2

A Framework for Problem-Specific QoS Based Scheduling in Grids

Mohamed Wahib[1], Asim Munawar[2], Masaharu Munetomo[3] and Kiyoshi Akama[4]
[1,2]*Gradute School of Information Science and Technology, Hokkaido University, Sapporo*
[3,4]*Information Initiative Center, Hokkaido University, Sapporo*
Japan

1. Introduction

Grid computing is generally viewed as task assignment to distributed resources. While in practice many factors do complicate this scheduling process (e.g. resource monitoring and discovery, resource failures, resources ownership and policies ... etc), the process is still viewed as resource scheduling according to some criteria. The criteria are referred to as Quality of Service (QoS) attributes in grid computing context. QoS in general are non-functional characters describing a process. QoS attributes are divided into two main groups, namely objective QoS and subjective QoS. Objective QoS attributes are used to specify performance parameters including timeliness, precision, accuracy, security requirements and availability. Subjective QoS attributes, on the other hand, capture application specific policies that govern how an application is treated by the resource manager. In this chapter, objective QoS attributes are referred to as basic QoS attributes, while subjective QoS attributes are referred to as application-specific QoS attributes. In comparison to application-specific QoS attributes, a huge legacy of scheduling with the basic QoS attributes exists. This chapter proposes adopting application-specific QoS attributes to define new criteria other than the basic ones to enhance the preference of which resources to choose in the task assignment process as the QoS attributes are application-specific (e.g. a Grid application involving service tasks that retrieve images from a service resource could have the color depth and resolution as QoS attributes).
For grid computing to go beyond the basic QoS attributes the following challenges need to be addressed: a) A method to formally define new application-specific QoS attributes. b) A method to measure the QoS attribute fidelity of a specific resource. c) Scheduling the tasks over the resources taking into consideration the defined application-specific QoS attributes. This chapter addresses these challenges and proposes a framework that starts from defining new application-specific QoS attributes until the tasks are executed over the resources. As for the first challenge, a system for defining QoS using a formal language is inspired from Canfora et al. (2006) which proposes service composition binding driven by application-specific QoS. The domain of Canfora et al. (2006) is different from the work in this chapter. This is because the authors are concerned with the composition of services that are initially defined as an abstract workflow, where for each service in the workflow a set of service providers are defined. Then the scheduler/binder makes a concrete workflow at which each service in the abstract workflow is bind to a service provider. Canfora et al. addressed how application

specific QoS attributes can be defined and aggregated to be used for scheduling. The basic idea is implementing a QoS aggregation function definition interface and a QoS definition language, so the administrator through a simple QoS aggregation function editor could define new QoS(s). The QoS definition in this framework here builds on and extends the work done by Canfora though it is defined for a different discipline (i.e. SOA). It is of no-awkwardness to use a mechanism that was originally defined in SOA (Service Oriented Architecture) service workflows to be used in grid computing. As grid computing and SOA have many tangency points. And this point in particular could be considered a convergence point between grid computing and SOA.
At the first glance, scheduling tasks in such an environment seem to add more complexity to the traditional problems of task assignment. However, the definition of task associated QoS attributes is utilized to enrich the task assignment process. The Chapter illustrates the framework, describes the design and implementation of the framework and finally demonstrates a use case to emphasize on the functional efficiency of the framework. The rest of this paper is organized as follows; the next section is a brief review on subjective QoS representation in service environments. Section 3.discusses the proposed framework. Section 4. discusses the experiments that were conducted using the proposed framework. Finally section 5.concludes and adds insight to future work.

