IJITKM Volume 7 • Number 1 • December 2013 pp. 103-107 (ISSN 0973-4414)

BPEL based scheduling in multi-agent system for business applications

[1] Surjeet Dalal, [2] Dr. Gundeep Tanwar, [3] Dr. Kamal Kumar Sharma

[1]Assistant Professor, E-max Institute of Engg. & tech. Ambala, India
[2]Associate Professor, BRCM College of Engg. & tech. Bhiwani, India
[3] E-max School of Engg. & Applied Research. Ambala, India

surjeetdalalcse@gmail.com, mr.tanwar@gmail.com, kamalsharma111@gmail.com

Abstract: To develop the proficient and full-bodied business application in the multifaceted business scenario is the leading confront for software developers. Mostly business applications consists the scheduling phenomenon for managing the numerous economic transaction during their operation. Hence to run the operation smooth and robustly, the multi-agent system is found as the supplementary appropriate alternative for implementing and designing the business applications.
In this paper, the BPEL based scheduling is being illustrated for handling the scheduling problem in MAS based business application. This approach provides more efficient mechanism for resolving this complicated problem.
Keywords: MAS, Scheduling problem, business applications, BPEL**.**

INTRODUCTION

Intelligent agents began to appear in computer science and artificial intelligence (AI) prose in the late 1980s as an outcome of work within the objected orientation and distributed AI fields. Despite over two decades of history, a description for the term agent still remains debated. Schleiffer et al. (2005) declared that "intelligent agent technology is the articulation of human decision-making activities in the form of a computer program" [1]. While this definition is graceful, it is not sufficient in that it does not explicitly specify the characteristics of human behavior agents seek to imitate. Wooldridge & Jennings (1995) put forward four distinct characteristics, namely: sovereignty, social ability, reactivity, and pro-activeness. These characteristics are widely accepted as they are at the heart of what agents represent as the human decision-making processes. This set of four properties has been prolonged on extensively over the years & across multiple fields [2].

Most early publications on agents cover work on single agent systems, which are agents that assemble information on behalf of a user, or do specific tasks for them. Multi-agent systems consisting of multiple agents interacting with each other and their environment are known as multi-agent systems. In such systems, not all agents are the same: each agent can have unique capabilities and objectives, representing its real-world complement. A multi-agent system is an assembly of different agents, with different roles, capabilities and goals – for different categories of agents.
In a multi-agent system the agent construct becomes additional than just an entity performing local responsibilities. The agent must also acquire the capability to communicate and synchronize. The important characteristics of a multi-agent system are given below:

- Agents need each other for wholeness of information and problem solving
- No global control system
- Data is decentralized
- Asynchronous computation
- Modularity
- The possibility to entrench multi-objective functions

The fact that intend can be a step wise procedure, as supplementary settlement of MAS [3]. Wooldridge et al. (2005) listed the three main potentials offered by multi-agent systems.

- First, a MAS system resembles the organization of the business itself, making it easier for programmers and analysts to understand its role and actions.
- Second, problem solving in the system is based on problem solving in the organization (decentralized: no "agent" owns the whole system).
- Third, because agents are autonomous and always active, the system is responsive to changes and problems [4].

These facts indicate that the multi-agent system is powerful in handling the business process during their operation.

I. SCHEDULING PROBLEM

The scheduling algorithms can be characterized by the following parameters

1. Hard real-time versus Soft real-time
2. Preemptive versus Non-preemptive scheduling
3. Dynamic versus Static
4. Centralized versus Decentralized

These algorithms endeavor to schedule a set of tasks for either a single processor or multiple processors. They are most anxious with the timing constraints each task has allied with its execution. Each task will cover a deadline before which it must be executed. Guarantees on meeting these timing constraints and how the system handles those tasks that cannot meet their deadline, differ based on which of the above characteristics the algorithm possesses. Hard real time defines those systems that require a 100% guarantee that all tasks meet their deadlines. Soft real time systems are more lax.

In a hard real time system a task has a negative value if it exceeds its deadline. In addition, it may even have ruinous consequences. In a soft real time system the task still has value, although that value is reduced. Soft real time is normally characterized as an "as close as possible" approach.

Preemptive and non-preemptive algorithms differ in their handling of task execution. A preemptive algorithm has the ability to suspend a task that is currently being executed so a task of a greater priority can execute first. Non-preemptive scheduling does not have this ability so all tasks are executed to completion once started.

