen chun-ying, Chen ling-feng. BEA WEBLOGIC SERVER management Guide[M].China machine press,2003.

[2] Overview of Monitoring and Management. [EB/OL]http://download.oracle.com/javase/1.5.0/docs/guide/management/overview.html.

[3] Java Management Extensions (JMX) Technology[EB/OL]. http://www.oracle.com/technetwork/java/javase/tech/index-jsp-135104.html.

[4] Hanson, JEFF. PRO JMX [M]. Springer-Verlag Ner York. 2005.

[5] JVMTM Tool Interface (JVM TI) [EB/OL]. http://download.oracle.com/javase/1.5.0/docs/guide/jvmti/index.html.

[6] Bai jiang-tao, Zhong yong, et al. Real-time garbage collection algorithm in Java virtual machine [J]. Application Research of Computers. 2010,27(9):3431-3433.

[7] Zou qiong, Wu ming, et al. An Instrument-Analysis Framework for Adaptive Prefetch Optimization in JVM [J]. Journal of Software.2008,19(7):1581-1589.

[8] The WebLogic Server® MBean Reference[EB/OL]. http://download.oracle.com/docs/cd/E13222_01/wls/docs92/index.html.

[9] Liu dan, Meng ling-kui. Research on Hardware Implementation of Java Virtual Machine[J]. Computer Engineering.2007,33(6):233-235.