An innovative workflow mapping mechanism for Grids in the frame of Quality of Service

Dimosthenis Kyriazis*, Konstantinos Tserpes, Andreas Menychtas, Antonis Litke, Theodora Varvarigou

Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str, GR-15773 Athens, Greece

Received 31 March 2007; received in revised form 2 July 2007; accepted 14 July 2007
Available online 27 July 2007

Abstract

The advent of Grid environments made feasible the solution of computational intensive problems in a reliable and cost-effective way. As workflow systems carry out more complex and mission-critical applications, Quality of Service (QoS) analysis serves to ensure that each application meets user requirements. In that frame, we present a novel algorithm which allows the mapping of workflow processes to Grid provided services assuring at the same time end-to-end provision of QoS based on user-defined parameters and preferences. We also demonstrate the operation of the implemented algorithm and evaluate its effectiveness using a Grid scenario, based on a 3D image rendering application.

Keywords: Grid computing; Quality of Service; Workflow management

1. Introduction

Grid computing [1] is increasingly considered as an infrastructure able to provide distributed and heterogeneous resources in order to deliver computational power to resource demanding applications in a transparent way [2,3]. Built on pervasive internet standards, Grids allow organizations to share computing and information resources across department and organizational boundaries in a secure and highly efficient manner. Grids support the sharing, interconnection and use of diverse resources, integrated in the framework of a dynamic computing system.

Managing the application workflow operations within Grid environments requires the orchestration of the aforementioned distributed resources [4]. In that frame, workflow is an important factor for application composition on Grids [5] promoting inter-organizational collaborations by integrating the teams involved in managing of different parts of a workflow. Besides, the literature [5] describes additional advantages of workflow management such as the utilization of resources to increase throughput or reduce execution costs and the ability to build dynamic applications which orchestrate these resources.

Since workflow is a wide concept in technology, the terminology regarding workflows that is used afterwards in this paper is defined. Workflow Management Coalition (WfMC) provides the following general definition [6]: “Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”.

In Grid environments, workflow can be defined as the orchestration of a set of activities to accomplish a complicated goal including application processes, business processes, and infrastructure processes [7]. Workflow is an architecturally important factor for dynamic interoperability and adaptation to different business models, which can be addressed as workflow policies, and deployment contexts.

A Workflow Model/Specification is used to define a workflow both in task and structure level. There are two types of workflows, namely Abstract and Concrete [8,9] while concrete workflows are also referred to as executable workflows in some work [10]. In an abstract model, the tasks are described in

* Corresponding author. Tel.: +30 6973319000.
E-mail addresses: dkyr@telecom.ntua.gr (D. Kyriazis), tserpes@telecom.ntua.gr (K. Tserpes), a_menychtas@telecom.ntua.gr (A. Menychtas), ali@telecom.ntua.gr (A. Litke), dora@telecom.ntua.gr (T. Varvarigou).

0167-739X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2007.07.009
Fig. 1. Workflow definitions (Application, Abstract, Concrete Workflow).

an abstract form without referring to specific Grid resources for task execution since it provides the ability to the users to define workflows in a flexible way, isolating execution details. Furthermore, an abstract model provides only service semantic information on how the workflow has been composed and therefore the sharing of workflow descriptions between Grid users is feasible, which is of major importance for the participants of Virtual Organizations (VOs) [2]. Abstract workflow can be composed with systems like the one presented in [11].

In the concrete model, the tasks of the workflow are bound to specific resources and therefore this model provides service semantic and execution information on how the workflow has been composed both for the service instances and for the overall composition (e.g. dataflow bindings, control flow structures). In correspondence with the abstract model and the relationship to the VO classes, the tasks included in a concrete model may also refer to data movement requests in order to publish newly derived data into VO [9]. There has to be mentioned that the service instances do not necessarily correspond to resources since within a resource more than one service instances may be available and executable.

In Fig. 1 we present the aforementioned workflow definitions.

Following the workflow models’ definitions, tasks in an abstract model are portable and can be mapped onto any suitable Grid services by using suitable discovery and mapping mechanisms. To this direction, our topic of research is a novel Workflow Mapping Mechanism, the outcome of which is a Concrete Workflow. The concrete workflow can be seen as a “path”, a selection of service instances from service types on specific order so as to achieve the goal of executing a given application workflow.

The major objective of the work presented in our paper is to identify and describe the process that needs to be implemented in order to define the concrete workflow given an application workflow and the essential Quality of Service (QoS) parameters. This research topic is referred as Workflow QoS Constraints, and remains one of the key factors in a

Grid Workflow Management System and more specific in the Workflow Design element [12].

A workflow mapping mechanism is an integral part of the QoS provisioning, and especially the end-to-end Quality of Service, since this is the only way to estimate, calculate and conclude to the mapping of workflows and the selection of the available service types and instances in order to deliver an overall quality of service across a federation of providers. This mechanism has been studied and evaluated through experiments and is presented in detail further on.

However, supporting end-to-end QoS with workflow mapping mechanisms requires the pursuit of a number of aspects that need to be addressed. In order to tackle each one of them in this paper, we will start from the examination of the QoS requirements and parameters that need to be fulfilled in order to enable the dynamic mapping of workflows to Grid services. Furthermore, this set of parameters are categorized based on their source (for example there can be application parameters or user-defined parameters) and consist as input for the workflow mapping mechanism. For each service instance, these parameters are taken into account within the mapping process. Based on the above, the provision of workflow mapping capabilities based on QoS metrics is of major importance since it allows the adoption of different business models by designing the workflow according to QoS metrics with a priori knowledge of the provided quality.

The remainder of the paper is structured as follows. Section 2 presents related work in the field of the selection processes and workflow mapping with regard to QoS. Section 3 introduces the workflow mapping mechanism while Section 4 presents a categorization of the QoS parameters that will be used. Use Cases analyzed based on user constraints and preferences are included thereafter in Section 5. The research topic and the focus of our work, which is the algorithm used within the workflow mapping mechanism, is addressed in Section 6. In the last section (Section 7), we present two (2) experiments which we conducted in order to demonstrate and evaluate the operation of the implemented algorithm. For the sake of the experiment, the infrastructure of our lab premises under GRIA middleware [13,14] was used for a real application scenario, the 3D image rendering. The performance of the proposed mechanism is depicted in the results and the evaluation section. Finally, Section 8 concludes with a discussion on future research and potentials for the current study.

