Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp706-710)

A Scalability Model for Managing Distributed-organized Internet Services

TSUN-YU HSIAO, KO-HSU SU, SHYAN-MING YUAN
Department of Computer Science,
National Chiao-Tung University.
No. 1001, Ta Hsueh Road, Hsinchu, Taiwan 300,
TAIWAN

Abstract: More and more administrators desire to monitor and control the exactly states of service components, especially enterprise platform. To meet the requirement of managing distributed internet services nowadays, in this paper, we introduce a more flexible and scalable framework to help manager to manage distributed internet services. It is an n-tiered architecture. We implement the idea on a MMOG platform to show its power of flexibility & scalability.

Key-Words: Distributed System, Management, Internet Service, Scalability, MMOG.

1 Introduction

The management issues of distributed system are getting important while Internet services are growing. While traditional management concerns on the exterior status of managed devices, such as, CPU, memory, I/O loading, and network status, more and more administrators desire to manage and control the internal status of managed devices precisely. Moreover, many Internet services are organized and collaborated by many individual computing nodes. Monitoring single node can not satisfy administrators. In order to provide a good Internet service, the existed management and control technology need to be improved.

More and more administrators desire to monitor and control the exactly states of service components, especially enterprise platform, for example, services based on J2EE technology. Main characteristic of these internet services are that it is grouped by services that has similar functionalities. Monitoring the load of CPU or usage of memory usually cannot help the administrator to realize what problem is encountered when services is unavailable. Therefore, several management technologies that are trying to make object manageable were introduced, such as, SNMP[1], MUWS[2], CIM[3]…etc. To coexist with legacy system or application, most of the management technologies are focusing on protocol integration. For example, most of network components were management by SNMP protocol adapters. MUWS is trying to manage existing services by using Web Services architecture and related technologies. Common Information Model (CIM) is focus on an object-oriented information model standardized within the DMTF for the purposes of providing a conceptual framework within which any management data may be modeled. However, most of these technologies are short on managing the information from managed objects with scalable and easy-to-use manners.

The purpose of this paper is to present a more flexible and scalable framework that will help administrators manage distributed internet services and so meet the need for managing objects in the current distributed computing environment. It has n-tiered architecture with three major layers: manager, delegation and agent. The manager layer defines the remote management APIs, which is to say that it uses a remote management application to retrieve information from managed objects by invoking these APIs. The delegation layer provides for managing all the distributed nodes in the computing environment, so as to create registers, and startups corresponding to the management beans. The agent layer at each service node packages the resource into the managed object and registers the local server information to the delegation server. In the process, we created the framework on a MMOG platform termed DoIT [4] (which is a platform of distributed-organized internet services).

The remainder of the paper is organized as follows: Section 2 describes related works; Section 3 illustrates our model and the architecture for a distributed internet service; Section 4 explains how the model is implemented on a MMOG platform; Sections 5 and 6 present what has been learned and reports conclusions, respectively.

2 Related Works

In recent year, many useful technologies were been purposed for managing network components or services. One catalogue of them is to integrate a variety of existing network management technologies, for example, MUWS, CIM, and SNMP. These technologies provide different aspect of integration capabilities from protocols to presentations. MUWS technology use web service related technology to manage several of distributed information technology resources. CIM provides a common definition of management information for systems, networks, applications and services, and enable vendors to exchange semantically rich management information between systems throughout the network. The Simple Network Management Protocol (SNMP) is an application layer protocol that facilitates the exchange of management information between network devices. The other catalogue of technology is to build a manageable framework that is to enable a technology to create a manageable environment, for example, JMX. Since web services related and Java technology are popular distributed technologies in recent year, we give a short description and analysis of advantages and disadvantages of management standard of each catalogue by introducing Management using Web Service (MUWS) and Java Management eXtensions (JMX)[5] in this section.

