QoS within Business Grid Quality of Service (BGQoS)

Reda Albodour, Anne James, Norlaily Yaacob

Faculty of Engineering and Computing, Coventry University, Coventry, CV1 5FB, UK

Reda.Albodour @coventry.ac.uk, A.James@coventry.ac.uk, N.Yaacob@coventry.ac.uk

Abstract

Differences in domain QoS requirements have been an obstacle to utilising Grid Computing for main stream applications. While the resource could potentially provide potentially vital services as well as providing significant computing and storage capabilities, the lack of high level QoS specification capabilities has proven to be a hindrance. Business Grid Quality of Service (BGQoS) is a QoS model for business-oriented applications on Grid computing systems. BGQoS defines QoS at a high level facilitating an easier request model for the Grid Resource Consumer (GRC) and eliminates confusion for the Grid Resource Provider in supplying the appropriate resources to meet the GRC requirements. It offers high level QoS specification within multi-domain environments in a flexible manner. Employing component separation and dynamic QoS calculation, it provides the necessary tools and execution environment for a scalable set of requirements tailoring to specific domain demands and requirements. Moreover, through reallocation, the model provides the insurance that all QoS requirements are met throughout the execution period, including migrating tasks to different resources if necessary. This process is not random and adheres to a set of conditions which ensures that task execution and resource allocation happen when and in accordance with execution requirements. This paper focuses on BGQoS' flexibility and QoS capability. More specifically, the concentration is on core operations within BGQoS and the methods used in order to deliver a sustained level of QoS which meets the GRC's requirements while being versatile and flexible such that it can be tailored to specific domains. This paper also presents an experimental evaluation of BGQoS. The evaluation investigates the behaviour and performance of the separate operations and components within BGQoS, and moreover, it presents an investigation and comparison between the different operations and their effect on the full model.

Keywords: QoS, Business Grid, Resource Allocation

1. Introduction

The emergence of Grid Computing as a mainstream solution [1] has allowed the progression and development of a new generation of applications that utilise resources on demand. Grids are systems that provide the user with seamless access to a variety of resources, such as CPUs, storage space, data and instruments. The Grid computing field is the result that has emerged from a series of evolutionary steps in computing [2] with each providing a major advancement, allowing users to solve more complex problems and gain otherwise unattainable results. This evolution started with the single user model, to Massively Parallel Processors (MPPs), to clusters [3] to distributed systems and finally to Grid computing. Recently, large companies such as SUN, IBM and Amazon have been providing Grid solutions by providing resources and services to third parties.

Giving the GRC control over their requests is essential in business-oriented environments and guaranteeing that their requests are delivered throughout the period in which they utilise specific resources is vital. This includes being able to request specific resources, set resource requirements or Quality of Service (QoS) requirements and obtain guarantees that these requirements are met according to their request throughout the duration within which there is an association between the user and the resources. Quality can be defined from the following three different perspectives and views:

- **Quality of functionality:** Quality is considered in terms of the amount of functionality that a service provider offers to the customers. It characterizes the design of an entity and can only be measured by comparing it against others offering similar functionalities.

- **Quality of conformance:** Quality is considered in terms of meeting user requirements and providers meeting their commitments and specification. Quality as conformance, which can be monitored for each service individually, usually requires the users' experience of a service in order to measure the 'promise' against the 'delivery'.

- **Quality of reputation:** Quality is considered in terms of the users' perception of a service in general. This perception is developed gradually over the time of a service provider's existence. Quality as reputation can be regarded as a reference to a service provider's consistency over time in offering both functionality and conformance qualities, and can therefore be measured through the other two types of qualities over time.

In BGQoS, Quality of Conformance has been chosen which sees QoS as meeting user requirements and specifications. For example, if a user requires a resource with computational power equal to *x* and the resource provider offers a resource that delivers the amount of computational power required, i.e. computational power $\geq x$, provided that the resource does so throughout the time the resource is dedicated to this user, then the resource can be said to have met the request, or conforms to the request. Moreover, each particular domain within the mainstream environment to which the user belongs, will have its own set of QoS requirements and parameters that apply to those domains' applications. For example, in the field of meteorology, compute intensive resources are required to deliver specific levels of power for a significant amount of time to run weather pattern simulations and achieve the required results in the shortest period possible. In the medical field compute-intensive resources are needed for image processing [4]. In other fields, for instance E-commerce, data intensive processing may be more of a priority. In EScience too, data-intensive processing is needed to handle the ever increasing quantities of data [5]. However, while each domain has specific requirements, there is a case for carrying the parameters across domains and establishing an arrangement for ensuring that the QoS carried between domains conform to the same definition. Resource information accessibility allows cross-domain requirement specifications and provides the platform for locating the appropriate resources that meet QoS parameters requirements submitted by the user. Current, up-to-date and accurate information relative to each resource ensures that resources selected are offering the level of QoS that is requested by the user and provides the base for creating a working relationship between user and resource provider.