2. Related work

There are various approaches for mechanisms that handle QoS parameters in order to optimize the selection process within a workflow. Generally, the way QoS is perceived to work in current Grid architectures is as part of the SLA negotiation process as tackled in [15–17]. The end-user’s constraints and preferences are parsed to several service providers through the functionality offered by a broker (usually the SLA Management Service) for finding the appropriate service providers. Aiming on assuring end-to-end provision of
QoS with the use of workflow mapping mechanisms, there are various approaches on how to handle this QoS information. The Globus Architecture for Reservation and Allocation (GARA) [18] addresses QoS at the level of facilitating and providing basic mechanisms for QoS support, namely resource configuration, discovery, selection, and allocation.

Furthermore, there are traditional selection and scheduling methods. Literatures [19,20] introduce OLB, UDA, Sufferage, GSA, Tabu and other algorithms with the essential target of obtaining the smallest makespan, which only consider the system performance, but they have neglected the user’s grade of service demand.

In [21,22] three algorithms are presented that use two parameters – cost and deadline time – in order to express quality of service dimensions. These parameters are used to implement a selection scheme, which refers to applications that consist of parametric processes that are independent of each other regarding their execution sequence. The first presented algorithm minimizes the cost, the second minimizes the cost with regard to the time and the third one is generic and is not trying to optimize one of the parameters but it assigns to the “cheapest” nodes a number of jobs which is reversely proportional with the time a node needs to complete a job.

Similar to the above, a scheduling algorithm taking into account two (2) major parameters (Cost and Time) is described in [23]. Four adaptive algorithms for deadline and budget constrained scheduling are incorporated. These algorithms are used for cost optimization — within time and budget constraints, for time optimization — within time and budget constraints, for conservative time optimization — within time and budget constraints and for cost-time optimization — within time and budget constraints.

Furthermore, a workflow QoS specification and methods to predict, analyze and monitor QoS are presented in [24]. The work is focused on the creation of QoS estimates and the QoS computation for specific metrics — time, cost, fidelity and reliability with the use of two methods: analysis and simulation. In this case, the parameters are handled one by one similar to [21,22] and not in a combined way while the overall estimation emerges from the individual tasks, which makes QoS provision unfeasible since the estimation should be done in a vice versa way in order to achieve the latter.

The literatures [25,26] propose some optimization scheduling algorithms under the limitation of time and cost in the Nimrod-G model. Nimrod-G is an economic or market based resource management and scheduling system for a Grid-enabled system [26]. It supports user-defined deadline and budget constraints for schedule optimisations and regulating demand and supply of resources in the Grid by leveraging services of GRACE (Grid Architecture for Computational Economy) resource trading services [27]. Nimrod-G [28] allows the users to lease and aggregate services of resources depending on their availability, capability, performance, cost, and users’ QoS requirements.

DAGMan [29] is a set of C libraries which allow the user to schedule programs based on dependencies. DAGMan is part of the Condor project and extends the Condor Job Scheduler to handle intrajob dependencies. As the name suggests, DAGMan represents a collection of job dependencies as a directed acyclic graph. It uses DAG as the data structure to represent job dependencies. In the same context, in [30] the authors present an algorithm that deals with the scheduling of workflow applications (represented by weighted DAGs) on a set of heterogeneous processors with different capabilities.

Other interesting works on selection algorithms and scheduling are presented in [31,32], where in the first one the authors present a heuristic scheduling of bag-of-tasks with QoS constraints, while the latter handles the problem of distributed job scheduling in Grids using multiple simultaneous requests. The estimation, monitor and control of QoS with workflow systems is also addressed in [33], in which an approach based on QWF-net is presented in order to associate tasks with exponential response time and exponential time to failure.

The mechanism we introduce in this paper advances the field of research in workflow mapping with regard to QoS since the approaches presented so far address the case of selecting services and nodes based on QoS parameters by dealing only with the specific cases of minimizing one of the parameters. In our study, these parameters are dealt in a combined way as well, while we also introduce the case on which the user sets preferences and therefore one of the parameters may play a higher role in the selection process (a weight attribute is attached to it). Finally, we present an embedded scheduling scheme within the algorithm that not only adds another criterion in the selection process — execution time, but also allows the contentment of the user’s execution time requirement.

3. Workflow mapping mechanism overview

As already mentioned, the aim of the work presented in this paper is to identify and describe the processes that need to be completed in order to map workflow processes to service instances with regard to the provided QoS metrics. To achieve the latter we introduce a selection algorithm along with its implementation. Each workflow contains processes – service types – that can be executed from a set of service instances (candidates), which are annotated with QoS information. The workflow mapping mechanism allows the selection of the service instances (candidates) for each service type based on the application workflow, the user constraints and preferences and the QoS parameters exposed from each service instance.

The input of the aforementioned mechanism is the following (as also presented in Fig. 2):

- Application workflow with “importance sorting” information, indicating which of the service types play a higher role within the workflow. This factor should be encompassed into the application workflow provided by the application developers (e.g. which service is of higher importance: movement of large amount of data or execution time) since it implies the execution order of the service types. The final result of this step is the construction of the abstract workflow, with a list of available service instances per service type.
- **QoS constraints and preferences set by the user.** A set of constraints (e.g. thresholds/restrictions to specific values of the parameters) along with possible user preferences, which attach a weight attribute to the QoS parameters, are also parsed into the workflow mapping mechanism.

- **QoS information for the Grid service instances.** Each service instance (candidate) of a service type is annotated with QoS information that will be used during the selection process. This information includes the QoS parameters’ values as published by the service providers.

Since the above information is parsed into the workflow mapping mechanism, the execution of the algorithm (described in Section 6 of this paper) is initiated in order to make the optimum selections and produce as a result the concrete workflow.