Management Using Web Services (MUWS) enables management of distributed information technology (IT) resources using Web services. By leveraging Web service technology, MUWS enables easier and formal representation management of IT resources. This is accomplished by providing a flexible framework for manageability interfaces that leverage key features of Web services and its protocols. Figure 1 describes the main concept of MUWS: manager implement a Web Service endpoint adapter for a resource. Currently, MUWS spec focuses upon how access is provided to manageable resources. The central element and focal point of the WSDM architecture is the manageable resource. The message patterns encapsulate access to resources into manageable resources instead of exposing message patterns to indirectly access the resource through agents, proxies, observers, etc. MUWS benefit greatly from web services technology, for example, isolation from implementation, simple and easy-to-understand representation. However, some drawbacks that are related to web services can be seen obviously, such as, high level representation and access end point for legacy components caused performance down. For some specified application domain that need for performance, MUWS may not a good solution.

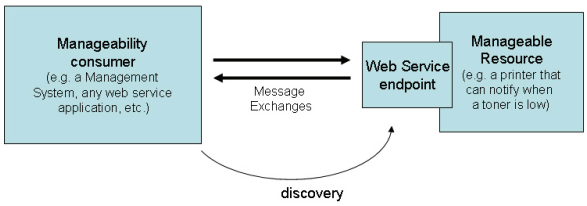

Fig 1. Concept of Management Using Web Services.

Java Management Extensions (JMX) technology provides Java technology a framework for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. The JMX management standard defines an complete management architecture for Java applications, including an API to instrument applications with manageability.

The Java Management Extensions (JMX) technology is an open technology for management and monitoring that can be deployed wherever management and monitoring are needed. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions and plugging into those of the future.

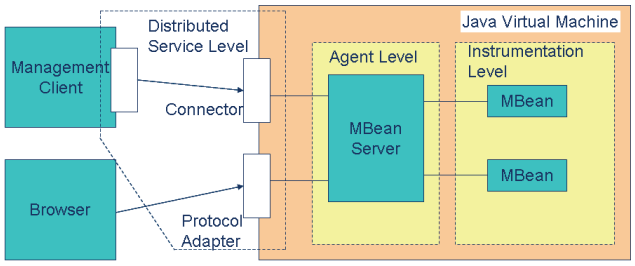

Fig 2. JMX Management Architecture

Figure 2 is the JMX management architecture; it is implemented by three levels architecture. First, in order to manage the applications, programmer create an MBean which is the most basic management component of JMX at the instrumentation level, programmer need to declare what management functions it has. And then programmer registers the MBean to the MBean server and it will be add to the MBean server's repository at the agent level (in the same Java Virtual Machine). The MBean server can management the applications by invoke the MBeans, and it provide a unify management interface which can be invoke by remote managers. At the distributed service level, the MBean server doesn't provide the remote service connection functions itself, so that programmer need to create a remote connector or

connection protocol MBeans at the server. Therefore, the managers can remote setup or manage the components on the MBean server through appropriate connectors.

In General, The JMX management standard enable Java applications to be managed without heavy investment, including an API to instrument applications with manageability. JMX also enable Java to integrate with existing management solution by using protocol adapters. WSDM is a set of management specifications from the WSDM Technical Committee in OASIS. WSDM provides more structure and semantics for a manageability interface than JMX.

In the following section, we propose a development environment and an architecture for distributed object management with enhanced scalability that satisfies current management requirements. This development kit, an extension of the idea of JMX, helps programmers manage distributed internet services rapidly and easily.

3 Our model and architecture

While most mechanisms or architecture focus on integrating certain network devices/services by a unified protocol or presentation (with such as XML, SNMP, GMPLS technologies [6][7]), it is very important to present a more scalable and flexible architecture/framework. For flexibility and manageability, an n-tiered level architecture that is similar to JMX architecture is presented. The device has three basic layers: manager, delegation, and agent, the relationship between which is illustrated in Figure 3. At the manager layer, a number of remote management APIs is defined. In this way, a remote management application is employed to retrieve/manipulate the information of managed objects by calling the APIs. The delegation layer provides a management component to help the manager deal with all the distributed nodes, such as create, register, and startup, corresponding to the management beans. The ServerFactory records the reference of the different service nodes. The delegation layer also provides a remote connection port that allows remote control. At the agent level, an individual service node packages the resource into the manageable object, and registers the local server information to the delegation server.

