# A Framework for QoS-based Resource Brokering in Grid Computing

Nadia Ranaldo and Eugenio Zimeo

Department of Engineering, Research Centre on Software Technology (RCOST)
University of Sannio
82100 Benevento, Italy
{ranaldo, zimeo}@unisannio.it

**Abstract.** The effective and efficient exploitation of Grid computing facilities requires advanced resource management systems to automatically and transparently ensure the fulfillment not only of functional requirements but also of non-functional ones. This paper presents a framework for brokering of Grid resources, virtualized through Web Services, which can be dynamically configured with respect to multiple syntactic and semantic description languages and related matching strategies. Hence, it discovers and selects resources and automatically allocates application tasks to them on the basis of both functional and Quality of Service (QoS) requirements. In particular the paper presents the framework specialization, which aims to select a pool of resources whose overall performance allows for satisfying time and cost constraints for the execution of an application partitioned in concurrent tasks according to the data parallelism pattern.

**Keywords:** Resource management, Brokering, Grid Computing, Web Services, Quality of Service.

## 1 Introduction

Thanks to the increasing amount of resources available across the Internet and to improvements of wide-area network performance, in recent years grid computing is emerging as a viable paradigm to satisfy the continuous growth of computation power demand, which often can not be fulfilled exploiting the inner resources of a single organization. This trend is also promoting new business models for providers that would deliver Grid computing functionalities, eventually customized on demand, to host applications and to meet customer needs [1].

After a first generation of solutions based on dedicated technologies, the diffusion of the Web has proposed new architectural (Service-Oriented Architecture – SOA) and technological (Web Services) approaches to address heterogeneity, distribution, security and interoperability in large-scale systems. So, grid applications can be viewed as the composition (workflow) of independent services delivered by distributed providers [2].

In particular, a Grid workflow can be obtained composing domain dependent

services, which virtualize the access to specific and high-level utilities typically delivered by providers that leverage high-performance dedicated clusters and software libraries for complex computations. In this case, functional matching strategies have to be adopted to discover and select the services that deliver the needed domain-dependent functionalities for the enactment of the overall application.

More often, differently from other application domains tied to B2B environments, Grid workflows orchestrate services that virtualize the access to resources delivering low-level functionalities, such as data acquisition, computation and storage. Such services are characterized by heterogeneous assets and performance (due to heterogeneity and sharing of resources onto which they are deployed). As a consequence, heterogeneity represents one of the key elements of Grid computing but also one of the main problems for an efficient execution of Grid applications.

For this reason, a lot of research in this area has been devoted to the definition of infrastructure components able to hide heterogeneity from the computational point of view: *computing power transparency*. In a connection model based on middle agents, such as the model proposed by SOA, a fundamental role for achieving the desired transparency is played by resource managers, matchmakers and brokers [3]. In grid context, these components have, typically, a slightly different behaviour with respect to equivalent components in a pure Web Services environment: the selection of services is performed mainly on the basis of QoS attributes that characterize the heterogeneity of resources onto which they are deployed. In this scenario, brokers are able to automatically discover available functionalities, choose and schedule the ones that satisfy the specific user needs.

Many compute and data-intensive functionalities in scientific and grid workflows (such as linear algebra, image processing, database searching, etc.) are characterized by coarse-grained parallelism that allows for increasing performance by exploiting a pool of distributed resources using parallel computing patterns such as simple parallelism, data parallelism and pipeline patterns [2]. A full exploitation of multiple resources to execute Grid workflows will be reached if the following main issues will be taken into account: (1) the adoption of matching strategies able to find a pool of resources satisfying global constraints on applications; (2) definition of formal languages for QoS description of Grid services in order to avoid ambiguity during matching; (3) mechanisms for dynamic and transparent composition and coordination of services.

In this paper, we answer to the first two issues by proposing a framework for QoS brokering of resources virtualized through Web Services and its customization. The framework is able to automatically allocate application tasks based on the data parallelism pattern through a time and cost-based matching strategy, called time minimization matching strategy. We proposed its basic algorithm, based on divisible load theory, in [4]. We proved, moreover, through an experimental analysis, the validity and accuracy of the system to search and select resources that ensure execution times of real complex applications within prefixed constraints.

The resource broker is based on a matchmaking framework designed in the context of the LOCOSP project [5] to support multi-criteria matching strategies in B2B and Grid based environments. Such framework is extensible and customizable with respect to application scenario through dynamic configuration of syntactic-, structural- and semantic-based discovery and matching strategies, features that make

it a suitable and easy-to-use environment to test new search and selection strategies.