### 4. Initial QoS parameters

As stated in the workflow mapping mechanism overview (Section 3), QoS parameters are taken into account for the definition of the concrete workflow and are prerequisites for the achievement of end-to-end QoS provisioning. In this section, we present the parameters that have been used in our study and are considered as Initial QoS Parameters.

In the following paragraphs, a classification of the QoS parameters is stated as a direct consequence of the logical categorization of them, which means that parameters under the same category are sharing common properties. Currently, this work has concluded to the classification of QoS parameters in three (3) major categories:

- **User-defined parameters,** which relate to requirements/constraints that the user who initiates the workflow execution process would like to pose, such as cost restrictions (e.g. maximum overall cost).

- **Application parameters,** which relate to the offered QoS from the application perspective. For example, the application configuration could play a significant role to the availability of the task to be executed.

- **Resource parameters,** which relate to all types of resources, including computational, storage and network resources. For example, from that perspective the network infrastructure can be regarded as a set of interacting resources that are offering a specific QoS level.

The abovementioned parameters and their classification are shown in Fig. 3.

It is clear that there are numerous parameters that are of particular interest in Grid infrastructures. Our work has been focused initially on **Availability** (expressed as numerical percentage), **Cost** (expressed as numerical account units) and **Time** (expressed as numerical time units). These parameters consist as input on the presented algorithm and in the experiments described later on. Nevertheless, they are selected as representative indicative parameters for the overall workflow process and were selected in the first place after a detailed study of the QoS parameters that are included in the SLAs as terms upon the users and the service providers agree. Therefore, cost and execution/completion time are considered to be essential terms in such an agreement. Moreover, availability was chosen in order to include an additional criterion in the selection process that expresses the service providers’ ability to offer a service for its corresponding cost in combination with the published execution time (or deadline). The set of parameters can be changed or extended in order to include new ones; for example availability could be replaced with “characterization” to express the service providers’ reputation based on their past behavior (e.g. number of SLA violations).
5. Use cases based on user constraints and preferences

The users that initiate a workflow execution state their QoS requirements in two ways: “hard limits” which are expressed as constraints on the requirements; and preferences which are expressed as importance factors on the aforementioned requirements, adds an extra value to the workflow mapping process since the selection is also made considering these preferences. In order to make the latter feasible, a weight attribute is attached to the QoS parameters. Based on that and regarding as initial indicative parameters availability, cost and time, the following use cases are identified:

- **Thresholds.** In this case the user sets thresholds/restrictions to the values of the parameters. This means that the overall cost should not exceed a pre-defined budget, the availability should be at least at a specific level (expressed as a percentage value) and a deadline for time should be achieved. The thresholds may apply to one or more parameters at the same time.
- **Availability optimization.** In this case the user sets a high preference on the availability parameter of the services and therefore the selection is made considering as a major factor the availability parameter of them. The algorithm proceeds with the selection of the instances that are annotated with the highest availability values.
- **Cost optimization.** In this case the user sets a high preference on the cost parameter and based on that, cost is the major factor during the selection process. The output is a concrete workflow that achieves the lower overall cost.
- **Time optimization.** Similar to the availability and cost optimization, the selection of the service instances is made with regard to their execution time in order to achieve the lowest overall execution time for the workflow.
- **Optimum solution.** In this case the user doesn’t set a preference and as a result the weight attributes for the three (3) parameters are equal. The algorithm execution results to a concrete workflow where the service instances selected offer the optimum value of availability and execution time for their corresponding cost.

6. Algorithm description

In the following, we describe the algorithm used within the workflow mapping mechanism in order to select the service instances per service type based on the QoS parameters and the application workflow. The main goal of the algorithm is to result to an optimum selection with regard to the QoS metrics requested by the user and offered by the service providers. The algorithm’s strategy is initially to map workflow processes to service instances in a way that the constraints set by the user are met (select instances that meet the requested availability level without violating the budget constraint). Afterwards, the instances that offer higher level of QoS (in terms of availability and execution time) for the corresponding cost are defined and replacements on the initial mapping take place. Finally, a scheduling scheme is applied either if the user has stated time optimization preference (as described in Section 5) or so as to meet the time constraint (deadline) set by the user. The embedded scheduling scheme considers as an acknowledgment that each time a service is split, the corresponding execution time and cost would be halved. However, experiments and simulations have shown that the time and cost are not halved in whole but a penalty is added (like a fixed charge). The scheduling scheme is designed to take into account this penalty as well.

Within the algorithm, the user’s preferences are expressed as slope values for availability, cost and time parameters. These values enact how important each parameter is considered to be by the user and each one may get values $0 \leq \text{Slope} \leq 1$. The higher the slope is, the more important the value is for the user. Decimal values in slope allow the detailed and exact input on user preferences. Consequently:

$$\text{AvailabilitySlope} + \text{CostSlope} + \text{TimeSlope} \leq 3. \quad (1)$$

In the following, we describe in detail the major steps of the algorithm along with their sub-steps.

**Step 1: Calculation of auxiliary values (pilots for the algorithm completion)** in order to define metrics that “characterise” the level of QoS provided by the service instances

(i) Calculation of the minimum and maximum values of availability, cost and time for each service type based on their service instances (candidates).

(ii) Computation of the pilot values for availability, cost and time parameters based on the minimum and maximum values of them with the use of the functions given in Box I.

In the above functions, $x$ is the value of availability, cost and time for each service instance and $\text{MinAvailabilityValue}$, $\text{MaxAvailabilityValue}$, $\text{MinCostValue}$, $\text{MaxCostValue}$, $\text{MinTimeValue}$ and $\text{MaxTimeValue}$ are the minimum and maximum values of the parameters (as described in the previous sub-step).

These functions, which came as a result of our study of various experiments and simulations, are nonlinear with positive slope and they enact the influence of any change in the availability, cost and time values to the user defined parameters and preferences.