Dynamic and static algorithms are different based upon when they make decisions about scheduling. A dynamic algorithm makes these decisions "on the fly" during execution. Static algorithms make all scheduling decisions before run time. For example, these decisions may be stored in a table. When a task enters the system a table lookup is performed to see how the task should be planned. A centralized system utilizes a single machine to collect information and to perform decision-making. In a decentralized system, decisions are made at the processor level [5].

II. BPEL

The concept of orchestration is embedded on existing infrastructures for application amalgamation, classically used to automate business processes and integrate a variety of legacy components in these systems, a centrally-controlled set of workflow logics is developed to assist interoperability among applications. A common implementation of orchestration is a central engine interfacing multiple external participants: this solution makes it possible to merge large business processes without re-developing the solutions that originally automated the individual processes, thus making maintenance easier.

The Web Services Business Process Execution Language (WS-BPEL), also known as BPEL, is a primary industry specification that standardizes orchestration in the context of Web Services. The BPEL Orchestration leverages the intrinsic interoperability provided by Web Services, conceiving orchestration itself as a service, specified in terms of a high level language and implemented through an engine [6].

More in detail, BPEL is an XML-based specification language for describing business processes, built on top of the WSDL language for describing the interface of Web Services. A Web Service interface is specified in terms of port types, operations, and messages—which, e.g. in an object-oriented setting, would roughly correspond to the interface types, the method names, and the method types, respectively. In the BPEL case, port types are lists of operations, which can be either one-way or request-response—depending on whether they are asynchronous or not. The content of a message is an XML data record.

On top of a WSDL document describing the above "boundary" aspects, a BPEL specification provides information on the internal orchestration process of the Web Service. More precisely, a BPEL specification is composed of four declaration parts: the partner link types, the variables, the correlation sets, and the activity.

Partner link types—it define the feasible categories of partner links, which are abstract references to the Web Services orchestrated by the engine: a business process can access other services only through a partner link, which is bound to an actual Web Service address either at deployment-time or dynamically at run-time. In this way, it is easy to articulate dynamic interconnecting structures—a feature particularly suited to scenarios where e.g. pools of Web Services are dynamically bound/unbound to a business process in an orthogonal way, based on load-balancing policies.

Variables—BPEL also defines variables, which can carry XML data values and messages, and are used to support the stateful character of orchestration processes. In particular, such variables store the content of sent and received messages, the results of partial computations, and any other information required during orchestration.

Correlation sets — the global task of an orchestration process is divided into different stateful sessions called method instances, each holding its own information about the conversation, stored in suitable variables. The survival of different process instances raises the problem of correctly routing the incoming messages to the proper occurrence, and of providing an uniqueness to each process instance in a declarative way. This is achieved by the mechanism of correlation sets.

Correlation sets are sets of late-bound constants called properties, which store sorts of session identifiers: each process instance is uniquely identified within the complete orchestration process by the values assumed by such properties. The correlation method is based on the proposal of aliasing a property with one (or more) part(s) of a message to be sent or received. The value of the property is guaranteed to equal the actual content of the message: for instance, when an incoming message contains an alias to a property p, its content c is checked and used to dispatch the message to the proper process instance—namely, the one having c as its value for the property p.

Activities—Activities illustrate the behavior of the business processes, and are generally built by composing basic activities into structured activities. Basic activities comprise the acts of transfer and getting requests and replies (invoke, receive, reply), variable assignments (assign), synchronizations of interior concurrent activities through private links (source and target),

waiting for a timeout (wait), and terminating the process occurrence (terminate). Structured activities recognize sequential compositions (sequence), protected choices (pick), parallel compositions (flow), iteration cycles (while), and multiple cases (switch).

In order to balance with the complexity of specifications, and grant an encapsulation generalization for different event kinds, the activity of an orchestration process can be split in different ways. First, a basic activity can take the structure of a scope, that is, a separately-defined sub process with its own main movement, variables and correlation sets: this mechanism recursively allows for dividing a process into different modules.

BPEL specification can also define fault handlers, i.e. sub processes similar to scopes, triggered either by an explicit throw statement, or automatically when an activity fails. Analogously, compensation handlers (triggered by the recompense statement) support the long-running transaction attribute [7], while occurrence handlers are executed when exception messages are received or timeouts take place.

III. RELATED WORK

Lanzhou et al. (2009) intended the job scheduling algorithm for the large-scale parallel applications based on job pool in the varied environment. With the help of this algorithm, the resources can be utilized according to their potentials and the load balancing of resources may be achieved [8].