The rest of the paper is organized as follows. Section 2 presents related work and technology. Section 3 describes the matchmaking framework. Section 4 briefly describes the time minimization matching strategy. Section 5 illustrates the matchmaker customization for the integration of the time minimization matching strategy. Section 6 presents an experimental analysis of the matchmaking system. Finally Section 7 concludes the paper and introduces future work.

## 2 Related Work and Technology

Many Grid systems adopt system-oriented or application-oriented matching heuristics that try to optimise respectively resource utilization and time execution with respect to available resources [6] [7]. G-QoSM [8], a framework for QoS-based service management, focuses on discovery of grid computing services on the basis of QoS criteria and on mechanisms to guarantee QoS levels by means of "contract-based agreements" between service provider and service requester. However, G-QoSM is based on a non standard extension of UDDI, which supports syntactic QoS specifications included in non standardized WSDL-based descriptions of computational services. In our work, instead, we aim to improve the matching process adopting a standardizable and semantic-based description of QoS properties that allow for well formalizing QoS knowledge and so for overcoming syntactic languages limitations due to heterogeneity.

On the other hand, while standard technologies for Web services (SOAP, WSDL and UDDI) define precisely syntax and data structure-based descriptions, there are currently no standards for semantic description and query. An interesting solution is the one used in Semantic Web [9]. While for the specification of functional requirements some proposals for standardization of ontologies have been promoted, such as OWL-S [10] and WSMO [11], only recently some results have been reached for QoS requirements. In particular, preliminary efforts can be found in DAML-QoS Ontology [12], QoSOnt ontology [13] and more recently in WSMO [14]. However, such approaches do not take into account QoS aspects that are specific for Grid services and scientific workflows. In fact, beyond to classical QoS attributes defined for a Web Service B2B environment, such as reliability, cost, time response, etc., other specific QoS requirements could be specified for Grid services, such as execution time for computational tasks, real-time capability of data acquisition services, minimum storage capability for storage services, etc.

In the context of workflow management, interesting works have been proposed in this field. For example a semantic-driven Web services discovery system for workflows is presented in [15], and a QoS-aware optimization matching approach for workflows is described in [16]. An interesting approach is, moreover, proposed in [17]. It takes into account the static scheduling of workflows modelled as a pipeline of parameter sweep tasks aiming to a fine-grained time optimization in a heterogeneous environment. On the other hand, our approach differs from them because it better focuses on the definition of a flexible brokering framework and matching strategies for QoS-driven and optimized execution of scientific workflows characterized by data

parallel tasks that can concurrently exploit multiple computing resources.

In the context of ontology definition of QoS attributes, some works have been proposed [18] [19]. In this paper, we adopt and extend to the Grid environment the onQoS ontology proposed in [20] for the description of QoS attributes of Web Services. This ontology tries to overcome limitations and scarce homogeneity of many semantic models currently available for QoS description and uses metrics to understand, describe and control the QoS in the matching phase in a quantitative way.

In the context of parallel programming pattern, linear programming approaches and related heuristics have been led to interesting and low-complex results in divisible load theory [21] that treats the overall task to execute as a continuous workload. Recently such theory has been applied for the minimization of time execution starting from performance description of computational and network resources of large-scale platforms [22]. On the other hand, in a future commercialisation of Grid systems, a resource characterization based only on performance features is not sufficient to properly guide the selection process.

Economic factor in task allocation has been taken into account in [23] for cost minimization under time constraints in the divisible loads context. On the contrary, we focus on time minimization under time and cost constraints and moreover deal with realistic granularity levels of workload fraction assigned to each resource, still keeping heuristic complexity non-dependent on the input size but only on the resource amount.

A QoS brokering system which deals with cost constraints is the Grid Service Broker (GSB) [24], which supports access to both computational and data Grids. GSB can transparently access resources that are exposed by various low–level, Grid middleware solutions, such as Globus Toolkit 4 and Alchemi [25] and published on a custom XML-based Grid Market Directory registry. It supports deadline and budget-constrained matching strategies for the scheduling of parameter sweet applications. Heuristics adopted by GSB dynamically allocate a task at a time considering the current state of resources until the budget is consumed. As a consequence they are not useful for scheduling generic data parallel tasks because do not deal the execution of all the required tasks within a specified deadline. On the contrary our approach aims to grant task execution within specified deadline and budget. As a consequence, it can be effectively adopted in a real environment in which reservation mechanisms (such as in ICENI [26] and GridARM [27]) grant the availability of a resource with negotiated QoS.