Grid computing research has not produced a comprehensive and flexible approach to QoS requests and resource allocation, which supports different types of Grids and applications. Current Grid technology is diverse with an inclination for adopting a service-oriented architecture [6] that supports and provides commercial, business-oriented and mainstream services to different domains. However, most current efforts address specific domains such as bioinformatics [7] producing specific solutions tailored for applications within those domains and a solution that cannot be carried across to another domain easily. BGQoS was designed to find a solution to the problem of lack of flexibility and lack of QoS support in Grid Systems. Essentially, it is proposed in line with the assumption that applications from different domains such as, education, engineering and medicine require QoS guarantees and that the requesting of those guarantees is carried out in an efficient manner at a high-level.

The paper is structured as follows. Section 2 discusses related work in the area. Section 3 describes the BGQoS approach to effect such delivery, presents the QoS definitions used and discusses BGQoS flexibiity. More detail on the capabilities of BGQoS is provided in section 4 where the implementation of the matching approach is covered including how the various parameters and constraints are used and how resources are ranked. Further presentation of how BGQoS provides flexibility is given in section 5, here concentrating on the multi-tier user interface and resource broker selection. Section 6 discusses reallocation which is a necessary run-time process to maintain QoS when problems occur with allocated resources. Our experimental results are presented in section 7. Section 8 offers conclusions.

2. Related work:

Recently, with the Grid expanding towards different commercial domains [8] the QoS of Grids has become an active field of research. Golconda and Ozguner compared different QoS based scheduling efforts [9] while Al-Ali [1] proposed the Grid QoS Management model (G-QoSM). Their model uses service abstraction in the Open Grid Services Architecture (OGSA) and extends it for QoS properties. G-QoSM reserves quantitative resources, such as CPUs, then allocates and monitors these resources, independently. Another, reservation based approach is presented by [10] where they used the alternate offer protocol to make advance reservations. In both of these approaches, it is assumed that all resources involved understand the reservation and negotiation protocols. GARA [11], the General-purpose Architecture for Reservation and Allocation allows users and applications to manipulate reservation of different resources in uniform ways. It provides capabilities to create, modify, bind and cancel reservations. Moreover, it supports flow specific end-to-end QoS specification and resource monitoring.

A Selection Manager is used by Yu and Lin [12] as a solution for the service selection problem in complex Grid services with multi-QoS requirements. The Selection Manager can be implemented as a combinatorial model or a graph based model. An heuristic is proposed for the combinatorial model based on the algorithms used for solving the multi-option, multi-dimension knapsack problem. This technique is also used by Wieczoreka in [13], who propose an approach for modelling scheduling problems as an extension to the knapsack problem solution. The graph model, on the other hand, is based on the algorithm proposed as a solution to the multi-constraint optimal path problem [12]. The main objective is maximizing the utility of the system. To achieve this, a utility function is proposed and the algorithm's attempt at maximising this function is intended to increase user satisfaction. Their approach is specifically tailored for the user, without taking the resource provider into consideration.

PBS [14], LSF [15], Grid Engine [16] and Condor [17] have queuing systems that can be used, efficiently, for delivering a single specific requirement. If all tasks and their requirements are known in advance, a static approach [18] can create a full schedule for all the tasks at the same time meeting multiple user requirements. Other static approaches include CCS [19] and GORBA [20]. They both use advance reservations of resources that schedule a sequence of related tasks. However, as they are static approaches, a complete recreation of the schedule is required, if there is a change in resources while executing the sequence of tasks. For example, if a resource fails while execution is in commencement, the whole schedule must be recreated from the beginning.

Triana [21], Askalon [22], Jopera [23], eXeGrid [24] can be used in the development of tools, languages and interfaces for the composition of workflows, while Pegasus [25] concentrates on supporting the creation of workflows for large scale Grid applications. Taverna [26] is a system which is concerned with semantic Grid workflows and is implemented as part of the ^{my}Grid [27] project. The aim is to develop sophisticated middleware technology tailored for bioinformatics [7]. It provides fault tolerance solutions and implements a Graphical User Interface (GUI) for the creation and representation of workflows. An overview of workflow systems and features for E-Science is provided in [28].