Our development environment is focus on managing distributed internet services, such as J2EE, and MMOG platform. Main characteristics of them are each service has the same functionalities, and are organized by many computers. For example, in a MMOG service, each node handle different geographical coordinates of the virtual world, but all of the nodes have the same individual object to be managed, such as, player characters, auction lists. Therefore, to manage the identical object state of each node with the same service component, we use a code generation model that provides a rapid development kit for the purpose.

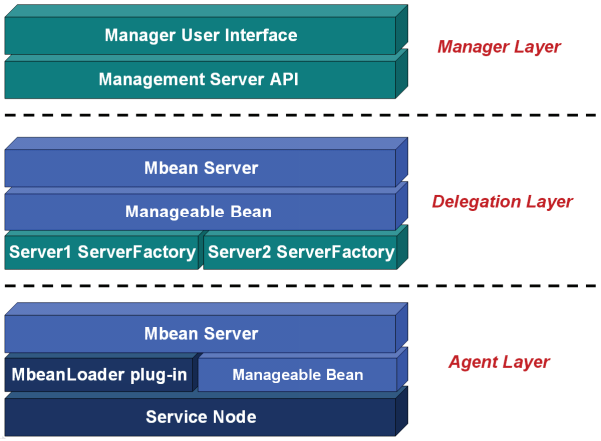

Fig 3. n-tiers management architecture

Figure 4 illustrates the process of code generation. The code generation model can be divided into 4 steps. First, programmers write the Manageable Bean interface in a file with XML [8] format. Second, programmers use our code-generation kit to generate the skeleton of these management beans. Third, programmers implement the interface in MBeancode. Finally, programmers deploy the management bean on the nodes that are need to be managed, and register to delegation server.

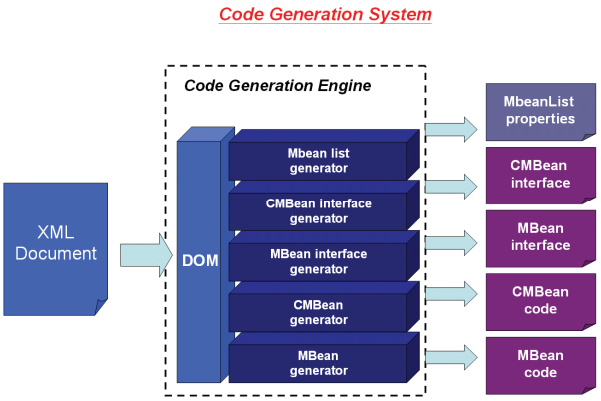

Fig 4. The flow of Code Generation System

4 Case Study: Our detailed Implementation on DoIT MMOG platform

Here, the implementation of a real management system based on the framework of the MMOG Platform: DoIT platform[4] is described. The MMOG environment is combined by many servers. Each service node handles different requests from different clients. To build a management environment on DoIT Platform, we introduced the management system architecture based on the architecture in section 3. Due to DoIT platform is built by pure Java language, we choice to realize our idea based on extending from JMX framework. The detail relationship of layers and components are shown in the Figure 5. It was divided into three layers: agent, delegation, and manager layers. The entire management system work flow and design detail will be discussed in the follow subsection.