(iii) Calculation of the new parameters’ values that will be used further on based on the aforementioned functions (with the use of equations given in Box I):

$$\text{NewCostValue} = \text{InitialCostValue} \times F_{\text{Cost}}(\text{InitialCostValue}) \quad (2)$$

$$\text{NewAvailabilityValue} = \text{InitialAvailabilityValue} \times F_{\text{Availability}}(\text{InitialAvailabilityValue}) \quad (3)$$

$$\text{NewTimeValue} = \text{InitialTimeValue} \times F_{\text{Time}}(\text{InitialTimeValue}) \quad (4)$$

In the above equations, $\text{InitialAvailabilityValue}$, $\text{InitialCostValue}$ and $\text{InitialTimeValue}$ are the values of the parameters that were initially obtained by the service instances as their offers for the job execution.
Calculation of the following $\text{ConvertedIndex}$ that will be used in the following in order to proceed with the selections:

\[
\text{ConvertedIndex} = \frac{\text{NewCostValue} \times \text{NewTimeValue}}{\text{NewAvailabilityValue}}. \quad (5)
\]

This index is the major criterion during the selection process since it shows for each service instance the offered level of availability with regard to the corresponding cost and execution time. The lower the index is, the higher level of quality is provided by the candidate since for a job execution it demands lower budget and offers lower execution time for a higher availability.

**Step 2: Initial workflow mapping** with service instances that meet the user’s requirement for availability without violating the cost constraint.

(i) For each service type, a service instance is selected that meets the user’s availability constraint with the lowest corresponding cost.

(ii) The overall workflow cost is calculated for the instances selected in the previous sub-step. If it exceeds the user’s cost constraint, the workflow cannot be mapped into service instances based on the user’s requests and the algorithm ends. Otherwise it continues with the following step.

**Step 3: Definition of a service instance (candidate) for each service type.** The reason of this step is to discover the candidates that provide higher level of QoS for each service type within a workflow.

(i) For each service type, the service instance with the lowest value of the $\text{ConvertedIndex}$ (Eq. (5)) is defined in comparison with the one selected in Step 2 of the algorithm. If no instances are defined, the service type is excluded from the rest of the algorithm execution since no optimization can be performed. If this applies for all service types of the workflow, the algorithm ends and the initial workflow mapping is considered to be the final one. Otherwise it continues with the following sub-step.

(ii) Selection of the service instances (candidates) for each service type with the lowest value of the $\text{ConvertedIndex}$ (Eq. (5)).

(iii) Calculation of the differences in the values of availability, cost and time between the initial service instance selection (from Step 2) and the replacement one (from the previous sub-step).

**Step 4: Creation of a list with the “best candidates” for each service type in order to find possible replacement(s)**

(i) For each difference that has been calculated in Step 3, the $\text{ConvertedIndex}$ (Eq. (5)) is re-calculated. Basically, Step 1 of the algorithm is re-executed considering as initial values for the service instances the aforementioned differences and the replacements are made based on their differences (e.g., it is preferable to spend 6 cost units in order to increase availability by 4% than spend 10 cost units in order to increase it by 5%).

(ii) The service instances with the lowest new $\text{ConvertedIndex}$ (Eq. (5)) are selected.

(iii) Based on the new selection of service instances, the overall workflow cost and time values are re-calculated. If the user’s cost constraint is met, the new service instances are the ones defined in the previous sub-step, otherwise they are excluded from the available service instances. Regarding the time constraint (possible deadline set by the user); it is dealt with by Step 5 — scheduling scheme — that is described later on.

(iv) The algorithm is looped and continues from Step 3 for all service types.

**Step 5: Scheduling Scheme** that allows more than one service instance to be selected per service type in order to meet time constraints set by the user or if a time optimization preference is stated.

(i) Categorization of the service types based on their importance sorting (as described in Section 3), which results to a priority value for each service type. For each different priority, a new Category is defined in which all service types with the specified priority belong to, and due to that these service types can be executed in a parallel way. The Category is introduced since the execution time of the category in which a service type belongs to should be taken into account and not the individual completion time of it. In the following the time that is needed for the overall workflow to be executed is calculated.

(ii) Determine the most time critical service type, as the one for which the execution time of the pre-selected service instance is higher than of another service type. For this time critical service type, there should be at least one more service instance available since some of them may be excluded (if they do not meet the user’s availability and cost constraints). Moreover, if there is another time critical service type, the category in which it belongs to and the most time consuming service instance are stored. If there aren’t any time critical service types this sub-step is completed.

(iii) Determine the service instance for the aforementioned time critical service type that has the lowest execution time and was not already selected or disabled (as it may occur in the following sub-step). We name it Primary Candidate.

(iv) Check if the addition of the Primary Candidate increases or decreases the execution time of the time critical
service type. If it increases the time, then it is disabled and the scheduling scheme returns to sub-step (ii).

(v) Check if the addition of the Primary Candidate decreases the execution time of the service type so that it is lower than the execution time of another service type of the same category. For each service type that the latter occurs, the scheduling scheme continues with the next sub-step (sub-step (vi)), otherwise it continues with sub-step (vii).

(vi) For the other service types of the same category, we also find the corresponding service instances by applying sub-step (iii). In the following, we see if their addition increases or decreases the execution time of the service type to which they belong. If it increases the time, the service instance is disabled and we proceed with the next service type of the same category. If it decreases the time, then it belongs to the so called Secondary Candidates. This step is looped for the service types that belong to the same category with the Primary Candidate.

(vii) Calculation of the overall workflow cost after the addition of the Secondary Candidates. If the overall workflow cost doesn’t exceed the user’s cost constraint the additions are confirmed and the Secondary Candidates are also selected. Otherwise, the Primary Candidate is disabled and the scheduling scheme returns to sub-step (ii).

7. Experiments

In the above paragraphs, we described the algorithm used within the workflow mapping mechanism. This algorithm was implemented as a service and various experiments have been performed. In the following, we present implementation details of the aforementioned service, a testbed for a real application scenario (3D image rendering) and two (2) experiments along with the corresponding results in order to evaluate the algorithm functionality.

7.1. Implementation

The paper proceeds further, providing details of a service that implements the functionalities of the workflow mapping mechanism which is compliant to the Open Grid Services Architecture (OGSA) [32].