## 3 Service Matchmaker Framework

The Service Matchmaker [28] is a key component of an ongoing project [5] that aims at defining and implementing a flexible broker for service composition in SOA-based environments. It supports customizable and multi-criteria discovery and matching service strategies for different application domains. To ensure a high flexibility, the framework is designed according to the component framework approach. Its basic infrastructure is able to automatically manage and trigger well-defined activities for discovery and matching of services, through the definition of the main abstractions,

components and behaviours necessary to execute them (figure 1).

The basic functions are captured in the *Matchmaker Core*, which represents the architectural skeleton of the framework, whilst the specialization is realized through *hot-spots* that permit to customize framework behaviours for specific application domains and requirements. The Matchmaker Core, at start-up, uses configuration files to load specified components that specialize the framework behaviour.

Because of the lack of a unique standard language to describe different aspects of a service, the matchmaking framework supports multi-criteria discovery and matching strategies that can be adopted on different service descriptions (including functional and non functional aspects), each of which describing a specific aspect of a service and eventually adopting a different language.

The matching process starts with a request containing the description of the desired query service (called *template*) submitted by the user through the *Matchmaking API*. As first step, the *Discovery Engine* uses the search functionality offered by the *Registry* to retrieve information on advertised services. The search space of candidate services (called *targets*), initially reduced by the Discovery Engine, is further filtered by the *Matchmaker Manager* that is based on a Pipe & Filter architecture.

The Matchmaker Manager, in particular, adopts two customizable *Search Pipes* of *Matching Filters*, one for functional aspects and another one for non-functional aspects. The Search Pipes are able to reduce more and more the service subspace, returning finally a matching result between the template and the targets. The Search Pipes are customized specifying the Matching Filters and their order. Each Matching Filter can be characterized by a distinct matching strategy.

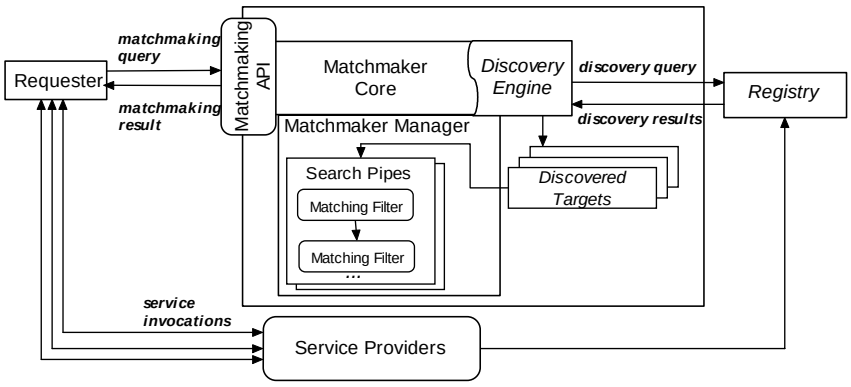


**Fig. 1.** Service Matchmaker Framework.

A Matching Filter analyzes a specific description of the targets of the received target subspace performing a matching strategy and is characterized by a Matching Engine and by a Matching Function. The *Matching Engine* receives the subspace of target descriptions and takes care to return the targets that satisfy a matching strategy. Typically, but not necessary, it associates the satisfaction degree of each target with respect to specified requirements for that description invoking the *Matching Function*, that returns a matching score for a specific strategy. The matching of semantic

description requires a component with automatic reasoning capabilities. To this end, a *Semantic Matching Function* is configured with respect to an inference engine, through a *Reasoner*, a configurable component whose specialization permits to define the reasoning engine more suitable to specific aims.

## 4 Time Minimization Matching Strategy

The time minimization matching strategy addresses tasks of a Grid application that can be parallelized through the data parallelism pattern, also known as the master-slave pattern. Similarly to the application model of divisible load theory [21], a pool of slaves performs the overall workload, that is decomposed by the master into a high number (but finite) of sub-tasks, called *atomic tasks*, the smallest parts of the original workload that can be independently mapped and executed onto different resources without casual precedence relationships.

The application is characterized by the computation size, which corresponds to the total number $N$ of atomic tasks, each of which is characterized by the same complexity in terms of computation, data storage and data transfer aspects.

A Grid system is modelled as a finite set $R = \{R_1, R_2, .., R_M\}$ of available resources communicating through a fully connected wide area network. Each resource is exposed as a Grid service and is characterized by performance parameters, cost and capacity, which depend on the application to execute. Time execution and cost are assumed to be proportional to the amount of atomic tasks assigned to the resource in a linear way. Following the Grid model proposed in [22], the bandwidth on WAN links is not shared and each resource reaches the WAN through a LAN link. In the case of resource reservation mechanisms, only one communication flow goes through the link, that so receives a fixed bandwidth that can be predicted in advance.

