A Reliable Framework for Adaptive Scientific Workflow Management Systems Based on SOA

Alireza Azimi*, Saeed Parsa**

*Department of Computer Engineering, Islamic Azad University, Sari Branch, Sari, Iran
**Department of Computer Engineering, Iran University of Science and Technology, Narmak, Iran
azimi@iausari.ac.ir, parsa@just.ac.ir

Abstract— in this paper a reliable framework for scientific workflow management system is presented. The reliability of workflow management systems could be guaranteed by supplying workflow engines with fault control components. A fault control component may receive fault notifications through exceptions raised by web service handlers. Receiving a fault exception, the program execution state has to be rolled back to the state before the fault. To achieve this, the execution state of each component could be kept in a file, separately. Using the log kept in this file, the program status could be returned to the states before invoking the fault web service. The program execution is continued after the fault web service is substituted with a correct one. The applicability of the proposed framework is practically evaluated by applying it to design three different scientific workflow engines.

Keywords— grid Service; QOS; reliable scientific workflow engine; SOA; web service;

I. INTRODUCTION

In this article a new framework to improve the reliability of scientific workflow management systems is presented. A scientific workflow management system (SWFMS) is a system that supports the specification, modification, execution, failure-handling, and monitoring of a scientific workflow using the workflow logic to control the order of executing workflow tasks [21]. This paper offers a new framework to support the security, composability, transactability, modifiability and usability in workflows as composite web services. The resultant composite web service advocates reliability by automatically substituting fault web services with correct ones.

Web services are defined as self-contained, modular units of application logic which provide business functionality to other applications via an Internet connection [30]. There are a few frameworks proposed for implementing scientific workflow engines [1], [7], [30], [36]. To enhance reliability, the design and implementation of a new framework for implementing scientific workflow management systems as composite substitutable web services is presented in this paper. The proposed framework provides a component to determine appropriate web services dependent on the user requests. The user requests may include quantitative quality features of the desired services. Considering the reliability, interoperability and compatibility of the available web services, the system automatically selects the appropriate web services to serve the user.

Reliability is the probability of failure-free operation for a specified period of time in a specified application [20]. Within our suggested framework a component called WS-QOS management supports the reliability by monitoring the execution of workflows. As soon as a fault is detected in the execution of a workflow activity, the faulty activity is replaced with an appropriate web service.

Interoperability is the ability of a service to work with the other services without special effort across the internet [33]. To provide interoperability, in the framework proposed in this paper, a component, called workflow management, keeps records of composable services and their cooperation protocols.

Compatibility of web services states the fitness of services that interact with each other and closely relates to substitutability of service peers [4]. Compatibility is concerned with both the static features and dynamic behaviors of service peers. In the proposed framework, the coordinator component describing dynamic behaviors of web services is used.

The remaining parts of this article are organized as follows: The related works are described in Section II. Section III presents a new Reliable framework for scientific workflow management systems. The proposed framework is evaluated both in practice and theory in Section IV. The conclusions and future works are presented in Section V.

II. RELATED WORK

In this section the problems concerning to existing workflow management systems are described. Workflow management systems such as Condor-G [22], [6] Taverna [35] and GridAnt [25], [41], UNICORE [2] could detect faults at workflow and task levels. These systems provide the users with a history of the faults detected at runtime. Automatic recovery from faults at workflow level is performed by Askalon [33], Triana [17], [18], Pegasus [28] and GWEE [10] workflow management systems. A few number of known workflow management systems such as Chemomentum [3],
Escogitare [6] and Triana [18] could detect runtime faults at task level. Job-related faults such as deadlock, livelock, memory leak, uncaught exceptions, missing shared libraries and incorrect output results are examples of faults at task level. There have been no provisions for automatic recovering from such faults because workflow systems have been mostly concerned with the flow of data and control between the task rather than the tasks contexts.

A major task of a workflow management system is to determine the most appropriate location for executing jobs assigned to the workflow manager. However it is observed that even GridAnt [25] and Triana [17], [18] which are two well-known workflow management systems ask the user to define the exact location of the jobs. Askalon [34] and Condor-G [22] determine the most appropriate location for executing jobs by referring to the information which is frequently collected about interconnections and volume of data to be transferred between the jobs and their activities. However, to determine the exact location of activities at runtime semantics descriptions of the activities are required. Semantics description of the activities implemented as composite web services could be simply attained via the UDDI descriptions. These descriptions could be applied to locate composite web services.

In our proposed framework, descriptions of each composite web service, including a unique identifier, sequence of participating web services, the interconnections with other composite web services, are dynamically recorded and kept in a common data structure. The information collected in this data structure could be further applied to locate faulty web services at runtime. Runtime errors are detected by a component, Fault-Analyzer, in our proposed framework. The Fault-Analyzer component applies exception handling mechanism to detect predicted error prone conditions. Predefined abnormal behaviors and expected execution time could be applied as a basis for detecting faulty web services. Considering the execution log of a faulty web service it is possible to roll-back the program execution to its state before the execution of the faulty web service. The faulty web service could be replaced with a correct one. Fault detection and recovery has been only applied at hardware an operating system levels [25].

