A New Message Middleware for Business Process Engine Integration Based On Publish/Subscribe Model

DaLei Zhou[1, a], Shuai Zhao[1, b] and Bo Cheng[1, c]

[1]Beijing University of Posts and Telecommunications

State Key Laboratory of Networking and Switching Technology

Beijing, China

[a]cauc_09@foxmail.com, [b]zhaoshuaiby@bupt.edu.cn, [c]chengbo@ bupt.edu.cn

Keywords: Publish/subscribe. BPM. EAI.

Abstract. This paper presents an outline on architecture and performance of a new message middleware based on publish/subscribe model. This architectural uses topic tree to organize topics in the publish/subscribe system. The system collects network topology information and then forward messages after routing calculation. By plugging this message middleware into the handle of jBPM, which can support message interaction between multi process engines, the multi process engine can be integrated. At last, the evaluation work focuses on studying the performance of topic tree and message routing.

Introduction

Business Process Management (BPM) is the discipline which combines knowledge from information technology and knowledge from management sciences and applies this to operational business processes [1, 2]. The trend of economic globalization promotes the development of information technology, the business management system of enterprise needs to adapt to the new requirements. Seamless combination of business processes between different organizations is significant for enterprises to adapt to the changeable business environment, so the enterprises can remain competitive in the international market. The enterprises establish the cooperation relations through the cooperation system to combine resources and share information to achieve common goals. The integration of multi business processes across the enterprise is the key to promote the establishment of good business cooperation. Process integration cross enterprises dynamically comply with the development trend of business management technology.

As most companies and organizations reach a certain size enterprise application integration (EAI) is a continuous challenge [3]. Process integration cross enterprise requires real-time and wide area distribution of communication service infrastructure as the underlying support services. There are many ways to realize the interaction between systems, such as share database, establish system data exchange package, interactive mode based on publish /subscribe topics, Interactive mode based on content, Interactive mode based on message [4].

In this paper, we propose a novel architecture of message middleware based on publish/subscribe model. The middleware consists of two layers. The bottom layer is a distributed event bus for network, and the upper layer is the interface layer for different services. Here we focus on the interface layer to design and implement interface services to build a high performance and high reliability of the publish/subscribe system.

The structure of this paper is as follows: Section 2 discusses background and related work. Section 3 proposes the architectural approach with details of its principle and technical solution. Section 4 evaluates the important part of system with two experiments. Conclusion is given in Section 5.

Related works

jBPM. jBPM is a flexible Business Process Management (BPM) Suite. It becomes the bridge between business analysts and developers. jBPM provides the way to manage the process which both business users and developers prefer. Send Message and Receive Message in jBPM is an abstract definition, the BPMN2 specification doesn't define exactly what it means. Plugging message middleware into the process engine is a way to integrate different process engines.

Publish/subscribe. Well adapted to the loosely coupled nature of distributed interaction in large-scale applications, the publish/subscribe communication model has recently received increasing attention [5]. The standard for publish/subscribe system based on Service Web was proposed by OASIS in 2006 and it has not been widely studied. The type of push-up publish/subscribe system is based on the WSN which support the initiative push and the keep semantic order. But its performance is lower, and the message throughput of system is very small, the function is also a bit poor, which cannot adapt to the complex situation in the real production environment.

Architecture and implementation

In the following section we highlight our architectural of the publish/subscribe message middleware, providing as well a brief overview of the system implementation.

Architecture. Each message in the publish/subscribe system has its own topic. The system will push the message to the subscriber according to routing information when the message's topic is exactly matching to the subscriber's interest. Traditional Message middleware often based on the list structured is inefficiently in simple matching. This paper proposes a new approach, called Topic Tree, to organize all the topics. The subscribers can subscribe one or more topics by requesting the whole tree or any sub tree at a time. Besides, Topic Tree can also organize the interrelated subjects together to simplify the management and merge process as well as to reduce the redundancy of information. The corresponding optimization problem can be efficiently solved since it provides a structured name system.