Under these assumptions, for each resource $R_i \in R$ the performance is modeled as the total time $t_i$ required for processing an atomic task. The resource cost, called $c_i$, is the cost of resource usage for the processing of an atomic task. Finally the capacity, $g_i$, represents the maximum number of atomic tasks that can be assigned to resource $R_i$. The QoS parameters specified by the user to model a Grid service request are: (1) the maximum execution time, which represents the deadline, called $D$, (2) the total available budget, called $B$, and (3) the capacity of requested data parallel task, $N$, that is the total number of atomic tasks which have to be executed.

The time minimization matching strategy regards the problem of finding the "best" set of resources among which to distribute the workload $N$, so that the aggregate cost for resource usage is lower than budget B (but not necessarily the minimum) and that are able to complete the application execution as quickly as possible (time minimization) and within deadline D.

More formally the matching strategy goal is:

Minimize:

$$\max_{i=1..M}(t_i n_i), \tag{1}$$

subject to:

$$t_i n_i \le D \quad i = 1..M\,, \tag{2}$$

$$\sum_{i=1}^{M} n_i = N\,, \tag{3}$$

$$\sum_{i=1}^{M} c_i n_i \le B\,, \tag{4}$$

$$n_i \le g_i \quad i = 1..M\,, \tag{5}$$

where $n_i \in \mathbf{N_0}$ is the number of atomic tasks assigned to resource $R_i$ and represents the used portion of capacity $g_i$ of resource $R_i$, (2) ensures that the deadline is not exceeded, (3) is the constraint to require execution of all $N$ atomic tasks, (4) ensures that the budget is not exceeded and finally (5) that the capacity constraint on each resource is not exceeded.

To minimize the execution time, considering the optimality principle of the divisible load theory [21], the time minimization strategy can be solved through a heuristic that starts assigning a fractional and non uniform number of atomic tasks to the available resources such that they will finish at the same time:

$$t_i n_i = t_j n_j \quad i, j = 1..M. \tag{6}$$

Indicated with $\bar{t}$ the execution time of an atomic task for the resource with the highest performance, and with $\bar{n}$ the number of atomic tasks assigned to it, the evaluation of $\bar{n}$ through (6) permits to evaluate all the $n_i$ as:

$$n_i = \frac{\bar{t}}{t_i}\ \bar{n} \quad i : 1..M\,, \tag{7}$$

that, applied to (3) and (4), leads to:

$$\sum_{i=1}^{M} n_i = \sum_{i=1}^{M} \frac{\bar{t}}{t_i}\ \bar{n} = N\,, \tag{8}$$

$$\sum_{i=1}^{M} n_i c_i = \sum_{i=1}^{M} \frac{\bar{t}}{t_i}\ \bar{n} c_i \le B. \tag{9}$$

Generally constraints (2), (5), (8) and (9) are not satisfied simultaneously. A possible approach to find a near-optimal solution is the adoption of the iterative and low-complexity algorithm described in [4] that allows for finding integer values of $n_i$. In particular, in the case of budget exceeding, the idea is to decrease the number of atomic tasks assigned to the most expensive resource, and to distribute them among the remaining resources in a proportional manner to their performance. When such re-distribution of atomic tasks is not enough to not exceed budget, the most expensive resource is removed from the list of available resources and the procedure is iterated.

## 5 Service Matchmaker Specialization

The proposed QoS-based resource broker for Grid computing was implemented as a specialization of the Service Matchmaker framework which integrates the time minimization matching strategy.

This grid-oriented specialization is obtained through an XML-based file that specifies description languages and related functional and non functional discovery and matching strategies and through the implementation of the Discovery Engine, Machine Engines, Matching Functions and Reasoners for semantic matching strategies of low-level Grid services which virtualise computational resources.

The family of adopted description languages consists of:

1) WSDL (version 1.1), for abstract syntactic description on service interface and concrete syntactic description on binding and endpoint;

2) OWL-S (version 1.1) for abstract descriptions of functional and data semantics.

3) An extension of the onQoS ontology for QoS description, called GonQoS, supporting the descriptions of the parameters necessary to execute the time minimization matching strategy.

The Discovery Engine specialization interacts with the UDDI registry through the UDDI proxy UDDI4J, [29], an open-source Java implementation of specification for business registry and UDDI API. It performs a minimum functional search exploiting UDDI meta-data on descriptions (for example taxonomy, categoryBag, tmodel, businessServices, etc.). In particular, we use a functional aspect-based query which permits a category-based discovery of Grid services through the NAICS taxonomy supported by UDDI.

