Service-Oriented Message Queue

Min Wang
Nanjing research institute of electronics engineering
Nanjing, China
e-mail: minlet@qq.com

Hui Xu
Nanjing research institute of electronics engineering
Nanjing, China
e-mail: fairydm@qq.com

***Abstract*—Message Oriented Middleware (MOM) implement platform-independent data exchange with efficient and reliable messaging mechanism, what is the key technology of distributed systems integration. The traditional messaging middleware based on a specific platform or development tools, it has been unable to achieve in a loosely coupled manner, lack of adequate scalability and flexibility, it is difficult to meet the demand of system integration variability. This paper designs and implements a service-oriented messaging middleware Service-Oriented Message Queue (SOMQ). Characteristics of the model is the messaging middleware system according to service-oriented architecture is divided into different functional services that are independent of each other, and can according to the need to minimize the cost function combined. And SOMQ architecture is designed based the model, and the design and implementation of the system of internal modules are given.**

Keywords-message oriented middleware; service oriented architecture; queue

I. INTRODUCTION

MOM and Service-Oriented Architecture (SOA) is a hot field of system integration of current research, and is widely applied to engineering practice, but combining the two existing projects in practice are rare. This paper presents messaging middleware service-oriented model according project application, it can be said is an innovation in the field of application.

The rest of the paper is organized as follows. Section II compares MOM and SOA, Section III describes the architecture and design of this solution, Section 4.0 describes the experimental tools and experimental comparison data, Section V got a little conclusion.

II. COMPARISON OF MOM AND WEB SERVICES

MOM provides unified, robust communications platform for application development as a good quality underlying communication framework. However, the traditional MOM architecture based on remote procedure call, and never implemented in a loosely coupled, scalable manner.

The most mainstream way of SOA is Web Services, as a prominent feature of the Web Services, distributed software components have good interoperability. Hypertext transfer protocol (HTTP) and Simple Object Access Protocol (SOAP) protocol provides interactive transmission and message, Web Services Description Language (WSDL) provide interoperability between distributed applications by describing deployed Web services. However, just based on these technologies, Web services does not guarantee interoperability functionality and quality. Web services exist some weaknesses: first, HTTP protocol is based on synchronous request/response ways, communication parties are tightly coupled each other. Synchronization leads to decreased system efficiency, waste of network resources, the client has to wait before returning the result. Secondly, HTTP protocol is stateless. Stateless makes it impossible to guarantee reliable transmission of messages, when an error occurs, you cannot return to the previous state of affairs, and cannot guarantee message transmission characteristics. Third, HTTP is a "best effort transfer", is not reliable, require application layer support. The messaging capabilities of Web Services are not as good as MOM.

III. FRAMEWORK AND DESIGN

A. Composition

As a message handling system, MOM provides asynchronous communication, reliable transmission and message persistence ability to improve interact of the system, but the realization of the framework is not flexible, reusable system level is low, Web services technology is based on service as the basic element, provide a loosely coupled solution. We can use a service-oriented architecture ideas design MOM, it will combine the two to make up for deficiencies in the traditional messaging middleware architecture and Web support. This paper presents a model named Service-Oriented Message Queue Model (SOMQM), as shown in Figure 1

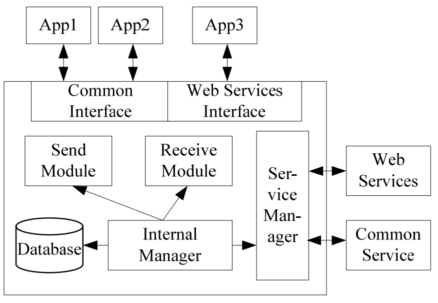

Figure 1. Service-Oriented Message Queue Model

SOMQM shown in Figure 1 shows that SOMQM conceptual model based services. Model is divided into three layers: the interface layer, an intermediate layer and service layer. The traditional messaging middleware only has the middle layer and the interface layer, there is no concept of the service layer.

Service layer provides for the intermediate layer specific function services, such as the upper layer protocol support services, message format transform services, and encryption and decryption services, what is the service infrastructure of the system. Services can include Web services and common services, Web services is specification; common services are the foundation services of the system development. Service Manager is responsible for unified management and scheduling services, add new services and delete services all in the service management module, the middle layer call the basic services without interact with services directly, service management unified call interface. The intermediate layer is the core layer of the system, use the services provided by the service layer to complete receive, manage and send messages, asynchronous transfer of messages and persistence mechanisms. Interface layer is applied to the interface to an external call, the system model provides two interfaces, a traditional client API interface, the other is a Web services interface, the system itself can be used as Web Services for the user to invoke.