2. Subjective QoS representation in service environments

Quality of Service is an overarching term covering different parts of end-to-end service quality. The general definition of QoS provided by the International Telecommunication Union (ITU)Recommendation (1994) is that QoS is Şthe collective effect of service performance, which determines the degree of satisfaction of a user of the serviceŤ. Different people and communities nevertheless interpret QoS differently, and at least the following viewpoints of QoS can be distinguished: QoS requirements of a user, QoS perceived by the user, QoS offered or planned by a provider, and QoS delivered or achieved by the provider. We are discussing QoS in the userŠs point of view. There are two main aspects of QoS: subjective and objective. Subjective QoS essentially is the userŠs overall perception of service quality, that is, it is the userŠs opinion whether a service is working satisfactorily or not. Subjective QoS is often difficult to be specified with objective measures, at least in a way meaningful for users, and thus user-perceived quality is often expressed also non-technicallyBouch et al. (2000). Objective QoS then refers to the technical aspects of QoS, and can be specified with quantitative measures. Grid Application QoS parameters are not necessarily applicable to express subjective QoS, since a user has a high-level perspective over application performance, rather than an in-depth conception of details of the underlying implementation and operation of the service. Therefore, application quality and its variation need to be expressed in terms that describe user-perceivable effects, instead of their causes in the end-to-end functionality. It should be noted also that subjective application quality deterioration is not solely caused by operational QoS fluctuations, but is attributable to numerous other factors, including characteristics of the ongoing task (e.g. urgency), applicationŠs incompatibility with the service provided, application or protocol malfunction, disturbing factors in usage environment (e.g. faulty equipment), and so forth.
Two principal approaches for subjective application quality assessment exist: user study methods and objective measurements. The user study methods include, e.g. Mean Opinion Scores (MOS), continuous assessment, Task Performance Measures (TPMs), and qualitative methodsBouch et al. (2001). Objective measurements, on the other hand, rely on measurement

of some application quality metric(s) (e.g. Peak-Signal-to-Noise-Ratio (PSNR) for video)Wang et al. (2003). Generally, MOS are used on a wide scale to collect the subjectsŠ opinions of the experienced service quality. In short, MOS enables performing controlled assessment of subjective QoS with untrained subjects and controlled levels of qualityBouch et al. (2001). The method employs a 5-point scale, according to which subjects judge the experienced quality after conducting a task. The given ratings are then averaged across the subjects to get the final MOS.
This paper proposes a problem-specific QoS based scheduling in grids. To determine the optimal choice of services/applications, the approach needs to estimate the subjective QoS. This can be done using some aggregation formulae. The formulae define aggregation functions for defined QoS attributes. While this approach is far from being general (i.e. needs to be customized for each application as the QoS aggregator can operate differently). Yet, we argue that the proposed approach gives a methodology for quantifying subjective QoS and thus is more reliable compared to the user study methods.

3. The proposed framework

3.1 Defining and evaluating subjective QoS attributes

In order for the framework to define and evaluate subjective QoS attributes we developed a language that permits to specify a new QoS attribute. As mentioned in the introduction, the language proposed here extends the work by Canfora et al. (2006) which defined QoS aggregation formulae for each pair QoS attribute–workflow construct. In most cases, the aforementioned aggregation formulae are cabled in the optimization algorithm the binder is using. Therefore, it is necessary to provide a language and a tool to specify aggregation formulae, and to allow the schedular to interpret such formulae for estimating the QoS of the grid services. The method by which the framework defines and evaluates subjective QoS attributes is as follows:
QoS definition language: For a language to permit specifying new QoS attributes, two things are required; type and scale. The type can be only primitive types (integer, real and Boolean) as in WSLA+ languageNepal et al. (2008), or include collection types (i.e. a set constituted of sets of atomic values) as Canfora did. The scale limits the set of admissible operations. The language developed by Canfora includes the scales required for our framework, so no change is required in this part. The point of difference here is the set of operators and functions inherited form the Object Constraint Language (OCL) Warmer & Kleppe (2003) that is used by Canfora. This is due to using those operators and functions in computing overall workflow QoS, while in the proposed framework accepts both inter-dependent task (i.e. workflows) and individual tasks. In the case of workflow tasks, the operators and functions defined by Canfora are sufficient, while in the case of individual tasks the operators are not used due to tasks independency. The next step is to show how the QoS formula specification is supported by a guided editor and type-checker.
QoS aggregation: The QoS aggregator introduced by Canfora et al. (2006) was implemented in Java using the Java Compiler (JavaCC) parser generator, while for the GUI, JSP was used. The aggregator was adopted here by including the aggregator in a Vine portlet and adding it to the gridsphere portal. Modifications were done to drop operators for individual tasks as mentioned earlier, associate the newly defined Qos attribute to any of the task types registered by the administrator and finally adapting the original JSP to work with the Adobe Flex GUI used in gridsphere 3.1. The aggregator include three basic modules; *QoS aggregation function editor* a portlet that the administrator can use to define new QoS attributes and their aggregation formula (see figure1), *Type checker* used at design time for verifying

integrity of the aggregation formula, *QoS formula interpreter* that at run time evaluates QoS for a possible workflow/individual task assignment.

