Enterprise-oriented Communication among Multiple ESBs based on WS-Notification and Cloud Queue Model

Gu Pingli, Shang Yanlei, Chen Junliang, Deng Miaoting, Lin Bojia

Key laboratory of Networking and Switching Technology
Beijing University of Posts & Telecommunications, Beijing, China
{gplqy98, linbojia }@gmail.com; {chjl, shangyl }@bupt.edu.cn; ilua2006@126.com
doi: 10.4156/ijact.vol3.issue7.30

Abstract

With the rapid development of Service-Oriented Architecture (SOA), more and more enterprise applications are built on the Enterprise Services Bus (ESB). An Enterprise Service Bus is an open standard, message-based, distributed integration infrastructure. It is called as distributed services, integrated interaction. ESB is responsible for routing of message of all services or components, so when ESB is faced with large-scale messages routing, processing efficiency of messages on ESB will directly affect the processing speed of business processes. From an actual project, a coal mine warning system project, it can be verified, when business process described by BPEL is deployed on ESB, the larger the message load, the slower the processing speed of the process. So, relative to integrated interaction, a distributed interaction is hoped. In this paper, a distributed interaction method is presented to complete business process, while services are deployed on application servers. Multiple ESBs are used to share load pressure that on a single ESB. In this method exchange of data among multiple ESBs is a key problem, if exchange can’t be supported, distributed interaction would not be implemented among multiple ESBs. To solve this problem, we propose an enterprise multi-ESBs communication method, on the basis of WS-Notification specification and cloud queue model, we design and implement Web Service notification broker and Subscribe Request Queue cloud service that supports real-time communication among multiple ESBs. Through theoretical analysis and practical application, our method shows better processing efficiency of business process.

Keywords: ESB, Web Services, Cloud Services, WS-Notification, Cloud Queue

1. Introduction

An Enterprise Service Bus is an open standard, message-based, distributed integration infrastructure that provides routing, invocation and mediation services to facilitate the interactions of disparate distributed applications and services in a secure and reliable manner [21]. With the development of Service-Oriented Architecture (SOA), more and more enterprise applications are built on the Enterprise Service Bus (ESB), such as: Apache ServiceMix[18], Mule[19], and so on. In some cases, enterprises will develop their own services or application, so, they need to deploy and manage their services, and ESB provides a convenient way of service deployment and management. Because of the convenient feature, ESBs further promote the universal application of Web services.

In the process of enterprise project development, distributed services have often been combined in a business process, for example, a business process is described by BPEL. Although ESB allows different service providers and consumers to communicate, while services are deployed on application servers, it integrates routing of messages of all services. We call it as distributed services, integrated interaction, for example [21] [22] [23]. In the case of large scale message communication, the load of ESB is too large. So, when ESB is faced with large-scale messages routing, processing efficiency of messages on ESB will directly affect the processing speed of business processes. From our actual project, a coal mine warning system project, it can be verified, when business process described by BPEL is deployed on ESB, the larger the message load, the slower the processing speed of the process. Relative to integrated interaction, we hope a distributed interaction.

To solve this problem, we hope to build communication among multiple ESBs, meanwhile to share load pressure from a single ESB to multiple ESBs. In multiple ESBs, each ESB is responsible for its own message routing, and it doesn’t need to deal with others’. In this way, a distributed interaction can
be got. In this method, exchange of data among multiple ESBs is key problem, if exchange can’t be supported, load pressure on a single ESB would not be shared by multiple ESBs.

In this paper, we focus on communication among multiple ESBs. We propose an enterprise multi-ESBs communication method. On the basis of Web Services Notification Specification[1][2][3], we design and implement Web Services Notification Broker(WS-NBr) to built communication among multiple ESBs. WS-Notification Specification is a Web Services specification about publication and subscription. In pub/sub system, the message routing is usually handled by special routing algorithm, management of routing table is one of the key issues in routing algorithm, adding or removing a routing node is very important for routing table management. Usually adding or removing a routing node need to go through inquiries, registration or cancellation, and matching themes, etc. It is a complex work to expand or reduce scale of nodes, and it will affect system scalability. In addition, in pub/sub systems, usually there is a globe table[5][12] to manage various information, such as themes, address, etc. when the global table is not available, the reliability of the global table directly affects the system reliability. So, in this paper, on the base of our method, we design and implement a Subscription Request Queue (SRQ) cloud service using Hadoop. Especially, pub/sub system based on multicast will be constrained by heterogeneous and distributed network, the SRQ cloud service shows advantage of heterogeneous and distributed. Here, the queue follows some common knowledge of queue cloud services, it is only logical queue, in fact, the storage of message in queue follows the distribution and the principle of multi-machine redundancy backup, which can ensure the reliability of the subscription message. Through this queue, Web Service Broker directly access to the queue for subscriber information. Get node information through the queue, then the join or deletion of nodes is easy; the system is also easy to expand. In addition, in this method, each Web Services Notification broker maintains their own subscription Information, to avoid the use of global subscribers table.