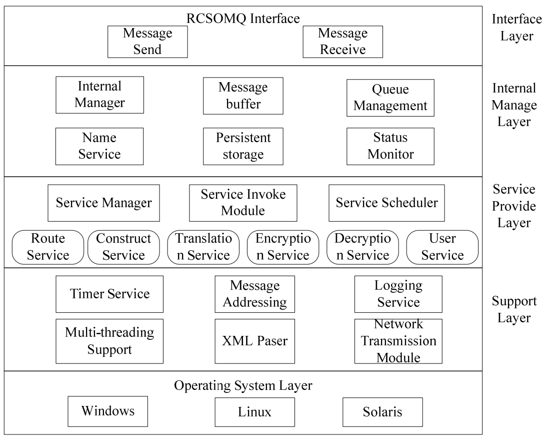

Figure 2. SOMQ hierarchical architecture

SOMQ hierarchical design based SOMQM, using the SOA style, following explains how the architecture is the embodiment of SOA thinking: the system architecture is based on service, the system is built to perform business functions in a series of reusable services basis. The system packaged message processing functions as a separate service, rather than simply function modular, and it emphasizes reusable and unit call interface of these modules, a combination of existing services to achieve greater service. Figure 2 rounded rectangle box in the service layer is the services that support system function.

SOMQ including the interface layer, internal management layer, service delivery layer and the underlying support layer, this article focuses on several layers of components. The following description is given of each component.

Underlying support layer includes the following modules: timer service, message addressing, logging service, multi-threading support, XML parsing, and network transmission module.

Timer Service: This module mainly support SOMQ time services, the life-cycle management and control messages.

Message Addressing: This module is responsible for determining the message destination address, the system uses the service registry module provides functions complete message addressing.

Logging Services: logging services for a variety of information recording SOMQ the details including the message, the message received, message send, service calls, message queue status, and debug information and error information. It supports different log levels and different log output format and the information to be recorded can be configured.

Multi-threading Support: Provides cross-platform thread support, including create and destroy thread, mutex for inter-thread synchronization, a variety of locks and condition variables. In this system, the processing of messages is very frequently, this paper manage threads thought the thread pool and reduce thread creation and destruction time and improve the processing efficiency of the system.

XML parsing: Provides parsing XML documents. Message format is based on XML files, the message queue, service management such as registration and unregistration of service are carried out by XML files.

Network transmission module: specific network send and receive functions. The network communication system integrated into a distributed collaborative development environment makes it possible to communicate with each other through the network, making each design tool can collaborate with each other through the network between the design tools on different machines .

Service provider layer including service manager, service calls and service scheduler, as well as the various service to support upper level functions.

Service Manager: the service management architecture for each of the service layer. Be achieved using an XML configuration file service registration, service cancellation and service search and other functions.

Service Invoke modules: each service invocation is not the same, the module encapsulates the invocation of various services such as Web Services and some custom services, providing an abstract interface to mask the different ways to call the service.

Service Scheduler: for scheduling and combination of services. By configuration an XML file to achieve service chain called, link a number of service to assemble. Support restructuring and customized services, you can modify the configuration file to achieve service assemble.

Internal management includes the following main modules: internal manager, message buffer, queue management, name service, persistent storage and status monitoring.

Internal Manager: Responsible for the organization and function of the layer and the service layer calls, and control of the entire messaging middleware, and maintain the normal operation of messaging middleware.

Message Buffer: this paper design a message buffer pool used to implement the message buffer. System design tools for system integration platform is a distributed collaborative environment, there are plenty of news during the run through SOMQ to interact with other modules or systems. In order to efficiently process these messages, and control SOMQ flow, it need message buffering mechanism to prevent the influx of large numbers of messages what causing the system takes a lot of resource, inefficient or even crash.

Queue Management: message channel mentioned is a message queue is SOMQ basic data structures, to send messages from one module to another module. Queue management is an important module of the system, is responsible for creating and deleting queues and control the behavior of the queue. There are several types of internal system queues: Point to point queue, publish / subscribe queue, response queues, queues with different priorities, error message queue. Message queue can have a name, it can be no name, SOMQ users can use to specify the name of the message queue.

Name Service: Name Service clients can access objects through SOMQ with the corresponding name what implemented in the system is a message channel, so that the client can specify a dedicated message channel to serve. Users can find and use object according to the name, without having to know the location of the object.

Persistent storage: In order to achieve reliable transmission, avoid errors in message handling process, the message needs to be persistent storage. At the same time, taking into account the distributed interactive features, you need to log message, making the process terminated unexpectedly resume execution at some future time. In these cases, the need for persistent storage capability. The system uses the database for persistence storage.