The scheme of this architecture is designed following the specifications under the Web Services Resource Framework (WSRF) [34] and therefore it is a federation of Web Services managing stateful resources. Furthermore, the way the service was implemented allows the use of the workflow mapping mechanism as a stand-alone module that can be used in order to achieve the optimum selection of service instances/candidates both from a given service type or from an overall workflow with regard to the QoS parameters obtained both from the service providers and from the user along with the user preferences.

For a detailed representation of the methods and the variables used within the service implementation of the algorithm please refer to the Appendix.

7.2. Testbed

For the purposes of our experiments we deployed a 3D image rendering application in a GRIA middleware environment [13]. GRIA is a service-based middleware, compatible to the OGSA, under active development which was implemented in the frame of the GRIA IST project. The infrastructure consisted of a number of various PCs including a cluster of 8 PCs with the specifications shown in Table 1.

The 3D Rendering Service is based on a distribution of the AIR renderer [35]. The GRIA Supplier that is deployed on the cluster offers the rendering service, by federating a number of intermediate services, such as collecting the data, checking the validity, invoking the rendering service and delivering the result. Additionally, the cluster offers the Shaders Compilation Service and Textures Compilation Service which are prerequisites for the rendering service. The rendering is achieved by using descriptors expressed by the RenderMan Interface Bytestream [36] (RIB) format which is a complete specification of the required interface between modelers and renderers. Specified renderers (such as AIR) are then able to reconstruct and rendered the image using the intermediate RIB files. The input of the whole process are the RIB files, the Shaders and the Textures while the output is the full rendered image as also illustrated in Fig. 4.

Therefore it is applicable to workflow management systems and engines and can be accessed either with the use of web services or with the implementation of an interface in order to obtain the concrete workflow from the provided QoS metrics and the abstract workflow.

Table 1

<table>
<thead>
<tr>
<th>Specifications of the computers comprising the cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td>×86 Intel P4 3 GHz</td>
</tr>
</tbody>
</table>

Fig. 4. 3D image rendering scenario.
7.3. Evaluation

The paper proceeds further, providing two (2) representative experiments, which have been completed in the frame of the aforementioned testbed, and the corresponding evaluation results. The experiments were performed for a given application workflow for the rendering case. In this application scenario, the QoS requirements mainly refer to the execution time since 3D rendering is a time consuming process and therefore the users’ requests focus on execution time with regard to the available budget. The workflow of the application consists of three (3) service types that are sorted based on importance; shaders and textures compilation service types are prerequisites for the rendering service type. Furthermore, since shaders and textures compilation services can be executed in a parallel way, they belong to the same category (Category 0 was introduced in Section 6). Each from the above service types includes eight (8) service instances – the nodes of the cluster – that were configured to provide different QoS parameters for the sake of the experiment both for an immediate concrete workflow execution and for an execution in a later time frame. The above are presented in Fig. 5.

In Table 2 we cite the obtained values for the QoS parameters (availability, cost and time) of all service instances.

Based on these values, two representative experiments will be described for the validation and the evaluation of the algorithm implemented within the workflow mapping mechanism service.

7.4. Experiment 1

The user defines via the GUI the following constraints/restrictions for the QoS parameters: (i) Service Instances’ Availability: At least 80%, (ii) Overall Cost: Threshold to 330 account units, (iii) Overall Execution Time: 200 time units. Furthermore, in this first experiment, the user does not define a major factor (preference) and thus: Cost Slope = Time Slope = Availability Slope = 1. This demonstrates the Optimum Selection use case as described in Section 5.

Based on the algorithm description the initial choice is: instance C (C1) from service type 1, instance B (B2) from service type 2 and instance B (B3) from service type 3.

In Table 3, we present the algorithm iterations based on the steps described in Section 6 (increase in the values is marked with the “+” symbol and decrease with the “−” symbol). For each service instance, there is a possible replacement based on the ConvertedIndex, with the corresponding availability, cost and time value differences. For the first experiment and since service type 1, is of major importance (as stated within the application workflow), in the first iteration the initial service instance C1 is replaced with the service instance that has the lowest ConvertedIndex (A1). Afterwards, for service type 2, the same process is attempted for instances H2, G2, A2, which have lower ConvertedIndex than the one of the selected service instance (B2). Nevertheless, the replacements fail due to the overall workflow cost limit that is being violated. The same applies for service type 3. Finally, the next optimum replacements are F3 and D2, which also fail due to the cost restriction. These replacements were attempted after the completion of the attempts for service type 3 since the algorithm steps describe that the replacements take place based on their differences so as to define the replacement order of the optimum service instances.