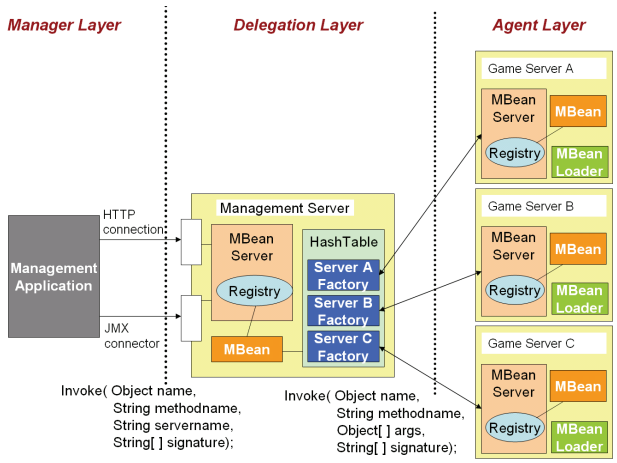

Fig 5. A Management System on DoIT platform

```
public interface ServerManagerMBean {
    public Integer getRegionCount(); //get the total number of Regions in single server
    public void setRegionNumber(Integer regionnumber); // set region number
    public String getRegionName(); // get name of region set by manager.
    public void setdebug(Integer debug);
    public void Debug();
    public Integer getAVAcount(); // get current number of Avatar in this region
    public Integer getNPCcount(); // get current number of NPC in this region.
    public Integer getRegionID(); // get current ID of Region
    public void resetRegion(); // reset Region info
    public void register() // register to Management Server
}
```

Fig 6. Interface of ServerManagerMBean

4.1 Agent Layer

At the agent layer, the architecture and work flow are show in the Figure 7. We designed a management plug-in program named MbeanLoader. This program is designed by following the DoIT platform plug-in module and will be executed when the server is startup. After game server startup, MBeanLoader will create each Mbean and register to MBserver on the same node and delegation management server. Figure 6 is an example of a common management API of ServerManagerMbean.

4.2 Delegation Layer

The process flow of delegation layer is shown in the Figure 7. In this layer, ManagementServer will startup an Mbean server and creates a ServersManagementMbean and also registers it to the Mbean server. After the management server have start, it will waiting for the remote connection. It provides two kind of the service, one is for the game server and another is for the manager.

The game server can connect to the management server and register a CustomMBean. When this request is received, the manager creates a ServerFactory object according to the register information. When the management server receives the manager's request, it can ascertain the responsible server factory, which can then be activated to execute the manager's request. Depending on the individual server name, the individual ServerFactory object reference can be ascertained. The individual ServerFactory object can select the method for connecting to the game server and invoke it in the ServerManagerMBean. Figure 10 shows the overview of the ServerManagementFactory class. It has a one-to-one relationship to the ServerManager Mbean. The ServerManagementFactory individually shares same-named methods with those of the ServerManager Mbean. Because the ServerManagementFactory plays an intermediary role, it is responsible for connecting to the game server and for invoking the Mbean's method. It therefore shares the same methods with the ServerManager Mbean.