Status Monitoring: real-time monitoring system using state resources, monitoring computer CPU and memory usage, messaging middleware decide whether to process incoming messages.

Interface(API) layer is the interface SOMQ for users, including message send API and messages receive API according by function, including common client interface and Web service interface according by form.

B. Interaction Architecture

Figure 3 is an architecture diagram of the SOMQ interaction perspective.

What in the dashed box is services support system functions, such as construction services, translation services, encryption and decryption services, addressing services, and user-defined services.

First, the external application using the client API to send a message to SOMQ, message receiving module listens to reach the message, and construct message object in the system's internal format using the message construction services, and then sent the message to the message buffer pool. In the message buffer process, you can use the service within the dashed box for further processing of the message, use the message translation service to convert messages into another format, use encryption and decryption services to implement message encryption and decryption.

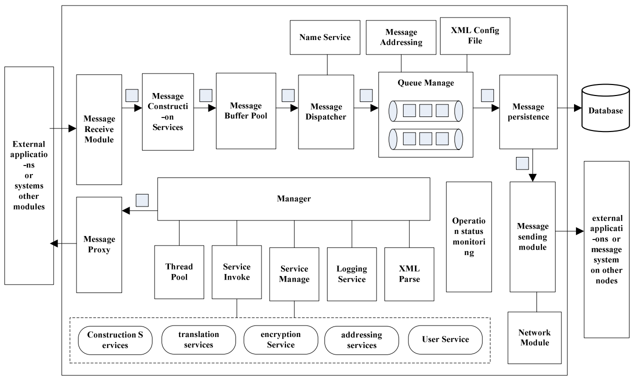

Figure 3. architecture diagram of the SOMQ interaction perspective

The message was sent to the message dispatcher after buffered, message dispatcher apply for new thread from a thread pool, distribute messages to a different message queue concurrently, there are several ways: If the user specifies the name of the message queue , the message dispatcher using the name service , then the message is sent to the message channel corresponding to the name; If you do not specify a name, the message sent by priority message priority messages to the corresponding channel ; If you do not set priorities , then distributed to the default priority channel. Queue Manager is responsible for the message channel (queue) management: design message channel pools, dynamically create and delete the message channel. When deploying the system, the user can configure the system 's initial queue.

Message channel according to function provided by the routing service, query destination address of message, the message is sent to the message send module. Before send out a message, the message needs to be persistence, record detailed information messages to the database.

Message send module and receive module implement the message send and using net module interface, interact with both module is designed port (or destination). Receive module can receive requests from other applications, it can also receive messages from SOMQ system on other computers node to distributed processing nodes; send module can send messages to other applications, it can also be sent to other computers SOMQ system on the node.

Message Agent for request / response mode, as a reply message proxy, the response message back to the requesting client.

IV. EXPERIMENT

Figure 4 shows the performance comparison of traditional messaging middleware and SOMQ. Experiments show that when the clients access to the message system is less at the same time, fewer concurrent processing threads in the system, the traditional messaging middleware performance better than SOMQ; And with the increasing number of client access, the more concurrent processing threads to increases a certain number, SOMQ transport delay is less than traditional messaging middleware, and the increase rate of delay is slower than traditional messaging middleware, showing better performance.

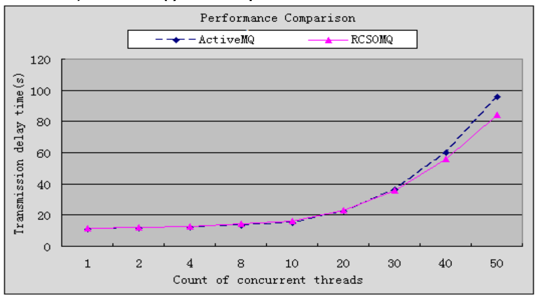

Figure 4. Performance Comparison of traditional messaging middleware and SOMQ

Experiments show that, SOMQ are reliable in dealing with large-scale message transmission, the transmission delay is acceptable, and compared to traditional messaging system has certain performance advantages.

REFERENCES

[1] Wilhelm Hasselbring. Information System Integration. Communications of the ACM. 2000, 43(6). Pages 33-38.

[2] KOMODA N. Service oriented architect ure (SOA) in industrial systems [C] // Proceedings of IEEE International Conference on Industrial Informatics. Washington, D. C. USA: IEEE. 2006. Pages 1-5.

[3] Sushant Goel, Hema Sharda, Dabid Taniar. Asychronous Messageing Using Message-Oriented middleware. J.Liuet a1. (Eds): IDEAL 2003, LNCS 2690, PP. 1118-1122. 2004. Springer-Verlag Berlin Heidelberg 2004.