An important point to note, the model for determining the expected QoS from each resource is executed as agents hosted on the resources. Note that these agents are also used for resource monitoring and discovery. As for the scheduling process, several projects in the literature attempted multiple QoS job scheduling. Generally speaking, these projects addressed mainly the basic QoS attributes in their work; cost, time, availability and fidelity. Those basic QoS attributes are agreed upon in literature to be the metrics for job scheduling in grids. Application specific QoS attributes were not considered in any of them as no service level presentation of resources provided features into which application-specific QoS attributes could be mapped. Li & Li (2007); Li et al. (2007) defined the 4 basic QoS attributes as utilities and identified an Integrated Utility Function (IUF) to be used for scheduling. A slight difference between them is that Li & Li (2007) used an iterative scheduling technique by separating the task agents and resources agents. Doğan & özgüner (2006) proposed a static scheduling algorithm that uses utility functions for scheduling meta-task with QoS requirements in heterogeneous computing systems. The main aim of this project was to provide resource transparency and not include application-specific QoS attributes. Kim et al. (2006) did not define a scheduling method, yet it defined for the basic QoS attributes a flexible multi-dimensional QoS performance measure that could be later aggregated to be used for scheduling. A notable point is that the authors defined a method that is theoretically applicable to model any new QoS attribute by representing the QoS attribute by a feature vector. This approach is generic but if used will shift the entire load to the application developer.

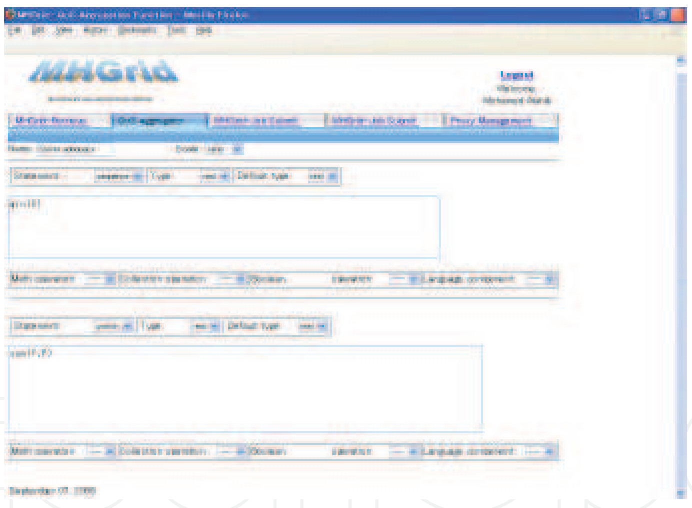


Fig. 1. QoS aggregation function definition interface

3.2 Problem formulation

Suppose that there are n independent users and user i is associated with task T_i, let $T = \{T_1, T_2, ..., T_n\}$ denote the set of n independent tasks where task T_i is assigned to resource R_j and $R = \{R1, R_2, ..., R_m\}$. Further, R_j can be composed of v resources, where $1 \leq v \leq m$. The process of QoS-constrained task scheduling could be summarized in three steps. Step one is Resource Discovery, which is finding available resources complied by QoS constraints and generate a list of resources. Step two is Resource Planning, which involves selecting the optimized resources from the available resources list according to the scheduling strategy