<table>
<thead>
<tr>
<th>Service instance</th>
<th>Obtained availability value</th>
<th>Obtained cost value</th>
<th>Obtained time value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>96.50</td>
<td>180.32</td>
<td>300.25</td>
</tr>
<tr>
<td>B3</td>
<td>90.10</td>
<td>175.20</td>
<td>400.05</td>
</tr>
<tr>
<td>C1</td>
<td>87.30</td>
<td>165.66</td>
<td>442.43</td>
</tr>
<tr>
<td>D1</td>
<td>93.80</td>
<td>178.90</td>
<td>321.78</td>
</tr>
<tr>
<td>E1</td>
<td>92.40</td>
<td>177.75</td>
<td>320.65</td>
</tr>
<tr>
<td>F1</td>
<td>90.20</td>
<td>177.10</td>
<td>330.12</td>
</tr>
<tr>
<td>G1</td>
<td>89.50</td>
<td>170.24</td>
<td>370.94</td>
</tr>
<tr>
<td>H1</td>
<td>90.10</td>
<td>170.69</td>
<td>368.49</td>
</tr>
<tr>
<td>A2</td>
<td>90.68</td>
<td>70.35</td>
<td>3.27</td>
</tr>
<tr>
<td>B2</td>
<td>80.15</td>
<td>62.76</td>
<td>6.64</td>
</tr>
<tr>
<td>C2</td>
<td>97.66</td>
<td>79.39</td>
<td>9.15</td>
</tr>
<tr>
<td>D2</td>
<td>99.84</td>
<td>82.27</td>
<td>5.85</td>
</tr>
<tr>
<td>E2</td>
<td>70.15</td>
<td>60.19</td>
<td>7.53</td>
</tr>
<tr>
<td>F2</td>
<td>88.38</td>
<td>69.84</td>
<td>5.99</td>
</tr>
<tr>
<td>G2</td>
<td>95.03</td>
<td>74.44</td>
<td>3.12</td>
</tr>
<tr>
<td>H2</td>
<td>90.60</td>
<td>72.90</td>
<td>2.28</td>
</tr>
<tr>
<td>A3</td>
<td>98.01</td>
<td>110.44</td>
<td>4.67</td>
</tr>
<tr>
<td>B3</td>
<td>81.90</td>
<td>80.94</td>
<td>5.89</td>
</tr>
<tr>
<td>C3</td>
<td>95.83</td>
<td>99.73</td>
<td>7.40</td>
</tr>
<tr>
<td>D3</td>
<td>90.64</td>
<td>96.90</td>
<td>5.12</td>
</tr>
<tr>
<td>E3</td>
<td>89.95</td>
<td>87.19</td>
<td>6.10</td>
</tr>
<tr>
<td>F3</td>
<td>97.58</td>
<td>105.84</td>
<td>4.99</td>
</tr>
<tr>
<td>G3</td>
<td>95.10</td>
<td>100.34</td>
<td>4.48</td>
</tr>
<tr>
<td>H3</td>
<td>94.58</td>
<td>100.96</td>
<td>5.19</td>
</tr>
</tbody>
</table>
Table 3
Algorithm iterations for experiment 1

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Initial service instance</th>
<th>Possible replacement service instance</th>
<th>Availability value difference</th>
<th>Cost value difference</th>
<th>Time value difference</th>
<th>Initial converted index</th>
<th>Possible new converted index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C_1</td>
<td>A_1</td>
<td>+9.20</td>
<td>+14.66</td>
<td>−142.18</td>
<td>16862.87</td>
<td>561.05</td>
</tr>
<tr>
<td>2</td>
<td>B_2</td>
<td>H_2</td>
<td>+10.45</td>
<td>+10.14</td>
<td>−4.36</td>
<td>34.90</td>
<td>1.84</td>
</tr>
<tr>
<td>3</td>
<td>B_2</td>
<td>G_2</td>
<td>+14.88</td>
<td>+11.68</td>
<td>−3.52</td>
<td>34.90</td>
<td>3.53</td>
</tr>
<tr>
<td>4</td>
<td>B_2</td>
<td>A_2</td>
<td>+10.53</td>
<td>+7.59</td>
<td>−3.37</td>
<td>34.90</td>
<td>3.91</td>
</tr>
<tr>
<td>5</td>
<td>B_3</td>
<td>G_3</td>
<td>+13.20</td>
<td>+19.40</td>
<td>−1.41</td>
<td>154.68</td>
<td>4.73</td>
</tr>
<tr>
<td>6</td>
<td>B_3</td>
<td>A_3</td>
<td>+16.11</td>
<td>+29.50</td>
<td>−1.22</td>
<td>154.68</td>
<td>6.40</td>
</tr>
<tr>
<td>7</td>
<td>B_3</td>
<td>F_3</td>
<td>+15.68</td>
<td>+24.90</td>
<td>−0.90</td>
<td>154.68</td>
<td>9.14</td>
</tr>
<tr>
<td>8</td>
<td>B_3</td>
<td>D_3</td>
<td>+8.74</td>
<td>+15.96</td>
<td>−0.77</td>
<td>154.68</td>
<td>10.56</td>
</tr>
<tr>
<td>9</td>
<td>B_3</td>
<td>H_3</td>
<td>+12.68</td>
<td>+20.02</td>
<td>−0.70</td>
<td>154.68</td>
<td>11.49</td>
</tr>
<tr>
<td>10</td>
<td>B_3</td>
<td>E_3</td>
<td>+8.05</td>
<td>+6.25</td>
<td>+0.21</td>
<td>154.68</td>
<td>31.23</td>
</tr>
<tr>
<td>11</td>
<td>B_2</td>
<td>D_2</td>
<td>+19.69</td>
<td>+19.51</td>
<td>−0.79</td>
<td>34.90</td>
<td>22.91</td>
</tr>
<tr>
<td>12</td>
<td>B_2</td>
<td>F_2</td>
<td>+8.23</td>
<td>+7.08</td>
<td>−0.66</td>
<td>34.90</td>
<td>23.92</td>
</tr>
</tbody>
</table>

After the above described iterations, no further possible replacements could be made and since the time constraint was not fulfilled due to service instance A_1, the scheduling scheme is initiated. The most time critical service type is service type 1, which belongs to category 1, while service types 2 and 3 can be executed in a parallel way and due to that they belong to the same category (category 0). The initial execution time for service instance A_1 is 300.25 time units, the overall workflow execution time is 306.89 (the time needed to execute A_1 plus the time needed to execute B2) and the overall workflow cost is 324.02 account units. In Table 4 we present the algorithm iterations within the scheduling scheme, where each time an additional service instance is selected based on the lowest execution time.

The next iterations of the algorithm do not conclude to any new additions either because the cost constraint/restriction is violated or because the execution time is increased with the addition of a new service instance.

The final concrete workflow is presented in Fig. 6.

Moreover, in Fig. 7, we cite a graphical representation of the algorithm iterations for this experiment. It represents the change of availability, cost and execution time based on the algorithm iterations (the “bold” horizontal lines state the user constraint and the “dashed” vertical one the scheduler scheme initialization). Since the user did not set any preference (major factor of selection), the figure shows that the mechanism made the selections based on the user constraints (thresholds to the QoS parameters), while on Iteration 13, the scheduler scheme is initiated in order to meet the time restriction.