Publish/subscribe system is composed of an administrator and some clusters. Administrator maintains fundamental information about every group, e.g., address information of representative. Administrator will provide or update the basic information when new nodes of clusters add into a cluster. Administrator loads all the information about Topic Tree at startup, thus it can distribute or update the corresponding information of Topic Tress once the command of update or retrieval performed.

Each cluster consists of a representative and a plurality of agents. Representatives collects all of subscribe and unsubscribe messages in its cluster, and be responsible for merge or spread to the whole network. Representatives publish State Advertisement Link (LSA) periodically, which contains the link information and subscription information to update subscription information for the whole network. At the same time, representatives store the subscription information of all other clusters, and according to that information to calculate the forwarding tree for each topic.

Agent belongs to a cluster. Agents and representatives are essentially similar. Both of them can provide subscribe, publish and receive services for clients. Agent is a candidate of representative. When the representative of a cluster is down, the new representative will be elected from the agents and the other cluster and then the administrator will be notified to update the information of the representative of this cluster.

Every client connects to a representative or an agent, and it can publish or subscribe messages. Then clients can subscribe topics what they are interested in, publish a message or create a topic. Plug the client into the process node of jBPM making the jBPM nodes which need to send and receive messages becoming the client of publish/subscribe system as shown in Fig. 1. The integration between the internal and external process engines can be achieved through the messages interactions.

Topology construction. Topology is the basis of the whole network, which ensures the normal operations of the network, so that the flooding of subscription message, consistency calculation of routing and event notification can be spread normally.

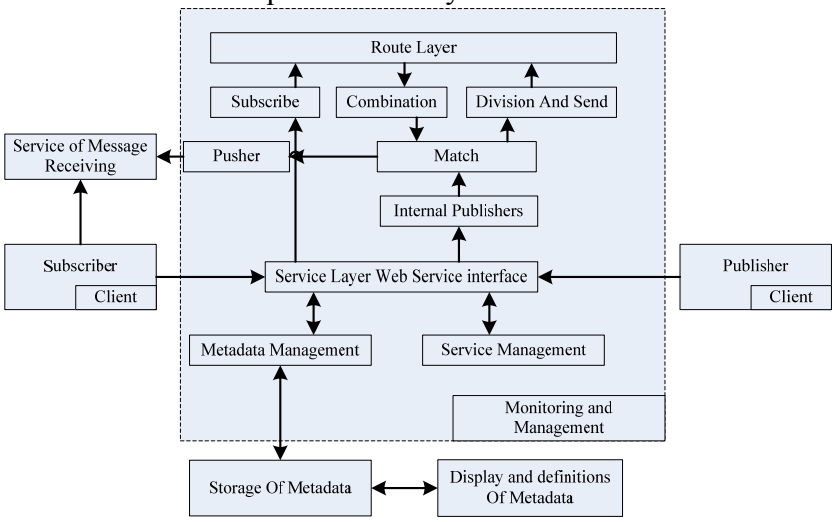

Fig. 1 Client plug into jBPM

Topology maintenance includes the selection and construction of neighbors, maintenance, and the maintenance of the topology information within the cluster. When choosing a neighbor, representatives consider the distance between the cluster and the number of its neighbors, and select the appropriate target and start to build neighbor maintenance after getting positive feedback. The neighbor is the basis of the routing computation. In the real environment, because of the consideration of routing calculation cost, the maintenance cost and the physical connection situation, a fully connected topological graph cannot be constructed. A number of appropriate neighbors should be chosen to maintain the Topology due to the actual situation. The setting of the number of neighbors in the cluster needs to be considered the physical link status between clusters and the total number of clusters, in order to reduce the unnecessary cost when construction the topology or the topology maintenance costs when there are too many links. The effective detection of the nodes mainly depends on the maintenance of the Hello detection mechanism between the nodes. When the link state changes, the LSA message is required to broadcast the change link, and each node needs to provide the corresponding topology storage to support routing information.