The Search Pipe for functional aspects uses three Matching Filters:

- Semantic matching on service operations based on OWL-S;
- Semantic matching on service input/output and fault based on OWL-S;
- Structural-syntactic matching on WSDL description of service operations.

The Search Pipe for non functional aspects uses two Matching Filters:

- *Basic QoS-based Matching Filter* (BQMF), for semantic matching on QoS metrics based on GonQoS ontology.
- *Aggregate QoS-based Matching Filter* (AQMF), performing a QoS-based matching strategy that returns the set of services which satisfy QoS requirements in an aggregate manner.

Functional Matching Filters based on ontology descriptions and BQMF are based on the matching approach proposed by Paolucci et al. [30]. The Matching Engine used by such filters is called *One-to-One Matching Engine.* It invokes repetitively the associated Matching Function for each target of the target subspace, assigns to each of them a matching result calculated by the Matching Function, and filters the targets which do not satisfy query criteria.

The structural-syntactic Matching Function is based on the strategy proposed by Wang e Stroulia [31].

The matching filter AQMF exploits the *One-to-Many Matching Engine*, which processes all the target subspace returned from the previous filter BQMS to satisfy QoS semantic parameters of a query expressed using GonQoS. The result returned by the filter AQMF is based on the time minimization matching strategy, which

calculates the portion of the overall number of atomic tasks specified in the query to assign to each target. The GonQoS ontology is accessed through a Reasoner which exploits the *Jena* (version 2.4) framework [32] specialized in order to use the inference engine *Pellet* (version 1.3) [33].

### 5.1 GonQoS for Grid Services

The GonQoS ontology for description of QoS parameters of Grid services is based on onQoS [20], an ontology developed using OWL for QoS description, advertising and query of Web services, designed in order to ensure simplicity while maintaining flexibility and extendibility features. It is tied to the OWL-S ontology, which permits to connect a QoS description to the corresponding functional one.

Following the classical approach for ontology definition, GonQoS is organized into three extensible complementary levels. The upper ontology defines the ontological language, which is the basic concepts to model Web service QoS, such as the main properties and restrictions of QoS metrics.

In this ontology, a QoS description of a Web Service is represented by a set of *QoS metrics*. For the QoS description of a service it is necessary to define a new entity of QoSMetric concept for each QoS parameter, that means to define: the measured parameter, the measurement scale, the measurement process, one or more measured values belonging to the measurement scale.

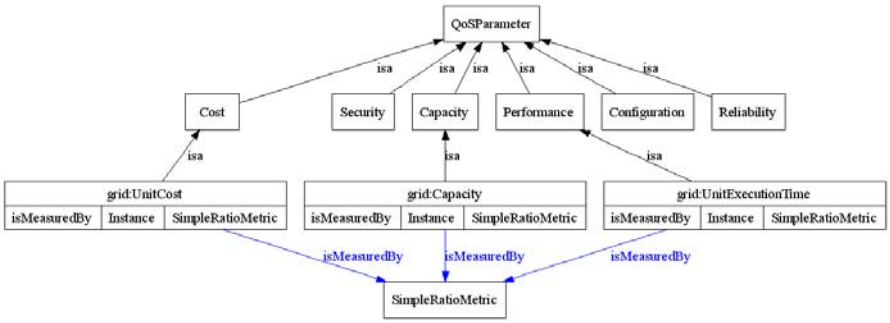


**Fig. 2.** GonQoS Ontology.

The middle ontology is a specialization of the first one and is domain independent. Examples are the specialization of QoS parameters for Avaialability, Performance, Reliabiliy, Cost and Capacity categories. Perfomance is further specialized in Throughput, ResponseTime, Latency, etc.

The low ontology can contain domain-dependent specializations of the ontology in a specific domain. At this level some grid-specific concepts for QoS definition are introduced. Figure 2 details the set of QoS Parameters to characterize the query, the advertised services and the returned result of the time minimization matching strategy.

– *grid-UnitExecutionTime*: for query: maximum interval time within which a task has to be executed; for advertising: interval time required to execute an atomic

task. It is expressed in seconds as float values.

- *grid-UnitCost*: for query: maximum budget which can be spent to execute a task; for advertising: cost for the execution of an atomic task. It is expressed in euros as float values.
- *grid-Capacity*: for query: overall number of atomic tasks to execute; for advertising: maximum number of atomic tasks which can be executed for a single request. It is an integer value.
- *grid-UnitCommunicationTime*: for query: interval time within which to transfer input data and output results of overall atomic tasks; for advertising: interval time required to transfer input data for an atomic task and its output results. It is expressed in seconds as float values. This parameter will be better defined in a future work in the context of a time minimization matching strategy which will take into account data transfer overheads for large scale distributed systems.