7.5. Experiment 2

In this experiment the user defines via the GUI the following constraints/ restrictions for the QoS parameters: (i) Service Instances’ Availability: At least 80%, (ii) Overall Cost: Threshold to 350 account units, (iii) Overall Execution Time: 200 time units. Moreover, the user defines execution time as the major factor (preference) for the selection process and thus: Availability Slope = Cost Slope = 0 and Time Slope = 3. This demonstrates the Time Optimization use case as described in Section 5.

Based on the algorithm description the initial choice is instance C (C_1) from service type 1, instance B (B_2) from service type 2 and instance B (B_3) from service type 3.

Similar to the first experiment, in Table 5 we present the algorithm iterations. Since service type 1 is of major importance (as stated within the application workflow), in the first iteration the initial service instance C_1 is replaced with the service instance that has the lowest ConvertedIndex (A_1). The same process is completed successfully for service type 2 (replacement of B_2 with H_2), while for service type 3, the replacements fail due to the violation of the overall workflow cost limit.

After the above described iterations, no further possible replacements could be made and since the time constraint was not fulfilled due to service instance A_1, the scheduling scheme is initiated. The most time critical service type is service type...
Table 5
Algorithm iterations for experiment 2

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Initial service instance</th>
<th>Possible replacement service instance</th>
<th>Availability value difference</th>
<th>Cost value difference</th>
<th>Time value difference</th>
<th>Initial converted index</th>
<th>Possible new converted index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$C_1$</td>
<td>$A_1$</td>
<td>+9.20</td>
<td>+14.66</td>
<td>−142.18</td>
<td>16862.87</td>
<td>561.05</td>
</tr>
<tr>
<td>2</td>
<td>$B_2$</td>
<td>$H_2$</td>
<td>+10.45</td>
<td>+10.14</td>
<td>−4.36</td>
<td>34.90</td>
<td>1.84</td>
</tr>
<tr>
<td>3</td>
<td>$B_3$</td>
<td>$G_3$</td>
<td>+13.20</td>
<td>+19.40</td>
<td>−1.41</td>
<td>154.68</td>
<td>4.73</td>
</tr>
<tr>
<td>4</td>
<td>$B_3$</td>
<td>$A_3$</td>
<td>+16.11</td>
<td>+29.50</td>
<td>−1.22</td>
<td>154.68</td>
<td>6.40</td>
</tr>
<tr>
<td>5</td>
<td>$B_3$</td>
<td>$F_3$</td>
<td>+15.68</td>
<td>+24.90</td>
<td>−0.90</td>
<td>154.68</td>
<td>9.14</td>
</tr>
<tr>
<td>6</td>
<td>$B_3$</td>
<td>$D_3$</td>
<td>+12.68</td>
<td>+20.02</td>
<td>−0.70</td>
<td>154.68</td>
<td>10.56</td>
</tr>
<tr>
<td>7</td>
<td>$B_3$</td>
<td>$H_3$</td>
<td>+8.74</td>
<td>+15.96</td>
<td>−0.77</td>
<td>154.68</td>
<td>11.49</td>
</tr>
<tr>
<td>8</td>
<td>$B_3$</td>
<td>$E_3$</td>
<td>+8.05</td>
<td>+6.25</td>
<td>+0.21</td>
<td>154.68</td>
<td>31.23</td>
</tr>
</tbody>
</table>

1, which belongs to category 1, while service types 2 and 3 can be executed in a parallel way and due to that they belong to the same category (category 0). The initial execution time for service instance $A_1$ is 300.25 time units, the overall workflow execution time is 306.89 (the time needed to execute $A_1$ plus the time needed to execute $B_2$) and the overall workflow cost is 334.16 account units. In Table 6 we present the algorithm iterations within the scheduling scheme, where each time an additional service instance is selected based on the lowest execution time.

The possible additional service instances for service type 1 are: $E_1$, $D_1$, $F_1$, $H_1$, $G_1$, $B_1$ and $C_1$. Nevertheless, $B_1$ addition fails due to the overall workflow cost limit and $C_1$ addition fails since the new execution time is increased. The rest of the service instances decrease the execution time of service type 1 and since the user restrictions are not violated, the additions take place. Moreover, time optimization is attempted for service types 2 and 3 since this experiment demonstrates the time optimization case which applies for the whole workflow. However, service instance $G_2$ cannot be added since the other
service instance $G_3$ of a service type that belongs to the same category cannot be split due to the overall workflow cost restriction.

The final concrete workflow that is produced for experiment 2 is represented in Fig. 8.

Similar to the first experiment, in Fig. 9, we cite a graphical representation of the algorithm iterations. It represents the change of availability, cost and execution time based on the algorithm iterations (the “bold” horizontal lines state the user constraint and the “dashed” vertical one the scheduler scheme initialization). Since the user set as preference (major factor of selection) the time optimization, the figure shows that the selections are made by the mechanism based on the user constraints (thresholds to the QoS parameters), while on Iteration 8, the scheduler scheme is initiated in order to minimize the overall workflow execution time. From Iteration 8 till Iteration 16 the available budget (cost restriction) is spent in order to minimize the time to minimum.

A number of experiments have been completed for the algorithm evaluation and the above are only indicative ones. However, they demonstrate the mapping of workflow processes to Grid service instances into a real application scenario by not only using a scheduler to meet time limits, or not only trying to minimize one parameter (e.g. cost) or maximize another (e.g. availability) but dealing with them in a combined way and based not only on the “hard limits” (expressed with the user’s constraints) but on the limits’ importance as well (expressed with the user’s preferences).

8. Conclusions

In this paper we have studied a mechanism for workflow mapping with regard to Quality of Service information which is expected to significantly increase the effort to provide the Grid environment with a dynamic QoS capability. The aim of our work is to define a concrete workflow based on various parameters stated from the user along with the innovation of taking into account user’s preferences, in comparison to the exposed QoS values from the service providers.

Consequently, we provided the experimental results that demonstrate and evaluate the operation of the proposed mechanism for a real application (3D image rendering), while the presented set of parameters (application, resource, user-defined) and their integration into the mechanism was revealed. Given the experimental results, it appears that the mechanism makes the selections and as a result defines the concrete workflow taking into account the user constraints and preferences. The experiments showed promising results and therefore the performance of the mechanism is considered to be well established allowing the adoption of it in any heterogeneous and especially Grid system that seeks to bring QoS knowledge within the workflow mapping process.