When the new agent server starts up, it will request the cluster information of the current network from the administrator. If the cluster information exists, this agent becomes the member of this cluster automatically, and then it will interacts with the cluster representatives maintain topology. If there is no information of its cluster or the original representative has failed then this agent become a new representative, and join the network as a new cluster. When the agent server is added into the network as a representative, it needs to select a few adjacent points for the request and receive the feedback information to build neighbors. And then the entire network topology information and subscription information will be request and stored in local format from one of the neighbors. Hello maintenance mechanism will start after neighbor relationships established. After the adding into the network, the new representative shall generate LSA about its cluster and spread it to the whole network.

When the representative of the cluster fails, the other clusters will add it to the waiting queue according to the timeout of the Hello message, and then wait for the re-election of the fail cluster. If any other agent within the cluster is worked, the cluster will choose a new representative. The new representative will inform the other members of the cluster as well as inform to the whole network by the neighbors of the cluster. At last, the representative information of the cluster will be updated in Administrator. Accordingly, when one of a neighbor has failed, the representative will wait for reelection, and remove the fail neighbor only when after the waiting time out.

Hello messages within the cluster are transmitted from each agent to the representative, and the cluster representative transmitted it by multicasting cluster. When a common agent fails within a cluster, the cluster representative will know its failure due to its timeout of Hello messages. Then the

representative cancels the subscription information of the fail agent and notifies the other agent within the cluster. Representatives determine whether the failure of the agent has affected the contents of the original cluster subscription table, any changes required to send the LSA notification to other clusters to update subscription information.

Message routing. Message routing of publish / subscribe system means to establish a routing path for each event, making sure that events can arrive at the subscribers correctly. Like the design of the system architecture, the routing of the publish/subscribe system is similar to the traditional network data routing. The difference is that the publish/subscribe system depends on the content of the event, including the title of the event and IP [6].

Routing calculation is based on topology structure and topic subscription. In this system, using Dijkstra algorithm to calculate the routing tree for each topic with policy information based on limit the number of next hops. When forwarding the data for each event, routers will call the route query interface to get the next hop. Then the query interface returns the next hop, and at the same time it records the total forwarding quantity of the cluster to the next hop cluster, which is the traffic flow between the cluster and the neighbor cluster. In the process of routing computation, all the topics are obtained from the subscription table, and the routing tree is computed for each topic when the topology information of the complete network will be used, and the update of the topological information is related to the traffic statistics. So the traffic information between clusters will also affect the results of calculation. It is key problem to statistic the traffic flow when statistic traffic information in the unit time. The distance between the cluster and the neighbor cluster is plus or minus a certain value according to the change of traffic flow, and then which is then spread the changes by LSA to the network for updating the topology.

Experiment and evaluation

Using the topic tree structure is a core feature of our design. The performance of the topic tree has a great influence on the performance of the whole system. Therefore, we designed a performance test for the topic tree specifically. In order to simplify the output information, the subscriber client would print a counter and timestamp only after received 500 messages. We use the MPS (Message Per Second) to evaluate the performance of the system, the calculation formula is:

$$MPS = \frac{\sum_{t=start}^{current} notification_t}{current - start}. \quad (1)$$

Three cases have been setting, the difference between first one and second one is that the depth of the topic tree. Case 2 and case 3 have the same parent topics in the topic tree, but the sub topics in case 3 were stored into linear table to test the effect to the performance when traversing the linear table of the topic tree. Test results for three cases are shown in Fig. 2 to Fig. 4.

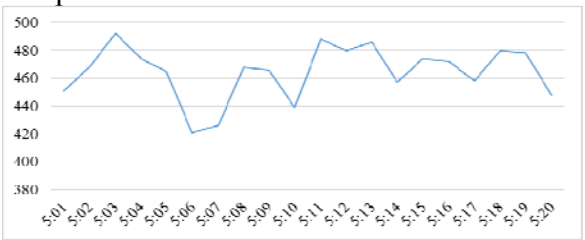

Fig. 2 Result for case 1

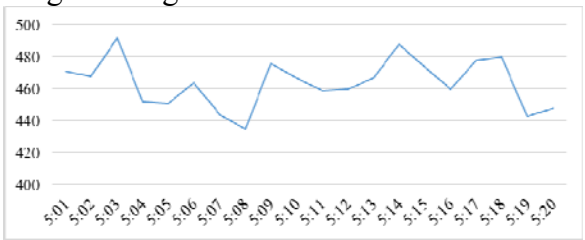

Fig. 3 Result for case 2

The figures show that the performance of the system has little difference between case1 and case 2, and the average value of MPS is about 460 in the two cases. When compared to case 2, the performance of case 3 is slightly lower. The results show that the performance of the system is not affected by the size of the subject tree in a relatively small scale. Because of the tree structure of the topic tree, it is able to store large amounts of data when the depth is not large and the small scale of the sub topic linear table. We can draw a conclusion that the extensions on horizontal and vertical on topic tree will not significantly affect the performance of the publish/subscribe system.