satisfying user's QoS constrains. The $\Gamma : T \rightarrow R$ denoting the matching function is a NP complete problemChristensen (2007). The third step is Task Execution. The tasks are scheduled or mapped onto selected resources to be executed.
In the proposed framework, QoS attributes are described as utility functions. So the integrated utility function is the accumulation of all QoS attributes utility functions. The integrated utility is considered as the objective function of the scheduling algorithm to drive the scheduling of resources and optimizing the task execution with maximum utility. A very important point here is that utility functions have long been used in QoS constrained scheduling, but for this case a new factor requires special handling. The new factor is having different QoS attributes defined according to underlying application. Thus, the integrated utility function includes a set of mutually exclusive utility functions corresponding to the QoS attributes. The next step, after defining the integrated utility function, is to compute the value of the utility u_k, $1 \leq k \leq d$ where d is the total number of QoS constraints defined by the administrator, what we will call the *dimension of QoS*. The method used for computing the utility functions uses switch constructs as in Anselmi et al. (2007) to iteratively update the selection probabilities of the selected resources. For each task T_i a *decision matrix* $Q = \left(q_{ij}\right)_{m \times d}$ is created, among it, m represents the number of resources that can host T_i, d represents the dimension of QoS attributes considered by the this type of task. Q matrix is not used directly, Q is normalized to make the *normalized matrix* $P = \left(p_{ij}\right)_{m \times d}$, $j = 1, ..., m, k = 1, ..., d$, where normalizing is done as follows:

$$p_{jk} = \begin{cases} min_j \left(q_{jk}\right) / q_{jk} \text{ if } u_k \text{ is optimally minimized} \\ \\ q_{jk} / max_j \left(q_{jk}\right) \text{ if } u_k \text{ is optimally maximized} \end{cases}$$

Supposing $\omega = (\omega_1, \omega_2, ..., \omega_d)$ is attribute weighting vector, Then the integrated utility function for the evaluation of the selected resources can be defined as:

$$U_{ij}\left(\omega\right) = \sum_{t=1}^{S} \left(C_t \sum_{l=x}^{y} p_{jl} \omega_l \right)$$

where S is the number of subjective QoS attributes defined by the administrator to be used in the system, C_t is a Boolean constant having value 1 only for the QoS attributes subset of the task scheduled (i.e. QoS divided to subsets having a mutually exclusive relation.). x and y denote the start and end of the QoS attributes subset from the set of all QoS attributes defined. By sorting and computing $U_i\left(\omega\right)$, the best resource can be selected. Weights for utility functions in the mentioned related work are calculated by maximizing the deviations in utiliy values. In our case the QoS attributes are defined by the administrator according to the application. Therefore, the default weights are defined by the administrator according to the system used. The end user could calibrate the weights according to his interest in which QoS attributes significant for the job he is about to submit.

3.3 Scheduling using quantified subjective QoS attributes

The scheduling process is basically viewed as a combinatorial optimization problem for the integrated utility function. Several approaches are used in the literature to solve this optimization problem, most approaches depend on a heuristic algorithms. Generally, GA (Genetic Algorithms)Jong (1992) based heuristic algorithms do not impose constraints on the linearity of the QoS composition operators, so they are considered the best option. However,

the prime focus in this work is to represent and utilize subjective QoS attributes. Thus, a simple rank-based algorithm is used to determine the focus on the aforementioned process
As shown in the previous section, QoS attributes are described as utility functions, The integrated utility function aggregating all utility functions is regarded as an objective function of the scheduling algorithm to drive dynamic scheduling to the resources and optimizing the task execution with maximum utility by accumulating all QoS attributes utility functions. Figure2 shows the algorithm used for scheduling. One important point to note here, the algorithm is designed in a central way, which leads to weak scalability. Yet, the design of centralized scheduling algorithms is an important step towards developing more complex decentralized scheduling algorithms.

```
1- For all $T_i$ if $T_i$ is a workflow or nested tasks, then $T_i = (T_{i1}, T_{i2},\ldots, T_{im})$,
     $m \geq 2$. For each  $T_{i2}$  go to step 1.
2- Use $T_i$ category to create available resources list that can host $T_{i.}$
3- For all $k \in m$, evaluate QoS attributes of all services in service list.
4- Create decision matrix $Q_i$.
5- Compute the integrate utility function of multiple QoS attributes $U_i(\omega)$.
6- Sort resources according to $U_i(\omega)$.
7- Execute $T_i$ on the selected resource.
```

