PN Santhosh kumar ,INDIA / International Journal of Research and Computational Technology, Vol.7 Issue.4
ISSN: 0975-5662, May, 2015 www.ijrct.org

Distributed Data Mining Workload Manager

P N santosh kumar. Asst Prof ,ECM dept, SNIST, hyd.
P.CHAKRADHAR,
N. ABHINAY – mail.-pnsk47@gmail.com

U.G. FINAL YEAR STUDENTS OF ECM, SNIST ,HYDERABAD,INDIA.

Abstract

The objective of this paper is to develop the web service console that performs the operations like [Send, Receive, Load, and Suspend] for the web services that is located in the distributed environment. Compilation and execution of the code can be done in a single window. The console is mainly used to manage the web services that are located in various places. Based on input the web services are chosen for manipulation and hence the console will automatically load the input to web services and obtain the output, this paper also presents how the developed console allows a safer collaboration between Web services with respect to integration where there is consideration of the input as criteria for allocating tasks.

Keywords: Web Services, Collaboration, Monitoring, Tasks Allocation.

1 Introduction

Web services are the standard technology in achieving distributed computing and a means by which peer-to-peer collaboration can be achieved. This success created several challenges at the industrial level and research level. One of these challenges is how to develop a successful console by integrating the Web services. Web services composition can be seen as collaboration between composite and component Web services to achieve some task.

A console can be defined as a monitor between Web services. Each one has a specific function to be achieved on the basis of allocated task. Tasks allocation is an input for this paper and it has been given by any client presented in the distributed environment, the console that we develop must be automatically identify the input task and search for the available web services and allocate the input. If the size of input is too large then the console splits the input into equal sizes and allocate to the available web services. Hence the allocation of input, compilation and execution can be viewed in the developed console and status of web services are also monitored in the same window.

The result of previous research papers helps us for tasks allocation for agents which include integration of Web service trust in order to guarantee a high level of trustworthiness in Web services collaborations. In some approach, each agent is defined by a set of services. It builds, from the set of services. When an agent receives a request for composing web services, it first computes the best partial plan. Then it coordinates with other agents by merging partial plans in order to find a global plan that responds to the submitted request. Each agent computes its best partial plan, using a distributed heuristic function. This function computes the distance between the local state and goal state, considering intermediate local plans of the other agents.

The contributions of this paper are threefold:

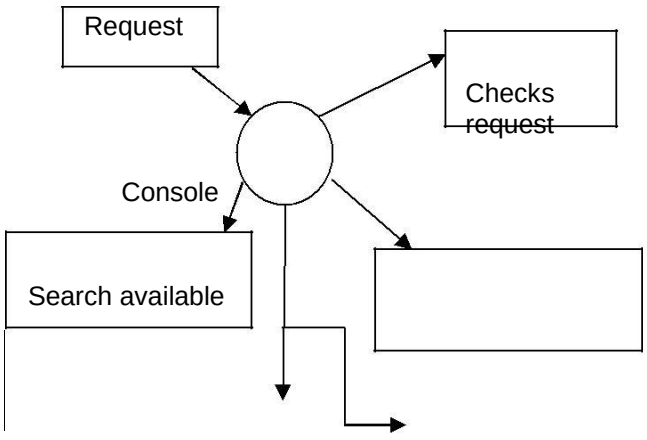
- To provide a clear console to display the status of the available web services in the distributed environment.
- To provide some experiments that show the allocation of task to the web services this has been received as an agent request.
- Integration of web services that runs on different web server in the distributed environment

The rest of the paper is organized as follows. Section 2 is dedicated to the presentation of the needed background for this paper. In Section 3, we discuss the related work. Section 4 is devoted to the presentation of our console for Web services integration. Section 5 outlines the proof of concept and the related results. Section 6 outlines the future work finally; we draw some conclusions in Section 7.

2 Background

The general setting of a mechanism design problem is as follows: There is lack of tool support for agent composition and implementation; failure to provide standard compliant, inter operable environment; difficult in deployment, monitoring, performance evaluation using common tools sets; lack support versioning of components as in web services; failure to integrate seamlessly with legacy system. Industry adoption is limited. Hence a separate strategy is followed to overcome all those things that are involved in developing the console.

Work flow



web services Splits into sub process if needed

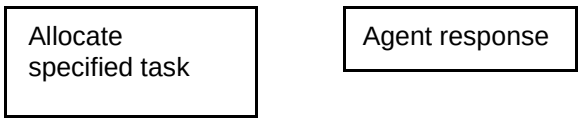


Figure 2.1 Workflow

The fig 2.1 clearly states the workflow of this paper and illustrates the job of developed console in which the client request is given to console and then console checks the input is valid or not and then search for available web services then it allocates the task for agent. If the input is too large then console will automatically splits the input into equal sizes.