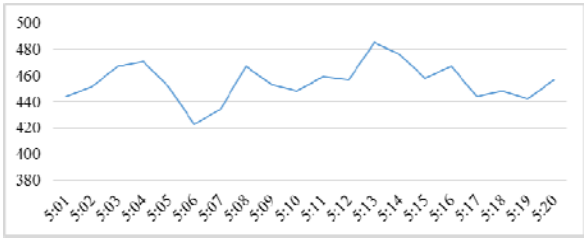

Fig. 4 Result for case 3

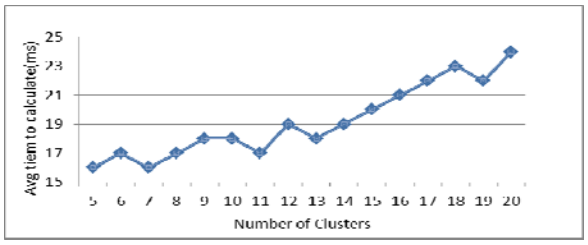

Fig. 5 Average time for route calculation

With the increase number of the cluster, the network topology becomes larger, and the time of route calculation is also increased. Therefore, the speed of routing is also a measure of whether the network design can be expanded. Fig. 5 shows the average time of route calculation corresponding to 5 to 20 subscribe clusters. It can be seen from the figure, the required routing computation time wave in range 15ms to 25ms, and the change is random. It shows that there is no obvious positive correlation between time of route computing and cluster numbers. It can be predicted that the time cost of route computing will not increase significantly when the number of clusters increase in a limited range. But the situation of large number has yet to be verified.

Conclusions

The system proposed in this paper is a high performance and high reliability of the publish/subscribe message middleware. In an information system, it plays the role of the message distributor. Based on this system, the user can develop a loose coupling, high efficiency, and asynchronous reliable application. With the use of tree structure, the performance of the topic tree is not significantly affected by the horizontal and vertical extension.

In this system, the format of data exchange between clients and publish/subscribe system is SOAP message. Json and ProtoBuf are the most popular data serialization technology currently in industry, and they can replace the XML to implement a similar function with a higher efficiency. From this perspective, we can utilize these techniques to improve the system performance in the future.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (Grant No. 61501048); China Postdoctoral Science Foundation funded project (Grant No. 2015M570060).

References

[1] W. M. P. van der Aalst, “Business process management demystified: a tutorial on models, systems and standards for workflow management,” in Lectures on Concurrency and Petri Nets, J. Desel, W. Reisig, and G. Rozenberg, Eds., vol. 3098 of Lecture Notes in Computer Science, pp. 1–65, Springer-Verlag, Berlin, Germany, 2004.

[2] M. Weske: Business Process Management-Concepts, Languages, Architectures, Springer-Verlag, New York, Inc., 2007.

[3] Rodney Gleghorn, “Enterprise Application Integration: A Manager's Perspective”, IT Professional, vol.7, no. 6, pp. 17-23.

[4] Dayal, U., Hsu, M., Ladin, R.: Business process coordination: State of the art, trends, and open issues. In The 27th VLDB Conference, Roma, Italy 2001.

[5] ¨P. Eugster, P. Felber, R. Guerraoui and A. Kermarrec: The Many Faces of Publish/Subscribe. In ACM Computing Surveys, Vol. 35, No.2, June 2003.

[6] Chockler G, Melamed R, Tock Y, et al: Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication[C]. Proceedings of the 2007 inaugural international conference on Distributed event-based systems. ACM, 2007:14-25.