Fig. 2. Algorithm used for scheduling

4. Use case and results

Task type	QoS attributes	Individual		Nested		Workflow		
Opimization algorithms as services (Service tasks)	* *Solver to problem adequacy* * *Parallel pattern* * *Fidelity*	Small	50	Small	50	Random size, 20 tasks ↓	🟪	🟩
		HPC	20	HPC	20		🟪	🟩
		HTC	10	HTC	10		🟪	🟩
Optimization algorithms as computational tasks	* *Latency tolerance* * *Data accuracy*	Small	50	Small	50	🟥	🟪	↑ Random size, 20 tasks ↓
		HPC	20	HPC	20	🟥	🟪	
		HTC	10	HTC	10	🟥	🟪	
Data tasks	* *Priority*	Small size staging	50	N/A	N/A	🟥	↑ Random size, 20 tasks	🟩
		Large size staging	20	N/A	N/A	🟥		🟩
		Large size flow	10	N/A	N/A	🟥		🟩

Table 1. Test bed configuration

This section presents the approach at work over Meta Heuristics Grid (MHGrid): a service oriented grid application offering meta heuristics based solvers for global optimization problems. MHGrid is designed to offer optimizing algorithms as grid services. The experiments were designed to cover possible task types in the framework. For the service tasks, individual and nested tasks of different sizes were introduced. The size is controlled by the problem length that the algorithms should solve. Three specific QoS attributes were defined; Solver to problem adequacy, the parallel pattern used and the fidelity or quality of output. These attributes are explained in details in Munawar et al. (2008). The same applies for computational tasks but with two different QoS attributes. Long timeouts were added in the solvers to test the latency tolerance with the solvers running in parallel mode. The other QoS