Mosincat et al. (2011) presented the novel scheduling algorithm for Cloud-based workflow applications. Their implementation was based on BPEL, an industry standard for workflow modeling, which did not require any changes to the standard. It is based on, but not limited to, the ActiveBPEL engine and Amazon's Elastic Compute Cloud. To automatically adapt the scheduling decisions to network-related changes, the data transmission speed between the available resources is monitored continuously. Experimental results for a real-life workflow from a medical domain indicate that both the workflow execution times and the corresponding costs can be reduced significantly [9].

Juhnke et al. (2010) presented the approach to allocate BPEL workflow steps to accessible resources. The approach took the data dependencies between workflow steps and the utilization of resources at runtime into account. This type of the developed scheduling algorithm simulated the result of the makespan of workflows whethere could be reduced by providing additional resources from a Cloud infrastructure. If yes, Cloud resources were automatically set up and used to increase throughput. The proposed approach does not required any changes to the BPEL standard. An implementation based on the ActiveBPEL engine and Amazon's Elastic Compute Cloud was presented. Experimental results for a real-life workflow from a medical application indicate that workflow execution times can be reduced significantly [10].

Srinivasan et al. (2013) proposed the orchestration engine for the supply chain system. The concept of Business Process Execution Language (BPEL) was being utilized to design the orchestration engine for the supply chain management system. The Eclipse BPEL designer was being used to build this engine for the supply chain management systems. The logic of the orchestration engine was being coded into the Java language. The orchestration engine was accomplished for resolving the challenges faced in the supply chain system competently. It was very competent in controlling the bullwhip effect occurred in the SCM [11].

IV. BPEL JOB SCHEDULING

To comprehend scheduling process, we will acquire the sample business circumstances in the business applications. The situation consists the process which is required to poll for a file daily between 6 AM to 6:30 AM. For this purpose we require to form a BPEL process in such way which can poll for particular file at some location of server by using file adaptor configuration wizard.

It may be created for polling for exacting file using File Adaptor Inbound Service and write the file information into database table. This process contains

- receive activity for obtaining payload from file adaptor,
- Invoke activity for writing payload to database and
- Transform activity to map source data and destination data.

This BPEL process will be reserved on polling for particular file each time as per polling frequency defined in configuring file adaptor wizard at design time. For polling the file at specific time like 6 AM to 6:30 AM, it is needed to add below piece of code in schedulebpel.xml file of the BPEL process.

```
<activationAgents>
   <activationAgent
   className="oracle.tip.adapter.fw.agent.jca.JCAActiv
   ationAgent"
   partnerLink="readFileService"
   heartBeatInterval="10">
  <property
  name="schedulerCallout">DefaultSchedulerCalloutI
  mpl</property>
  <property name="endpointScheduleOn">0 0 06 * *
?</property>
  <property name="endpointScheduleOff">0 30 06 * *
?</property>
   </activationAgent>
</activationAgents>
```

Activation Agent is one which generates the endpoint of adaptors active. There exists the input for adaptor endpoints to establish the process. In the discussed example readFileService partner link is lively end point of BPEL process. The activation agent called "heartbeat" that does scheduling action. The heartBeatInterval is calculated in seconds which specifies the frequently the schedule checked.

Quartz is implemented as part of java class named DefaultSchedulerCalloutImpl. The quartz scheduler turns heartbeat on and off. The subsequently chattels in scheduler code is endpointScheduleOn and endpointScheduleOff. from side to side these properties we will situate the scheduling time for polling mechanism. The attribute for endpointScheduleOn and endpointScheduleoff element is cron sequence. The Cron sequence is unix utility that permit tasks to be repeatedly run in the background at regular intervals.

This cron Sequence follows the specified set of syntax. In the above code cron sequence is as follows.

<property name="endpointScheduleOn">
*0 0 06 * * ?</property>*
<property name="endpointScheduleOff">
*0 30 06 * * ?</property>*

Here 0 0 19 * * ? and 0 30 19 * * ? specify that process should fire everyday between 6 AM and 6:30 AM. The Cron sequence is a string included of 6 or 7 fields separated by whitespace. The 7th field is optional.

V. CONCLUSION

There are several job scheduling solutions. But it can also be done within BPEL. The benefit of this approach is that you can trigger BPEL or ESB services or other Web services without doing any programming. It is all started from the BPEL Console, where the definite instances can also be terminated. And you can make it as standard as you want and schedule as many instances as you want. So you can have a daily, weekly and/or a monthly scheduled instance the same time. As a database programmer at heart, I think the quickest way to get a BPEL process started according to a potentially complex schedule with minimum coding effort would be using a database job (Schedule) that updates a database record in conjunction with a BPEL process that polls the database.