These QoS parameters are classified as *Simple Ratio Metric*, a specialization of generic QoS metric that permits to define queries adopting relation operators (such as better or equal, tightly less of a certain value, etc.).

## 6 Matching Strategy Evaluation

The proposed framework for grid resource brokering was tested in order to evaluate its validity and accuracy for the discovery and selection of resources that satisfy deadline and budget constraints through the time minimization matching strategy.

An UDDI registry was used for the advertisement of a set of Grid computing services. For each of them the providers specified WSDL descriptions, an UDDI categoryBag meta-data for functional aspects and QoS metrics through the GonQoS ontology language. The query is formulated through the UDDI categoryBag for the functional discovery of computation services and a QoS description of the required deadline, budget and number of atomic tasks to execute. Semantic functional aspects are not exploited in this experimentation, since we consider services virtualizing only computing functionalities. The QoS description is adopted to perform the BQMF in order to throw out the targets which does not satisfy the following conditions:

$$c_i \leq B \ , \quad t_i \leq D \ , \quad g_i \leq N \ .$$

The AQMF performs the time minimization matching strategy assigning to the filtered targets a part of the query capacity *N*.

A computation service is implemented as a Web Service that takes a certain interval time to execute an atomic task on the basis of performance capability of the resource on which it is deployed. The overall service query is satisfied invoking in a concurrent way the selected services and waiting for their completion.

In this experimentation, we consider ten Grid services deployed in Axis 2.0 container based on Tomcat onto ten distributed resources equipped with a Pentium IV 2.4 GHz and 512 MB of RAM. Resources are inter-connected through a Fast Ethernet LAN that does not cause significant effects on adopted Grid model since data transfers of atomic tasks are kept slight with respect to computation tasks. In this experimentation, resource heterogeneity in terms of cost and performance are

emulated taking into account experimentation results previously conducted on a compute-intensive application for power system security analysis [34]. In particular table 1 summarizes the QoS parameters associated to each service. The cost parameters were chosen to be nearly inversely proportional to resource performance. In table 1 the service instances with the same QoS parameters are grouped in the same service template and the number of services for each template is also reported.

**Table 1.** Test bed configuration.

| Service | | | QoS per Atomic Task | |
|---|---|---|---|---|
| Template | # instances | Capacity | Execution Time | Cost |
| S1 | 1 | 100 | 1.39 s | 60.0 € |
| S2 | 1 | 100 | 9.6 s | 10.0 € |
| S3 | 4 | 100 | 75.0 s | 4.0 € |
| S4 | 1 | 100 | 77.0 s | 4.0 € |
| S5 | 3 | 100 | 70.0 s | 5.0 € |

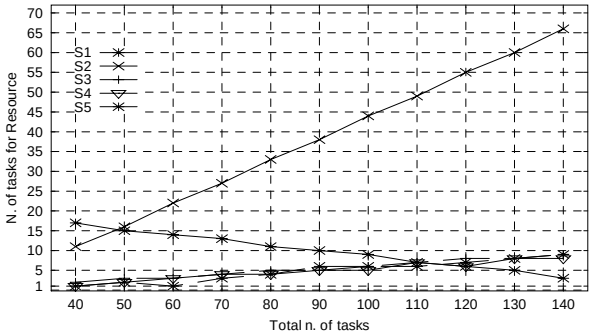


**Fig. 3.** Tasks assigned to each service varying the total amount of tasks.

Figure 3 shows the number of atomic tasks assigned to the services considering a user request of 900.0 s for deadline, 1200.00 € for budget and a varying capacity, from 40 to 140 atomic tasks, in order to simulate different application computation sizes. Because of similar capabilities of services with template S3, S4 and S5, such strategy assigns roughly the same number of tasks to each of them. In this scenario the budget value of 1200.00 € is not sufficient to completely exploit expensive services. For this reason, in order to assign all the atomic tasks, the time minimization strategy decreases the number of atomic tasks assigned to the most expensive service, that in this case is the one with template S2, and assigns them to the less expensive ones, until the deadline is not exceeded for each of them. Finally, we note that because the service with template S1 has better performance with respect to services with template S3, S4 and S5, it receives a larger amount of atomic tasks, which increases with the query capacity.

Figure 4 shows the estimated execution times for the overall computation considering a query capacity of 180 atomic tasks, deadline of 900.00 s and a varying budget starting from 1200.00 €, which is the minimum value to satisfy deadline, to

4800.00 €. The estimated execution time is evaluated as the maximum value among the execution times of each service. We can note that by increasing the budget, the algorithm ensures a decreasing execution time thanks to the possibility to allocate more tasks to the expensive and higher performance services.