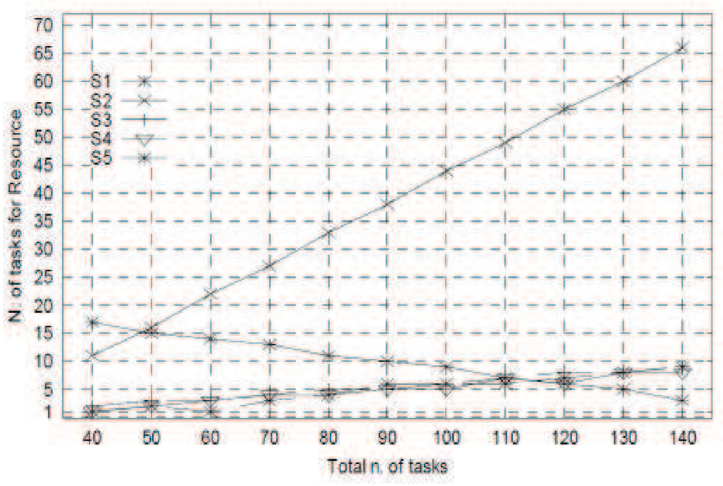


Fig. 3. Tasks assigned to each service varying the total amount of tasks

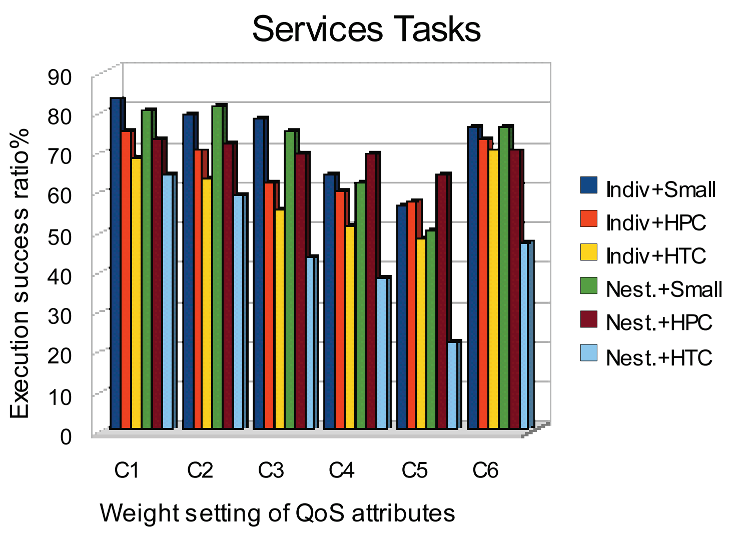


Fig. 4. The effect of services tasks weight vector on execution success ratio

attribute is data accuracy which is measured by round off errors resulting from float-point operations intentionally introduced to the solvers. As for the data tasks, dummy files where used to be transferred via GridFTPFoster (2006). The file sizes varied from 100MB to 20TB to represent different task sizes. One QoS attribute (i.e. priority) was defined for the data tasks where each task was assigned priority rank for the scheduler to attempt to meet the priority requirement of those tasks. In addition to that, workflow tasks for different task types where also created by the workflow portlet to be tested. Finally two basic QoS attributes (make span time and cost) were used for all tasks along with the specific QoS attributes defined for each task type. Table 1 shows the configuration used in the experiments. Note that tasks were introduced randomly with 20 tasks a time. Figure 3 shows the distribution different task sets over the resources. Each task was assigned to one of the five services (S1,S2,S3,S4,S5) overlaying the computational and data requirements.
The experiments were executed on a grid having a dedicated 64 core mini-cluster with 2 x AMD Opteron 2.6 GHz Dual Core 64 bit processors and 2GB RAM for each node, the grid also has 2 dedicated servers each having a 2 x Xeon 2.8 GHz Dual Core with 2GB RAM. Execution

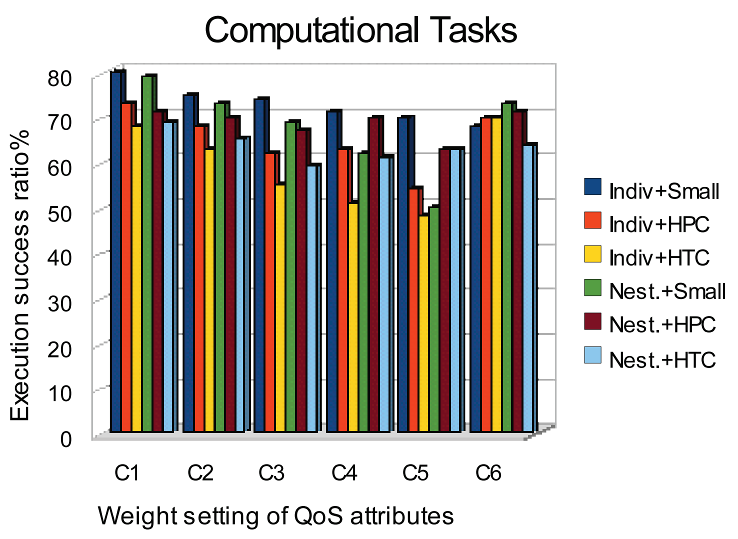


Fig. 5. The effect of computational tasks weight vector on execution success ratio

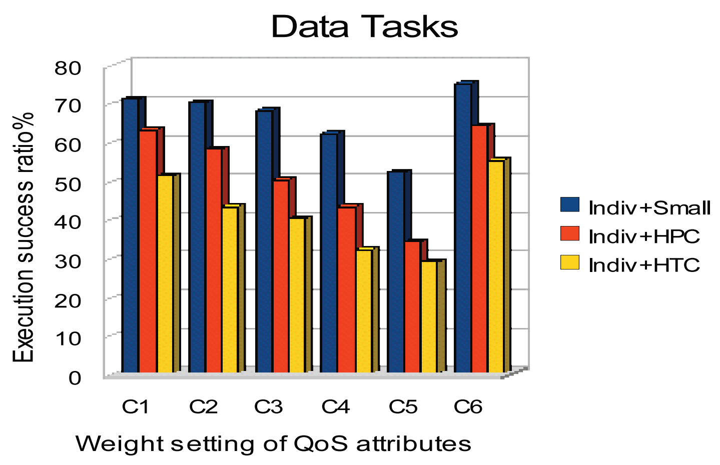


Fig. 6. The effect of data tasks weight vector on execution success ratio

success ratio is the metric used in the experiments to measure the framework's ability to host and schedule variant task triplets. Other metrics such as resource utilization and task waiting time are important as well, but due to space limitation, results in terms of execution success ratio only are illustrated to evidate that a system can function whilst benefiting from subjective QoS attributes. Figures 4, 5 and 6 show the execution success ratio for the individual and nested tasks on different task types/sizes. The x-axis in all three figures is a configuration state for the QoS attributes considered. The last state (i.e. C6) is the plain state where all weights for task-specific QoS attributes are set to zero and only the basic time and cost QoS attributes with weight = 0.5 are considered. For the other states (i.e. C1 to C5) the weights are having values with an increasing mean from C1 to C5 and also an increasing standard deviation to represent variant weight vector settings. Note that upon moving from C1 to C5 the success rate tends to decrease which is normal as the more the mean of the weight vector increases the more the scheduling process relay on QoS other than the time and cost. This consequently minimizes the set of resource candidates for each task and causes less success rate. Another point to note is that for figure6, C6 gives much better performance than other weight settings.