In our real project Coal Mine Warning System, we use our method, through theoretical analysis and practical application, our method shows better processing efficiency of business process.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 then presents our method design. Through an example, section 4 gives approve and performance comparison. Finally, Section 5 concludes.

2. Related work

WS-Notification is a family of documents: WS-BaseNotification(WS-BN) specification[1], WS-brokeredNotification (WS-BrN) specification[2], and WS-Topics (WS-T) specification[3]. Fig. 1 illustrates relationship among the NotificationProducer (NP), NotificationConsumer (NC) and broker.

Fig 1. Relationship of WS-BrN, NP and NC

WS-Notification specification in early application was implemented only based on WS-Basenotification [1] and WS-Topic [3], for example, Apache’s Pubscribe [7], pyGridWare [8], GT4 Globus Toolkit [9]. WS-Messenger[10] is a implementation of all the WSN specifications, as well as of WS-Eventing [11]. WS-Eventing is another publish/subscribe standard for Web services, WSRF.NET supports all of the specifications, and [16] compare these implementations. Paper [17] designs content-based publish/subscribe systems with an efficient mapping between content descriptors and nodes in
the overlay network. The meghdoot system [18] is designed with a locality-preserving mapping, but the multidimensional address space used by its overlay network is costly.

There are also some applications with distributed publication/subscribe middleware, for example, in [6], Ivan M. Delamer applies pub/sub into electronics production based on IPC/CAMX.

Based on a Distributed Hashtabe built on a structured overlay of peer nodes, paper [5] design and implement a distributed content-based notification broker for WS-notification. A Distributed Hashtabe(DHT) is a data structure that associates objects identified with a set of keys with nodes in a distributed system and that provides the mechanisms to store or retrieve these objects through a put/get interface, and DHT is used to map matching topics to the same nodes. The work in [12] addresses this issue to some extent. In this structure, you must maintain relationship between all topics and all nodes, and ensure the reliability of DHT. It is not easy to extend scale of system.

Cloud queue service is driven primarily by Amazon, Microsoft. Amazon's Simple Queue Service (SQS) [13] and Windows Azure Queue service [14] all provide reliable storage and delivery of messages between any clients or computers with access to the Internet. They allows message senders and recipients to interact without having to communicate directly with each other. SQS and Azure Queue service both only support polling mode to extract messages from queue. Amazon Simple Notification Service (Amazon SNS) [15] is a web service that makes it easy to set up, operate, and send notifications from the cloud. It is designed to make web-scale computing easier for developers. The same problem to SNS, you must maintain globe topics.

Queue in queue service isn’t a really queue, it is just a logic queue, and the storage of message in queue follows the distribution and the principle of multi-machine redundancy backup, which can ensure the reliability of message. So, we also can use the advantage of reliability to replace globe table in pub/sub system.

Here, applications based on WS-Notification Specification, through subscription and publication, a business process can be broken down into multiple parts by topic, and then the business process can be completed by multiple ESBs together, while an ESB is only focused on the treatment of a function based on a topic to improve processing efficiency of whole business process.

In this paper, topic is not our focus, so, in an enterprise project development, we assume that developers have some common knowledge of notification topics.

3. Enterprise multi-ESBs communication method

With development of Cloud Computing, we can not only conveniently access to computational resources like Virtual Machines, storage, but also bring new pattern for enterprise solutions. Cloud services are defined as software services that use computation resources [4]. So, enterprise architecture designed by Cloud services has better scalability and reliability. In our method, we design a Subscribe Request Queue (SRQ) Cloud service for message communication among multiple ESBs[21]. On the basis of SRQ, we can build relationship among ESBs. Get ESB node information through the queue, then the join or deletion of nodes is easy; the system is also easy to expand. At the same time, we use WS-NotificationBroker (WS-NBr) to implement real-time exchange of message among different services, as shown in Fig. 2.
In Figure 2, SRQ cloud service in multiple ESBs method consists of three basic parts: WS-NBr, Queue, and message. The WS-NBr is responsible for receiving services’ registration, notification and subscription, and these services can be located on different ESBs or on the same ESB. WS-NBr is integrated in ESB, user’s requests are routed to WS-NBr through ESB, and WS-NBr also route message to subscriber (services or components) through ESB.