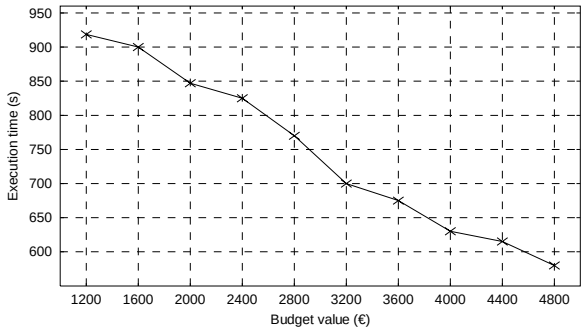


**Fig. 4.** Execution time varying budget with D = 900.0 s.

Finally, figure 5 shows the execution times that we actually measured running the application on the test-bed varying the query capacity. It shows the measured execution times and the estimated execution times by using the time minimization matching strategy. As it is possible to observe, the measured execution times have a nearly linear trend with respect to the atomic tasks to execute, condition that proves the efficiency of the overall system. Such time are, moreover, very near to the execution times estimated by the algorithm.

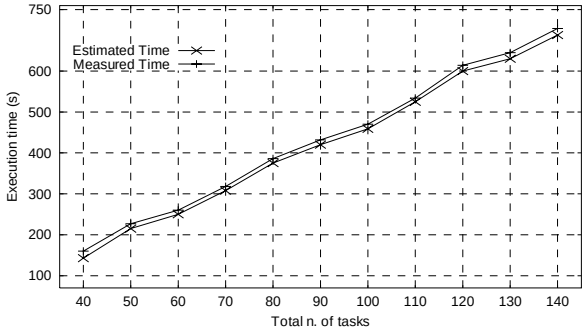


**Fig. 5**. Experimental results.

Finally these experimental results proved that the proposed QoS-based brokering framework represents a useful and flexible system for automatically acquiring computational resources when they are necessary, since its accuracy is high and the overhead that users pay for using such system for performing complex tasks is negligible if compared to the improvement of performance and usability.

## 7 Conclusion

The paper presented the design and evaluation of a framework for QoS brokering of Grid resources virtualized by Web Services. It is based on the Service Matchmaker, a framework that delivers customizable syntactic and semantic discovery and matching strategies. In this work, we presented its customization for supporting the selection resources among which to distribute the workload of a data parallel task to minimize execution time and to satisfy deadline and budget constraints.

The integration of service invocation mechanisms through workflow technologies, in order to make automatic and transparent to the user the distribution and deployment of applications on multiple resources, will be taken into account in a future work.

In particular, we are currently focusing on a dynamic composition and binding technique of services able to transparently and hierarchically distribute applications based on the data parallelism pattern, following the approach proposed by the authors in [35] for the specification of partition policy of input data and of assembling policy of results.

**Acknowledgments.** The work described in this paper is framed within the activities of the Core Grid FP6 Network of Excellence funded by the European Commission.

## References

1. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical Report, Open Grid Service Infrastructure WG, Global Grid Forum (2002)
2. Pautasso, C., Alonso, G.: Parallel Computing Patterns for Grid Workflows. In: the HPDC2006 Workshop on Workflows in Support of Large-Scale Science. France (2006)
3. Krauter, K., Buyya, R., Maheswaran, M.: A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. International Journal of Software, Practice and Experience, vol. 32(2), pp. 135--164. Wiley Press, USA (2002)
4. Ranaldo, N., Zimeo, E.: An Economy-driven Mapping Heuristic for Hierarchical Master-Slave Applications in Grid Systems. In: IEEE IPDPS 06, Greece (2006)
5. LOCOSP project, http://plone.rcost.unisannio.it/locosp
6. Li, K.: Job Scheduling and Processor Allocation for Grid Computing on Metacomputers. Journal of Parallel and Distributed Computing, vol. 65(11), pp. 1406--1418 (2005)
7. Braun, T., at All: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, vol. 61(6), pp. 810--837 (2001)
8. Li, M., van Santen, P., Walker, D.W., Rana, O.F., Baker, M.A.: PortalLab: A Web Services Oriented Toolkit for Semantic Grid Portals. In: IEEE CCGrid, pp. 190--197. Japan (2003)
9. Tangmunarunkit, H., Decker, S., Kesselman, C.: Ontology-based Resource Matching in the Grid – The Grid Meets the Semantic Web. In: the Second ISWC, Miami, Florida (2003)
10. OWL-S, An OWL-based Web service ontology, http://www.daml.org/services/owl-s/1.0/
11. WSMO, http://www.wsmo.org/
12. Zhou, C., Chia, L.-T., Lee, B.-S.: DAML-QoS Ontology for Web Services. In: the International Conference on Web Services 2004, pp. 472--479. IEEE Press San Diego, California, USA (2004)
13. Dobson, G., Lock, R.: QoSOnt: an Ontology for QoS in Service-Centric Systems. UK e-