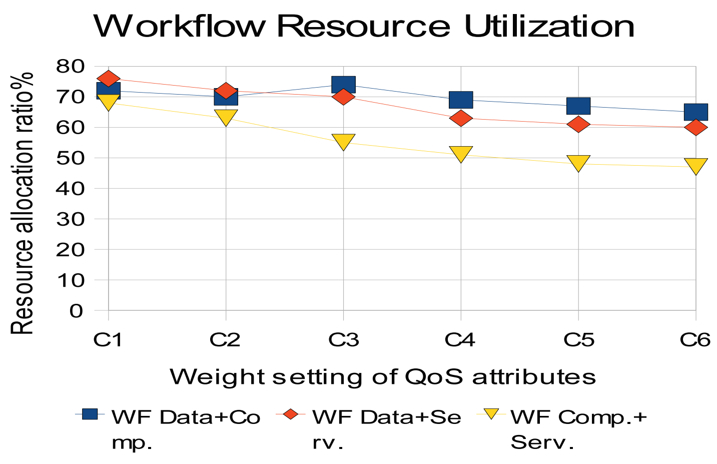


Fig. 7. Resource utilization for workflows with variant tasks weight values

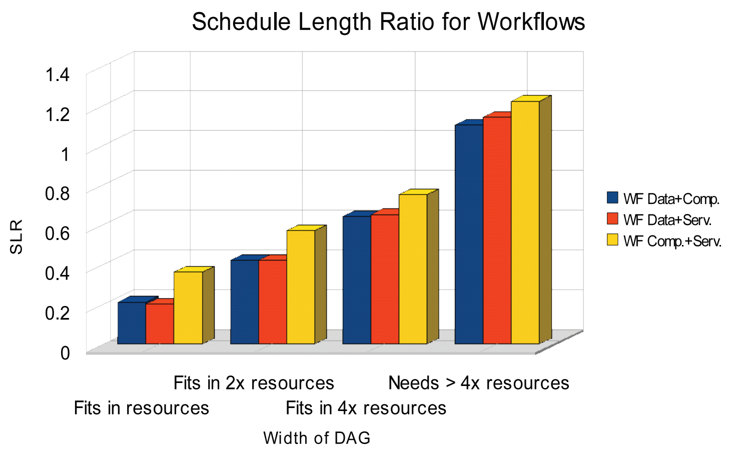


Fig. 8. SLR for the workflows with different DAG width

This is because the *priority* QoS defined for data tasks highly effects the scheduling process making it mostly depending on what the user wants. Figure7 shows the resource utilization for workflow tasks. Again the change in weight value does not have a significant effect on the resource utilization and this is because the workflows enforce an order for task execution on the system. Figure8 shows another aspect of the workflows. The figure shows the Schedule Length Ratio (SLR) for the workflows that were created with different DAG widths.

5. Conclusion and future work

This chapter introduced a system for quantifying subjective QoS attributes and using them for assigning tasks in a grid application. First a close-up was given to the representation of subjective QoS attributes in service-based environments. The framework was inspired by a system of application-specific QoS attributes in service composition. While the domain is apparently different, the flexibility offered by the service composition in defining subjective QoS attributes is of great relevance to the requirements of establishing an enviroment that is not only relying on conventional QoS attributes in job assignment. The framework allows the administrator to define subjective QoS attributes through a QoS aggregation function editor enclosed in a portlet. After the administrator registers the categorized resources, the end user can start submitting tasks to the system. With the user unaware, the system schedules the tasks on behalf of the user, and assures the use of the subjective QoS attributes to assign the tasks to the most appropriate resources. Experiments were conducted to assure that the framework is functioning as designed. Many points are promising to be considered as future work. Making the best use of subjetuve QoS attributes by incorporating a SLA mechanism is pivot point. Expanding the function aggregation editor to make the framework generic is as well a challenging task.