As shown in Figure 3, the queue is a communication bridge among distributed WS-NBrs, and it is also a message container. The queue stores users’ subscribe request as message.

![Figure 3. Subscribe Request Queue Model](image)

Figure 3. Subscribe Request Queue Model

Here, the queue is built on the basis of Hadoop, and it follows some common knowledge of queue cloud services, it is only logical queue, in fact, and the storage of message in queue follows the distribution and the principle of multi-machine redundancy backup, which can ensure the reliability of the subscription message. One message in queue is stored on multiple physical machines, and when message consumer asks to return an available message in the queue, system extracts message from stored server to message receiver.

Figure 4 shows the lifecycle of message.

![Figure 4. Lifecycle of Message](image)

Figure 4. Lifecycle of Message

There are two basic kinds of message in multiple ESBs method: subscribe request message and notification message. The format of all these messages is consistent with WSN message specification. Subscribe request is produced by user, user sends this subscribe request to WS-NBr (step 1), then this WS-NBr sends the subscribe request to queue (step 2). WS-NBr only cares about which WS-NBr subscribes its message, do not care about which real user or service subscribes it, and user only needs to send subscribe request to WS-NBr, do not care where the notification message comes from and how comes. The notification message is produced by users or services, user sends notification message to WS-NBr (step 4). According to address of subscribe WS-NBr (step 3), the WS-NBr sends notification message to those subscribe WS-NBr (step 5), then those WS-NBr notify their user or services (step 6). The flow of message is shown in Figure 5.
4. Approve and performance comparison

Through the above message processing steps, we can get a distributed interaction, relative to the way of integrated interaction (communication among distributed services through a single ESB), we can prove our messages processing speed will be better and faster.

Assuming ESB can handle certain number messages per unit time, the number can be set to \( I \).

Assuming in integrated interaction, a business process includes \( m \) services, \( S_1, S_2, \ldots, S_m \), as shown in Figure 6.

Assuming there are \( N \) request messages, we set the output message number of \( S_i \) is \( N_i \), then, ESB needs to handle \( N + N_1 + N_2 + \ldots + N_{m-1} \) messages when ESB finished the whole business process, because \( I \) is a fixed number, so, ESB needs \( \sum_{i=0}^{m-1} N_i = T \) units time to finish the business process.

In distributed interaction, we assume that there are \( m \) services on \( m \) ESBs, as shown in Figure 7.

Because these ESBs are distributed, their request can be processed in parallel. Here, we consider the worst case.

These ESBs handle their requests by sequence. That is, when one ESB finished to process all requests, the next ESB started processing its input messages.

We still assume that there are \( N \) request messages, the output message number of \( S_i \) is \( N_i \), In the worst case, \( ESB_i \) will have \( N_i \) output messages to next \( ESB_{i+1} \) in one time, the \( ESB_i \) requires \( \sum_{i=0}^{m-1} N_i = T \) units time, so, in the worst case, all ESBs need
Enterprise-oriented Communication among Multiple ESBs based on WS-Notification and Cloud Queue Model
Gu Pingli, Shang Yanlei, Chen Junliang, Deng Miaoting, Lin Bojia
International Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011

\[
\frac{N}{I} + \frac{N_1}{I} + \ldots + \frac{N_{m-1}}{I} = \frac{N+N_1+\ldots+N_{m-1}}{I} = \sum_{i=0}^{m-1} \frac{N_i}{I} = T
\]

units time to finish the whole business process.

We assume their requests can be processed in parallel, all ESBs need \(T^*\) units time to finish the whole business process. Then

\[
T^* < \frac{N}{I} + \frac{N_1}{I} + \ldots + \frac{N_{m-1}}{I} = \frac{N+N_1+\ldots+N_{m-1}}{I} = T
\]

Then,

\[
T^* < T
\]  \hspace{1cm} (1)

Form (1), we can see that when there is the same number of requests to be processed, the multiple ESBs system required less time than integrated interaction system; on the contrary, in a given time interval, the number of requests can be handled in multiple ESBs system is greater than in integrated interaction system. It means that our messages processing speed will be better and faster.