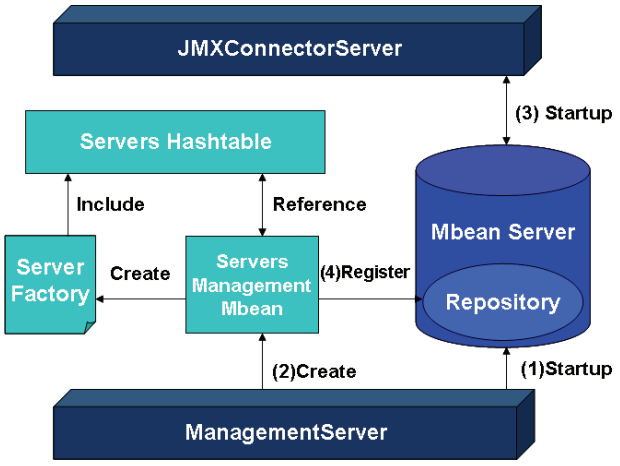

Fig 7. Central Management Server Work Flow

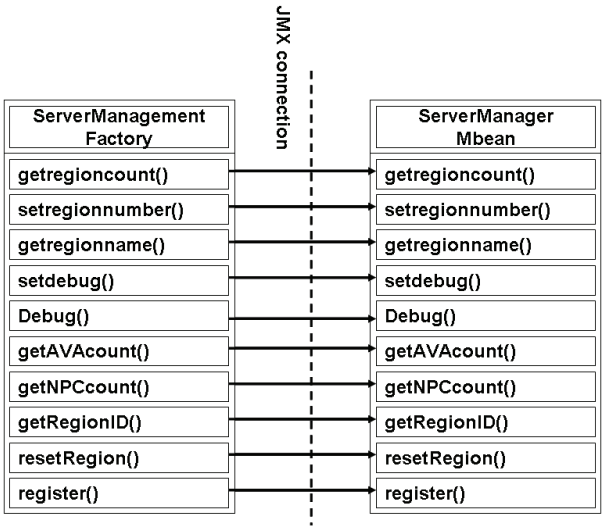

Fig 8. Overview of the ServerManagementFactory Class

4.3 Manger Layer

At manager layer, the managers can do management work by writing their own management application. By using JMX techonologies, Manager can connect to Management Server throw JMXConnector or different protocol adapter (such as HTTP Connector). However, in this layer, we still define a set of the servers' management API for programmer to connect to management server. ServersManagementFactory class provides some management methods which can be used by the management application. For example, manager can initialize a ServersManagementFactory object by a given central management server's IP address and the server port number of management server. And then we can use this object's method to connect to the management server, retrieve MBean lists, or get specified game server's state.

5 Discussion and Conclusion

After we implemented the development and management system on the DOIT MMOG platform, we have got some results in the below:

(1)To use the XML to be the MMOG content description language will have some advantage based on the XML's characteristic. First, the game content designer will have a simply way to learn the description syntax, and they can edit a MMOG description document by any text edit software. Second, because the XML uses the Unicode encoding, so the content designer can use any language code to edit the document. Third, a lot of the data in the MMOG are formed by structured data, so they are suitable be described by XML. After edited the description document, we can use a XML schema file to verify if it is valid, and we can also use the existing XML parser to analyze the document content and then process these data.

(2)In the development of the MMOG management system, we extend JMX to solve the network communication problems and use the Mbean server to implement a n-tier management server architecture. It will reduce the work load of the MMOG management system development. And due to all of the resource which we want to management will be packaged into the Mbean in our management system. If there have some new resource also must be management, then we just only add some methods into the Mbean and make an Mbean re-registry action, and then we can keep on the management work and also add these new management resource. And the manager can connect to the management server by http connection or other JMX support connection protocols. According to the method, it will make the management work with more elasticity and more convenient.

References:

[1] RFC 1157 - Simple Network Management Protocol (SNMP)

[2] Oasis, Management Using Web Service (MUWS), http://www.oasis-open.org/specs/index.php#wsdm-muwsv1.0

[3] Distributed Management Task Force, "Common Information Model", http://www.dmtf.org/

[4] Tsun-Yu Hsiao, Shyan-Ming Yuan, "Practical Middleware for Massively Multiplayer Online Games," *IEEE Internet Computing*, Vol. 09, No. 5, 2005, pp. 47-54.

[5] Java Management Extensions (JMX), http://java.sun.com/products/JavaManagement/

[6] Klie. T, Straub. F, "Integrating SNMP agents with XML-based management systems", *IEEE Communications*, Vol. 42, No. 7, 2004, pp. 76 – 83.

[7] Munoz. R, Pinart. C, Martinez. R, Sorribes. J, Junyent. G, Maier. M, Amrani. A, "Integrating GMPLS, XML, and SNMP in transparent DWDM networks", *IEEE Communications*, Vol. 43, No.8, 2005, pp: s40 - s48.

[8] W3, Extensible Markup Language (XML), http://www.w3.org/XML/