6. References

Anselmi, J., Ardagna, D. & Cremonesi, P. (2007). A qos-based selection approach of autonomic grid services, *SOCP '07: Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches*, ACM, pp. 1–8.

Bouch, A., Kuchinsky, A. & Bhatti, N. (2000). Quality is in the eye of the beholder: meeting users' requirements for internet quality of service, *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 297–304.

Bouch, A., Sasse, M. A., Demeer, H. & Bt, W. (2001). Of packets and people: A user-centered approach to quality of service.

Canfora, G., Penta, M. D., Esposito, R., Perfetto, F. & Villani, M. L. (2006). Service composition (re)binding driven by application-specific qos, *ICSOC*, pp. 141–152.

Christensen, T. V. (2007). Heuristic algorithms for NP-complete problems.

Doğan, A. & özgüner, F. (2006). Scheduling of a meta-task with qos requirements in heterogeneous computing systems, *J. Parallel Distrib. Comput.* 66(2): 181–196.

Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems, pp. 2–13.

Jong, K. A. D. (1992). Are genetic algorithms function optimizers?, *PPSN*, pp. 3–14.

Kim, J.-K., Hensgen, D. A., Kidd, T., Siegel, H. J., John, D. S., Irvine, C., Levin, T., Porter, N. W., Prasanna, V. K. & Freund, R. F. (2006). A flexible multi-dimensional qos performance measure framework for distributed heterogeneous systems, *Cluster Computing* 9(3): 281–296.

Li, C. & Li, L. (2007). Utility-based qos optimisation strategy for multi-criteria scheduling on the grid, *J. Parallel Distrib. Comput.* 67(2): 142–153.

Li, Y., Zhao, D. & Li, J. (2007). Scheduling algorithm based on integrated utility of multiple qos attributes on service grid, *GCC '07: Proceedings of the Sixth International Conference on Grid and Cooperative Computing*, IEEE Computer Society, Washington, DC, USA, pp. 288–295.

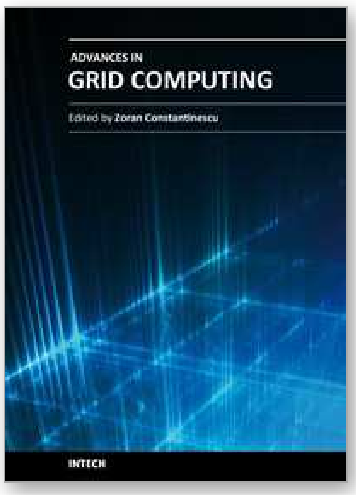
Munawar, A., Wahib, M., Munetomo, M. & Akama, K. (2008). *Linkage in Evolutionary Computation*, Springer Berlin / Heidelberg, chapter Parallel GEAs with Linkage Analysis over Grid, pp. 159–187.

Nepal, S., Zic, J. & Chen, S. (2008). Wsla+: Web service level agreement language for collaborations, *scc* 2: 485–488.

Recommendation, I.-T. (1994). Terms and definitions related to quality of service and network performance including dependability, *Technical Report E.800*.

Wang, Z., Banerjee, S. & Jamin, S. (2003). Studying streaming video quality: from an application point of view, *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, ACM.

Warmer, J. & Kleppe, A. (2003). *The Object Constraint Language: Getting Your Models Ready for MDA*, Addison-Wesley.



Advances in Grid Computing
Edited by Dr. Zoran Constantinescu

ISBN 978-953-307-301-9
Hard cover, 272 pages
Publisher InTech
Published online 28, February, 2011
Published in print edition February, 2011

This book approaches the grid computing with a perspective on the latest achievements in the field, providing an insight into the current research trends and advances, and presenting a large range of innovative research papers. The topics covered in this book include resource and data management, grid architectures and development, and grid-enabled applications. New ideas employing heuristic methods from swarm intelligence or genetic algorithm and quantum encryption are considered in order to explain two main aspects of grid computing: resource management and data management. The book addresses also some aspects of grid computing that regard architecture and development, and includes a diverse range of applications for grid computing, including possible human grid computing system, simulation of the fusion reaction, ubiquitous healthcare service provisioning and complex water systems.