To verify the feasibility of our process, we implement this method, and our method is used in Coal Mine Warning System (CMWS) project. Through practical application, our method shows better processing efficiency of business process.

We choose Servicemix as our ESB, and services are deployed on multi-Servicemix. Our method is implemented in Java. We give the performance comparison between integrated interaction and distributed interaction.

In CMWS project, we have business process:

![Fig 8. Business Process of Coal Mine Warning](image)

In Figure 8, service S1 collects sensor data from KJ95 every 10 seconds once, so S2 is necessary to process data every 10 seconds, S1 and S2 all have a lot of data to process. In the whole process, there will have a large number of messages to interact with each other.

According to the conclusion we have already proved, formula (1), we assume there are 10000 requests, and we will compare processing speed on integrated interaction system and distributed interaction system. Here, on the one hand, in integrated interaction system, we use BPEL to combine service S1and service S2 on one ESB, we call this ESB as ESB1. On the other hand, in distributed interaction, our multiple ESBs communication method, we have two ESBs to deploy service S1 and service S2, we call the two ESBs as ESBa and ESBb. And we use two topics: CH4 and CO, to communicate between ESBa and ESBb. ESBa not only collection sensors data, but also deal with data with topic CH4, ESBb only handle data with topic CO. The requests number of CH4 and CO holds 50% each. We use performance tool LoadRunner to compare them.

From Figure 9, we can see that ESB1 needs 3 minutes and 20 seconds to complete all requests, and the max hits per second is 69, and then we can get the average hits per second 48.309.
From Figure 10, we can get the average response time is 0.136 seconds.

For ESBa and ESBb, here, because ESBa and ESBb are independently and (can be) simultaneously with each other. So, we choose the longest message load ESB as our comparison ESB. ESBa is the longest running time ESB. Figure 11 shows that all ESBs only need 1 minute 52 seconds to complete all requests, and the max hits per second is 117.6 on ESBa, and then we can get the average hits per second 85.47.

From Figure 12, we can get the average response time is 0.052 and 0.053 on ESBa and ESBb.
From our experiment data, with the precondition of the same requests number, ESB1 (integrated interaction system) needs 3 minutes and 20 seconds to complete all requests, all ESBs (distributed interaction system) only need 1 minute 52 seconds to complete all requests. So, processing speed of integrated interaction system is slower than that of distributed interaction system. Our enterprise multiple ESBs communication method shows better processing efficiency of business process.

5. Conclusions

In this paper, we have presented an enterprise multiple ESBs communication method, a method based on WS-Notification specification and cloud queue model, we design and implement Web Service notification broker and Subscribe Request Queue cloud service that supports real-time communication among multiple ESBs. Firstly, we introduced enterprise multiple ESBs communication method, on the basis of WS-Notification specification and cloud queue model, we design and implement our Subscribe Request Queue cloud service model. This cloud service consists of three basic parts: WS-Notification Broker, Queue, and message. Secondly, on the basis of our method, we presented a workflow of running composition services. Finally, we give a performance comparison between integrated interaction and distributed interaction. Through theoretical analysis and practical application, our method shows better processing efficiency of business process and better scalability.

6. Acknowledgements

We gratefully acknowledge the support of “973” program of The National Grand Fundamental Research 973 Program of China (Grant No. 2011CB302506); The Novel Mobile Service Control Network Architecture and Key Technologies (Grant No.2010ZX03004-001); National Natural Science Foundation of China (No. 61001118) and the Fundamental Research Funds for the Central Universities (BUPT Grant No.2011RC0502).

An earlier version of this paper (ECB: Enterprise Cloud Bus based on WS-Notification and Cloud Queue Model) was presented at the 2011 IEEE International Workshop on Management in Cloud Computing (MCC 2011). In this paper, we continue to complete our work and improve some description about our method, add theory demonstrate and performance comparison about our method to make our method more complete presentation.

7. References

Enterprise-oriented Communication among Multiple ESBs based on WS-Notification and Cloud Queue Model
Gu Pingli, Shang Yanlei, Chen Junliang, Deng Miaoting, Lin Bojia
International Journal of Advancements in Computing Technology Volume 3, Number 7, August 2011