The console development is mainly based on the java web services that run on different web application server and the client side programming is written on java language and the server side programming is developed using application server to create the console and that must be operated in the distributed environment. Hence we choose the j2ee and to develop the console for interoperability and to support the distributed environment.

3 Related Work

In this section we present some recent works on the composition of web services in the distributed environment and collaboration, integration of web services and event-based monitoring.

Few trust and reputation models for Web services [1,2,4],Maroua Bouzid [1] a general framework of distributed planning techniques for web services composition is introduced. We describe in this section its basic principles and

limits. The web services composition domain is defined by a set of web services *{WS\,..., WS_n).* An agent *Ai* is associated with each *WSi* . *A* Central Agent A_c plays the role of an interface between a client request *R = (init, goal)* and the composition system, and manages a portion of the coordination between the agents. When a client sends to the system a request *R,* the central agent A_c broadcasts *R* to all the agents Ai

Paradesi et al. [2] devised a trust model for composition of Web services. The model is based on a Bayesian formalization of the trust and it is updated depending on experiences with Web services. Based on trust values of individual Web services and on typical composition operators, they derive trust for Web services compositions.

John A. Miller [4] determines the Mash up execution ends at the root level by the dispatch operator which sends result to the end-user. Several operators in between the root level and the leaf level exist which process data and refine it based on end-user needs, such as filter and sort operators. Each component in a mash up has a string representation that defines the component and it is constructed by concatenating the representation of compo-nent's attributes.

To the best of our knowledge, there is no other console approach to compose web services. The main difference between those methods is: in the former CASCOM architecture, there is only one planning agent and the other agents are involved in other tasks (selection, execution and monitoring of services) to help the planning agent. In the latter approaches, however, a planning agent is associated with each web service. Therefore the load is shared among several agents.

These models paved the way to understand the issues behind using Web services either individually or in com-position. We go forward these initiatives by focusing on two aspects:

The first thing is to develop the console for distributed environment and the second one is the integration of Web services that runs on different web server. Our intent is to provide a Work load manager in which the objective is to assign tasks to Web services in a way that maximizes the likelihood of performing successfully these tasks.

4 Distributed Workload Manager

This section presents the formulation of the Web service workload manager based on the generic model that already presented in background session. Also introduces a console that will be followed in the rest of the paper.
The detailed about this approach are given as follows. The workload manager is mainly designed for the distributed environment and three types of web servers can be chosen in which these three services running successfully on each, specific functions for each services are described, here a console will be developed that monitors these three web services and displays the status of each, in this console the integration of web services are done and hence these contains options like send, receive, load and suspend.

The **load** option is mainly used to load the input from the agent which has been received from the agent through the console. The **send** option is used to send the completed output to the agent. The **receive** option is used by the console to receive the input from the agent and we have on more option in the console named as **suspend** which is mainly used to stop the current process and insert the new request from the agent in case of importance or emergency.

Thus the developed console has the four important options which are mainly used in the distributed environment. The compilation and execution of the agent code can be done in a single window of the developed console and the furnished workload manager can manage the request from the agent located in the distributed environment.

5 Experiments and Results

The developed workload manager is subjected to withstand the interoperability of web services and maintain the collaboration of web services. The first web service is addition math service that runs on sun app server in one local machine, the second web service is the subtraction math service that runs on the tomcat app server and the third web service is the multiplication math web service that runs on the web server on another machine that are connected in the network

The request from the agent is received by the manager and checks out the input that is a valid input for the web services and if the input is not valid it certainly displays the message like invalid input and if the input is too large to process then the manager itself splits the input into the equal sizes and allocates the agent request to the available web services.

Let us show the overview of the workload manager in which the agent request is processed and the output is displayed in the same window applicable for the distributed environment.