III. THE PROPOSED FRAMEWORK

The workflow management system proposed in this section complies with a known model called workflow coalition [37], [38], [39]. The main idea behind the design of this system has been to provide an engine capable of managing workflows on heterogeneous platforms with high reliability and the possibility for automatic fault detection and recovery. The proposed framework also supports dynamic replacement of existing workflow activities with modified versions of the activities. Implementing the workflow activities as loosely coupled web services it has been possible to dynamically replace existing web services with the ones with higher quality. To achieve this, a quality of service component, QOS, is considered in the proposed framework. The QOS component is described in section A. Dynamic replacement of the workflow activities may be also required when recovering from runtime faults. After a runtime error is detected, the faulty task could be replaced with a correct one. Fault detection is carried on by a fault monitoring component called Fault-analyzer. The Fault-analyzer component is a service provided by the WS-QOS Management component, described in section B. A distinct component called WorkFlowDef provides the users with an environment to define and modify workflows. The WorkFlowDef component is described in section C. User requests are handled by a component called WS-Enquires. The WS-Enquires component is described in section D. The appropriate services to handle the user’s enquiries are selected by the SearchAndRecovery component, described in section E. New instances of the selected services are created by the WS-Instance component described in section F. All the data and the messages to be passed between the workflow activities are kept by the WS-Library component are described in section G. A log of activities participating in each job executing by the workflow engine is kept by the Context-Management component of the WS-Composition management component described in section H.

The overall framework of the proposed workflow management system is presented in Figure 1. The framework components are further described in the following sub sections.

A. The QOS Component

The QOS component provides an interface to search the internet for the web services similar to the ones applied in the workflow system. The component uses the UDDI registries across the internet to find qualified services considering the services qualities defined by the users. After the user approval, the qualified component is replaced with its corresponding component in a library called WS-Library. The component is also replaced with it corresponding component within the workflow engine.

B. The WS-QOS Management Component

The proposed framework supports reliability through the WS-QOS Management component. As shown in Figure 1 this component comprises four sub components named Fault-Controller, WS-Recovery, Fault-Analyzer and QOS component. The Fault-Analyzer component receives a notification from the workflow program exception handler whenever any predefined faulty behavior is detected. Predefined faulty behaviors are catch by the program exception handler. The Fault-Analyzer component also detects faults by controlling the execution time of the web services after a fault is detected. The Fault-Analyzer reports the detected fault to the Fault-Controller component. The Fault-Controller component after a fault is detected the WS-Recovery component replaces the detected faulty web service with correct one. Fault recovery is carried out by the WS-Recovery Component. This component uses the workflow program execution log to roll-back the execution of the current job to the state immediately before the start of the faulty
activity. All the information concerning the execution of the current web services are kept by the Context-Management component in a log file. The WS-Recovery component asks the SearchAndRecovery component to look for a web service similar to the faulty one. The SearchAndRecovery component gets the description of the faulty web service from a component called Fault-Controller component.

C. The WorkFlowDef Component
A new workflow could be specified through the environment provided by the WorkFlowDef component. WorkFlowDef component applies workflow definition and description tools such as workflow creator in .Net framework to define rules concerning workflows. These rules are further applied by the workflow engine to build specific workflows in accordance with the user requirements.

D. Service-Enquiry
The Service-Enquiry component receives high level descriptions of compound services from the users. The component breaks down the compound service requests into simple sub-requests. The resultant sub-requests are analyzed and used by the component to look for appropriate web services. The selected web services are further applied to build service orchestrations reply to the user requests.

In fact a workflow defines a set of activities which may be executed in any order dependent on the decisions made. Each activity within a workflow may be defined as a sub-workflow. Therefore, a complicated user’s enquiry may be defined as a workflow where each activity itself may be defined as a sub-workflow.

E. The SearchAndRecovery Component
After a faulty web service is detected, the SearchAndRecovery component is invoked to look for corresponding web service across the internet. The component collects information describing the faulty web service from a table handled by the Context-Management component. The Context-Management component fills this table through user enquiries.

F. WS-Instance
After a user enquiry is interpreted and analyzed by the Service-Enquiry component, new instance of the web services responding to the user enquiry are built by the WS-Instance component. These instances are applied by the workflow engine to run the workflow in accordance with the user enquiry. The users could follow the requests through these instances.

G. The WS-Library Component
The WS-Library keeps a record of each workflow and each activity or web service being executed with in the workflow management system. After a workflow is terminated all its records are wiped off. These records are applied by the context-management component to roll-back the faulty component to safe states.