In general, providing workflow mapping capabilities based on QoS metrics is of major importance for enterprises since it allows the adoption of different business models by designing the workflow according to QoS metrics with a priori knowledge of the provided QoS for the customers. The presented workflow mapping mechanism allows the selection of the service instances within given workflows based on their QoS in order to fulfil customer expectations. The adoption of different business models and objectives is achieved by evaluating the alternative strategies and monitoring any changes/violations that may occur, which will have an important impact on the strategies, methodologies, and structure of business processes. Therefore, the completion of a workflow according to initial QoS requirements may require a rescheduling in response to unexpected progress, delays, or violations. When adaptation is necessary, a set of potential alternatives is generated, with the objective of changing a workflow as its QoS continues to meet initial requirements. For each alternative, prior to actually carrying out the adaptation in a running workflow, it is necessary to estimate its impact on the workflow QoS. The latter is achieved by not only selecting the optimum service instances within a given workflow but also by characterizing the not-selected instances in order to have a priori knowledge if they are needed in a rescheduling workflow process. Each time a change is completed in a workflow, the end-to-end QoS level is re-estimated in order to meet the user constraints.
Notwithstanding, it is within our future plans to attempt to comprise a set of parameters in the selection process, while other pertinent issues such as the integration of the QoS information to dynamic SLA negotiations will be examined. These steps are regarded as significantly important for the dynamic end-to-end provision of QoS, since monitoring of the signed SLAs will provide an additional criterion for the workflow mapping by characterizing the service providers based on their prior behaviour (e.g., SLA violations). Furthermore, the algorithm implemented within the mechanism can be improved by including the dependencies between service instances in order to proceed with selections based on them. This can be achieved by setting levels on the QoS parameters that imply the dependencies between them. Finally, given the fact that the current trends in Grid computing imply the adoption of the OGSA based architectural design implemented by WSRF, it is interesting to translate the workflow mapping scheme into requests for selections as Simple Object Access Protocol (SOAP) [38] messages towards services that would be running on the Grid infrastructure.

In conclusion, unlike other heterogeneous systems [39], Grids have not yet adopted an effective scheme that will facilitate end-to-end QoS provisioning. In that rationale, we have shown the importance of mapping workflow application processes to Grid services with regard to the Quality of Service offered by the service providers and the corresponding one demanded by the users. While QoS management is of high importance for organizations, research is necessary in mechanisms to control the quality of service and map workflows based on it. Based on the reviewed literature on quality of service in other areas, and accounting for the particularities of workflow systems and applications, we define an algorithm (implemented within the workflow mapping component) which currently includes three dimensions: availability, cost and time — as initial indicative ones. The use of the aforementioned mechanism increases the added value of workflow systems to organizations, since non-functional aspects of workflows can be described and end-to-end QoS provision can be achieved.

Acknowledgments

This work has been supported by the NextGRID project and has been partly funded by the European Commission’s IST activity of the 6th Framework Programme under contract number 511563.
Appendix

In this section we cite a figure (Fig. A.1) presenting the methods and the variables used within the service implementation of the workflow mapping mechanism.

References


Dr. Dimosthenis Kyriazis received the diploma from the National Technical University of Athens, Athens, Greece in 2001, his MBA in Information Systems Management from the Inter-Departmental Postgraduate Program “Engineering and Economic Systems” co-organized by NTUA, UNIPL, UOA in 2003 and his Ph.D. from the Electrical and Computer Engineering Department of the National Technical University of Athens in 2007. He is currently a Research Associate in the Telecommunication Laboratory of the Institute of Communication and Computer Systems (ICCS), Athens, Greece and participating in several EU and Nationally funded projects. His research interests include Grid computing, quality of service and workflow management in heterogeneous systems, information engineering and their application and business extensions.

Konstantinos Tserpes is a Research Associate in the Telecommunication Laboratory of the Institute of Communication and Computer Systems (ICCS), Athens, Greece. He graduated from the Computer Engineering and Informatics Department, University of Patras, Greece. In 2006 he received his MBA in Information Systems Management. He is currently studying for his Ph.D. in the area of Grid Computing in the school of Electrical and Computer Engineers of the National Technical University of Athens (NTUA).

He has been involved in several EU and Nationally funded projects and his research interests are revolving around Grid Computing and their application and business extensions.

Andreas Menychtas received the diploma from the School of Electrical and Computer Engineering, National Technical University of Athens, Greece in 2004. Currently, he is pursuing his Ph.D. in the Telecommunication Laboratory of Electrical and Computer Engineering School of the National Technical University of Athens and works as research associate in the Institute of Communication and Computer Systems participating in several EU and Nationally funded projects such as GRIA and NEXTINGRID. His research interests include Grid Computing, Distributed Systems, Web Services, Security and Information Engineering.

Dr. Antonio Litke received the diploma from the Department of Computer Engineering and Informatics of the University of Patras, Greece in 1999, and the Ph.D. from the Electrical and Computer Engineering Department of the National Technical University of Athens in 2006. Currently, he is working in the Telecommunication Laboratory of the Electrical and Computer Engineering Department of the National Technical University of Athens as researcher participating in numerous EU and Nationally funded projects. His research interests include Grid computing, resource management in heterogeneous systems, Web services and information engineering.

Prof. Theodora Varvarigou received the B. Tech. degree from the National Technical University of Athens, Greece in 1988, the MS degrees in Electrical Engineering (1989) and in Computer Science (1991) from Stanford University, Stanford, California in 1989 and the Ph.D. degree from Stanford University as well in 1991. She worked at AT&T Bell Labs, Holmdel, New Jersey between 1991 and 1995. Between 1995 and 1997 she worked as an Assistant Professor at the Technical University of Crete, Chania, Greece. Since 1997 she has been working as an Associate Professor at the National Technical University of Athens. Her research interests include Grid Computings parallel algorithms and architectures, fault-tolerant computation, optimisation algorithms and content management.