Fig 5.1 represents the developed workload manager and is shown below

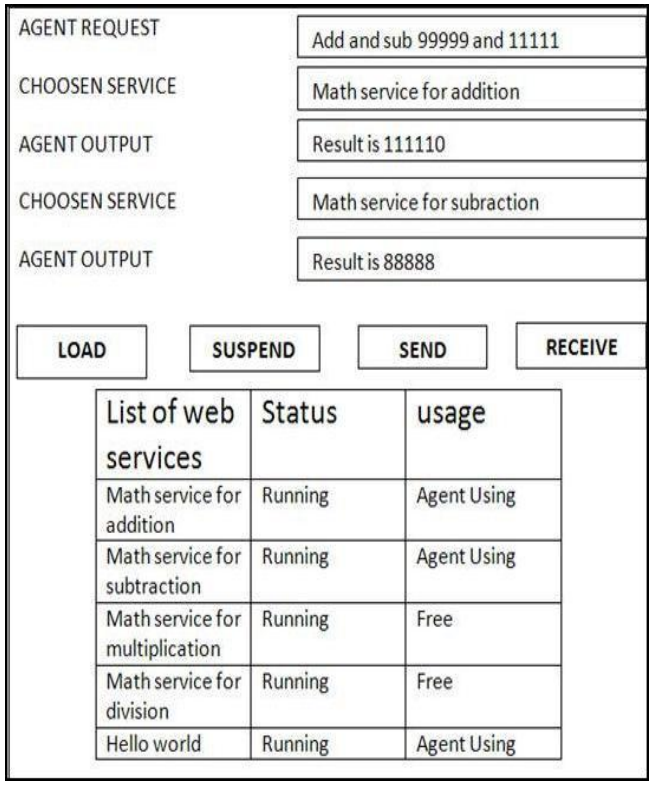


Figure 5.1 Distributed Workload Manager

6 Future Work

It can extensible from the distributed workload manager to the field of bio-informatics. In this area, the molecule separation, pattern matching of the particles can be easily done using the distributed work load manager; the protein structure can be also predicted using the developed workload manager.

The combination of proteins will be quick but it won't have a separate structure, the formation of protein structure is a challenging problem, the ability to identify protein-protein-protein binding sites has important implications for drug design and understanding cell activity. Thus the SVM algorithm in combination with Bayesian method is used to construct the interface between the protein structures.

The workload manager can be useful to calculate the interface method and to integrate the proteins presented in the distributed environment. The distributed data mining workload manager mainly focuses on prediction of protein structures and drug discovery.

7. Conclusion

In this new approach, a console is associated with each web service. Each web service in the distributed environment are clearly stated or listed in the console, the console format is shown in the fig 5.1 Also creates links between its own services and the services of all other agents in the console. With this, the response time of the system is decreased, since a large part of planning computation is made offline and only once for all the requests. Agents

collaborate by merging their best plans to reach the goal quickly.

Finally, the model has been enhanced through the area of bio-informatics and to include the data mining concept for the protein structure prediction and drug discovery, hence the distributed data mining work load manager is developed in order to integrate the web services and make the agent request simple to avoid conflicts with other agents being concurrently executed over the community of web services.

References:

[1] S. Meilin, Y. Guangxin, X. Yong, "Workflow Management Systems: A Survey". Proceedings of IEEE International Conference on Communication Technology, 1998.

[2] Zh. Piefa, H. Kaihu, Zh. Jinfeng, Q.Liang. chun,"Research on enterprise workflow manager system based on Web," Development & Innovation of Machinery & Electrical Products, 2006, 19(5), pp.31-33.

[3] Zh. Hongshan, "A design of workflow management system based on Web,"Journal of Capital Normal University (Natural Science Edition), 2006, 27(2):12-14.

[4] S. Dalal, S. Temel, M. Little, M. Potts, and J. Webber, "Coordinating Business Transactions on the Web," IEEE Internet Computing, vol. 7, no. 1, pp. 30-39, Jan./Feb. 2003.

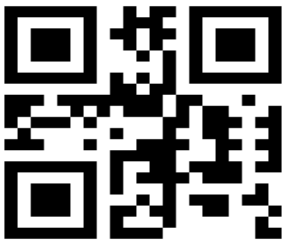
[5] M.P. Papazoglou, "Web Services and Business Transactions," World Wide Web, vol. 6, no. 1, pp. 49-91, 2003.

[5] B. Limthanmaphon and Y. Zhang, "Web Service Composition Transaction Management," Proc. Australasian Database Conf. (ADC '04), pp. 171-179, 2004.

[6] A. Elmagarmid, Transaction Models for Advanced Database Applications. Morgan-Kaufmann, 1992.

[7] G. Weikum and G. Vossen, Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan-Kaufmann, 2002.

[8] T. Mikalsen, T. Tai, and I. Rouvellou, "Transactional Attitudes: Reliable Composition of Autonomous Web Services," Proc. Workshop.



PN Santhosh kumar ,INDIA / International Journal of Research and Computational Technology, Vol.7 Issue.4
ISSN: 0975-5662, May, 2015
www.ijrct.org