H. WS-Composition Management
WS-Composition management component is a special description contained list of participating services in composition, their orders, and their connection method, coming and going services between them for making composition services. The component is a standard mechanism for services to distribute and record of content descriptions which presented for activities. It formulated from 3 parts of WS-Handler component, context management component and selecting protocols. WS-Handler component has the duty of making relation between web services and connection with other framework components. Context management component makes a common field for partial services of a composition that contained a place for maintaining common information field between services and state maintenance of composition services.

IV. EVALUATION
In this section, evaluation of workflow engine is presented. To this end, in the framework proposed in this paper, Condor (Condor-G) workflow engine [22] and MyGWFMiddWare workflow engine [15] done previously at this faculty, have been selected, and sample workflow will be executed on these three ones. Besides, in order to compare these three workflow engines accurately, a reassessment has been done on the
proposed framework based on the previous comparison evaluation.

One of the useful and appealing items of statistics is data number clustering. For example, let us cluster the below data based on numbers 3, 10 and 14. Three machines A, B and C compute the square subtracting of input data 3, 10 and 14 successively and write in files A.Out, B.Out and C.Out. Figure 2 is shown the entrance file and the internal files of clustering. Machine M evaluates input file (In.Dat) and produces the clustered data files (Out.Dat) as follow. Machine M reads the first file line of In.Dat and calculates the supposed data of subtracting absolute value (no.6) C.Out, (no.16) B.Out, (no.9) A.Out. The three result numbers are compared and the lowest one indicates the belonging of no.6 to that cluster. Thus, no.6 belongs to cluster no.3.

The operating environment considered for this purpose has been chosen for 7 computers connected to the internet with different IP addresses in one local network with MS Windows 2003 operating system and Fedora Core 4, successively for the job sending to proposed workflow engine and Condor. The connective protocol of machines is TCP/IP for both workflow engines and both of them take advantage of the Intel processors with computed speeds and Pentium 4 category.

<table>
<thead>
<tr>
<th>Clustering Number</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>232141.00</td>
</tr>
<tr>
<td>3</td>
<td>239675.00</td>
</tr>
<tr>
<td>4</td>
<td>243105.00</td>
</tr>
<tr>
<td>5</td>
<td>246257.00</td>
</tr>
<tr>
<td>6</td>
<td>267875.00</td>
</tr>
<tr>
<td>7</td>
<td>275355.00</td>
</tr>
</tbody>
</table>

Table 1. The result of clustering using the proposed framework.

As it was raised earlier 7 machines were considered to execute the workflow. Because in the first timed process of workflow the proposed model has the potentiality of choosing suitable machines for executing the workflow, the time limit for the proposed engine is far less in comparison with Condor. This condition, gradually, with the increase of clusters, and because of the busy machines has changed so that the time of executing workflow in clustering 5, 6, and 7 by far increased and has approached to the Condor time.

The reason for the appearance of this phenomenon can be due to the lack of ability of scheduler in choosing the right machines. Because with the passage of time when the number of clusters go up, the machines are devoted to the workflow; hence, the target of the scheduler for choosing the right machine becomes smaller and smaller. For example, in order to choose the sixth machine for computing the sixth cluster, only two machines, to assign the workflow, are at the disposal of the scheduler. As a result with the increase of the machines we can assign scheduler to the society of a more varied target to choose more suitable resources.

V. CONCLUSION and FUTURE WORK

The aim of this paper is to present scientific workflow management system on the basis of the web service over the grid network. Running the workflow and the inner tasks in the proposed model, through the published web services exists in the grid network machines. In this paper, the problems workflow management systems have been considered and in
continuation a certain solution in the form a proposed framework with exploiting the service-oriented architecture (SOA) was presented. Also, it was explained that how the proposed framework is harmonized with workflow management systems. In this line, a definite infrastructure is defined for managing workflow which is in connection with Globus standards, W3c and workflow coalition. In the end a workflow execution language (WFEL) for GT3 (Globus-Toolkit 3) environment was presented. In line with the continuation the carried out activities one can identify the current impediment to activate the workflow management systems and with regard to the suitable mechanism you can reduce and finally remove them. The advent of this affair can enormously reduce the worried of the users connected to the grid. Among the other activities ahead one can point to the improvement of the proposed framework; can better the fault tolerance in the time of executing workflow, and with the increase of the tag range of workflow execution language (WFEL) you can improve the quality of service (Qos) presented. In addition, overflow of jobs, the expenses of the exploiting the resources and the scalability of the scheduler at the runtime can be taken into consideration.

REFERENCES


Saida Ziane and Hichem Bacha, “AROUND QUALITY OF SERVICE FOR BUSINESS WEB SERVICES: FROM AVAILABILITY AND SECURITY TO MOBILITY”; IADIS International Conference e-Commerce 2006.