Science AHM (2005)
14. Toma, I., Foxvoug, D., Jaeger, M. C., Roman, D., Strang, T., Fensel, D.: Modeling QoS Characteristics in WSMO. In: the Middleware for Service Oriented Computing Workshop (MW4SOC 2006), Melbourne, Australia (2006)
15. Cardoso, J., Sheth A.: Semantic e-Workflow Composition. Journal of Intelligent Information Systems, vol. 21(3), pp. 191-225. Springer (2003)
16. Zhang, C., Chang, R.N., Perng, C., So, E., Tang, C., Tao, T.: QoS-Aware Optimization of Composite-Service Fulfillment Policy. In: IEEE SCC. pp. 11--19. IEEE Press (2007)
17. Ma, T., Buyya, R.: Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids. In: IEEE SBAC-PAD, (2005)
18. Truong, H. L., Fahringer, T., Nerieri, F., Dustdar S.: Performance Metrics and Ontology for Describing Performance Data of Grid Workflows. In: IEEE International Workshop on Grid Performability colocated at the IEEE CCGrid. Cardiff, UK (2005)
19. The Semantic Grid Community Portal, http://www.semanticgrid.org/
20. Giallonardo, E., Zimeo, E.: More Semantics in QoS Matching. In: the International Conference on Service Oriented Computing and Applications, SOCA 2007. USA (2007)
21. Bharadwaj, V., Ghose, D., Robertazzi, T. G.: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. Cluster Computing, vol. 6(1), pp. 7--17 (2003)
22. Marchal, L., Yang, Y., Casanova, H., Robert, Y.: A Realistic Network/Application Model for Scheduling Divisible Loads on Large-Scale Platforms. In: the International Parallel and Distributed Processing Symposium. IEEE Press (2005)
23. Charcranoon, S., Robertazzi, T.G., Luryi, S.: Load Sequencing for a Parallel Processing Utility. Journal of Parallel and Distributed Computing, pp. 29--37. Elsevier (2004).
24. Venugopal, S., Buyya R., Winton, L.: A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. Technical Report, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, (2004)
25. Luther, A., Buyya, R., Ranjan, R., Venugopal, S.: Alchemi: A .NET-Based Enterprise Grid Computing System. In: 6th Intern. Conference on Internet Computing. Las Vegas (2005)
26. McGouch, A. S., Afzal, A., Darlington, J., Furmento, N., Mayer, A., Young, L.: Making the Grid Predictable through Reservation and Performance Modelling. The Computer Journal, vol. 48(3), pp.358--368. Oxford University Press (2005)
27. Siddiqui, M., Fahringer, T.: GridARM: Askalon's Grid Resource Management System. In: Advances in Grid Computing - EGC 2005. Lecture Notes in Computer Science, vol. 3470, pp. 122--131. Springer-Verlag, Berlin Heidelberg (2005)
28. Tretola, G., Zimeo, E.: Structure Matching for Enhancing UDDI Query Results. In: International Conference on Service Oriented Computing and Applications, USA (2007)
29. UDDI4J, http://www.uddi4j.org
30. Paolucci, M., Kawmura, T., Payne, T., Sycara, K.: Importing the Semantic Web in UDDI. In: Web Services, E-Business and Semantic Web Workshop, CAiSE 2002. Toronto (2002)
31. Wang Y. Stroulia, E.: Flexible Interface Matching for Web-Service Discovery. In: IEEE WISE, IEEE Press, USA (2003)
32. Jena 2.4, http://jena.sourceforge.net/
33. Pellet Reasoner, 1.3, April 17, 2006, http://www.mindswap.org/2003/pellet/
34. Morante, Q., Ranaldo, N., Vaccaro A., Zimeo, E.: Pervasive Grid for Intensive Power System Contingency Analysis. IEEE Trans. on Industrial Informatics, vol. 2(3), pp. 165--175 (2006)
35. Ranaldo, N., Zimeo, E.: A Transparent Framework for Hierarchical Master-Slave Grid Computing. In: EuroPar06 – CoreGrid Workshop on Grid Middleware, Dresden Germany, Springer (2006)