The huge difficulty, separately from the insignificant transparency of recurrently polling the database by the BPEL, is the split of process logic between the database and the BPEL PM. For uncomplicated schedule, it uses the alternative of creating a BPEL Process Dispatcher process with the WHILE-WAIT-INVOKE logic.

REFERENCES

[1] Wooldridge, M., and Jennings, N.R. “Intelligent agents: theory and practice”. The Knowledge Engineering Review, 10, 2, 115-152. 2002.

[2] FIPA. “The foundation for intelligent physical agents”. URL: http://www.fipa.org, 2003.

[3] Katia P. Sycara, “Multiagent Systems”, AI magazine Volume 19, No.2 Intelligent Agents Summer, pp. 79-92, 1998.

[4] P. Charlton and E. Mamdani, “A Developer's Perspective on Multi-agent System Design”, Multi-Agent System Engineering, Lecture Notes in Computer Science Volume 1647, pp 41-51, 1999.

[5] Peter Brucker, Scheduling Algorithms Fifth Edition, Springer ISBN 978-3-540-69515-8 Springer Berlin Heidelberg New York, 2005.

[6] BEA, IBM, Microsoft, SAP and Siebel, “Business Process Execution Language for Web Services Version 1.1”, S. Thatte, et al., May 2003. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[7] S. A. White. Business Process Modeling Notation. Specification, BPMI.org, 2004

[8] Cancan Liu, Weimin Zhang, "BPEL-Based Workflow Management and Parallel Job Scheduling in Ensemble Prediction," gcc, pp.409-414, 2009 Eighth International Conference on Grid and Cooperative Computing, 2009.

[9] Juhnke, E.; Dornemann, T.; "Multi-objective Scheduling of BPEL Workflows in Geographically Distributed Clouds," Cloud Computing (CLOUD), 2011 IEEE International Conference on , vol., no., pp.412,419, 4-9 July 2011

[10] Do□rnemann, T.; Juhnke, E.; "Data Flow Driven Scheduling of BPEL Workflows Using Cloud Resources," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , vol., no., pp.196,203, 5-10 July 2010

[11] S. Dalal and Dr. S. Srinivasan, “Constructing an Orchestration Engine for Supply Chain Management system”, International Journal of Computer Science and Management Research, ISSN: 2278–733X, Vol. 2, Issue 6, pp. 2709–2715 July 2013.

[12] Chern Han Yong and Risto Miikkulainen, “Cooperative Coevolution Of Multi-Agent Systems”, Technical Report AI07-338, Department of Computer Sciences, The University of Texas at Austin, pp. 1-15, 2001.

[13] Federico Bergenti and Agostino Poggi, “Middleware and Programming Support for Agent Systems”, in Proceedings of the 2nd International Symposium from Agent Theory to Agent Implementation, Vienna (A), 2002, pp. 617-622.

[14] Franco Zambonelli, “Developing Multiagent Systems: The Gaia Methodology”, ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3, Pages 317–370, July 2003.

[15] Yan Li and Xi-Zhao Wang, “An on-line multi-CBR agent dispatching algorithm”, Soft Computing , Vol. 11, pp. 391–395, 2007.

[16] Matteo Baldoni and Cristina Baroglio, “Agents, Multi-Agent Systems and Declarative Programming: What, When, Where, Why, Who, How?” Lecture Notes in Computer Science Volume 6125, pp 204-230, 2010.

[17] N Viswanadham and S Kameshwaran, “Orchestrating a Network of Activities in the Value Chain” in proceeding of 5th Annual IEEE Conference on Automation Science and Engineering Bangalore, India, August 22-25, 2009, pp. 501-506.

[18] Tony Andrews, Specification: Business Process Execution Language for Web Services version 1.1. http://www.106.ibm.com/developerworks/webservices/library/ws-bpel/, 2003

S. Dalal and Dr. V. Athavale, “Approach of Multi-agent system in Timetable scheduling problem using case based reasoning: in proceeding of DST & CSI Sponsored International Conference on Issues and Challenges in Networking, Intelligence and Computing Technologies ICNICT-2011 ISBN: 978-93-81126-27-1 pp 401-406 organized by KIET Ghazibad (U.P.) dated 2-3rd Sep 2011.