The efforts described above have achieved a number of advances in Grid Computing in general, and in resource operations and QoS in particular. The approaches fall into the categories of reservation, system optimisation, scheduling or workflow. We can observe that the currently existing QoS efforts in Grids concentrate on local optimal QoS scheduling. Although these approaches take user information and resource information into consideration when allocating resources to different tasks, the concentration is on local resources. Issues include the assumption of a reservation compact between broker and resource, time required in finding optimal solutions for complex and dynamic combination, lack of dynamism in static schedules when resources fail and assumed knowledge of users in manipulation of workflows. BGQoS addresses these problems by using a tiered system which supports varying levels of users, allowing specification of QoS resource requirements in various ways including: as first choice and second choice requirements; as time requirements; as cost requirements; and as combinations of all these. BGQoS also supports search of global resource brokers, reallocation in case of problems with allocated resources, selection of resources based on reputation, both QoS (Quality of Service) and BE (Best Effort) consumers and the use of a tolerance ratio. These facilities allow BGQoS to provide varying levels of flexible support for varying levels of users (who may differ in terms of expertise and requirements) without being constrained to find optimal solutions or use only Grids where reservation compacts exist. The difference between QoS and BE consumers is that a QoS consumer has specified some required QoS whereas a BE consumer has just asked for the system to complete the job according to its best effort. The two types of consumer have different service level contracts.

3. BGQoS

In this section we explain the overall approach of BGQoS, present the QoS definitions that are used by BGQoS to effect appropriate QoS delivery and discuss BGQoS flexibility.

3.1. The BGQoS Approach

The GRC is the focal point which steers the resource discovery phase and is also the main guideline for resource selection. This is done when the resource discovery phase yields a list of resources or resource sets that could potentially provide the GRC with their requirements. However, BGQoS allows the GRC to specify a set of constraints within their description when a request is made. These criteria, along with other domain specific criteria that could be introduced according to each domain and their requirements, are used for ranking resources for selection purposes. The highest ranked list is to be chosen, the rest of the potential lists are stored in databases that can be accessed for reallocation and migration purposes, if necessary. An approach for reallocation using integrated enhanced stop/start and resource swapping techniques is used if there is resource failure to contend with and/or there is the situation where the level of specified QoS is not met. This is detected through monitoring the resources and task execution until application completion and session termination.

The resource allocation process within Grid Computing has been generally split into two distinct phases: (1) resource discovery or resource selection; and (2) resource allocation. However, when BGQoS was designed and implemented, a separation between the resource discovery and resource selection operations has been implemented, each defined to be associated with a distinct operational phase. This distinction between the two phases has allowed for the introduction of a more accurate resource operational model. In other words, the approach used within BGQoS decouples the resource specification and discovery process from resource selection, introducing an explicit method for resource selection. This has allowed BGQoS not only to locate the appropriate resources but select the most appropriate in order to execute tasks and applications.

Once the appropriate resources have been located, BGQoS executes applications accordingly, sending tasks to the resources to which they are assigned. Allocation and execution management components are responsible for the actual task execution that is carried out with the resources that are selected. This is followed by the creation of an appropriate execution environment, which BGQoS accomplishes by first submitting one of its components called the *Task Launcher* to the resources. The Task Launcher's concept was designed such that it can run in any environment without modifications or additions necessary for its operation. It is responsible for the input and output files for the application, as well as keeping track of the number of locations where the resources are located, the location of the input files and maintaining a unique $\mathrm{Task_{ID}}$ for each task until its completion where the output files are also its responsibility.

The purpose of BGQoS is not only to provide the GRC with the QoS requirements that they request when they submit their mainstream application but to sustain the level of QoS that was promised. The premise that both parties will adhere to what they agree upon is documented in a contract or agreement that is initiated by the GRC, received by the model and offered by the GRP (Grid Resource Provider). If there is a violation of the contract, which might occur for multiple reasons, including resource failure and performance degradation, middleware malfunctions and user errors [29, 30] then the *Rescheduler* in BGQoS is activated and the reallocation process is initiated.

3.2. QoS definitions within BGQoS

The flexibility of BGQoS means that there could be many QoS requirements and different types of resource requests. Moreover, the GRC (Grid Resource Consumer) may wish to create a budget (Cost) and deadline (Time) constraint. Table 1 outlines the QoS related definitions that are used within BGQoS.

Table 1: QoS definitions

$\mathbf{QoS_{parameter(s)}}$
A $QoS_{parameter}$ is defined as a specific GRC requirement, input when submitting an application. BGQoS supports two types of resources, mainly Computing and Storage resources; therefore, most of the parameters that are mentioned hereafter are related to these two types. However, it is important to point out that the model can be expanded easily to accommodate other types of resources as required and relevant to different domains.
$\mathbf{QoS_{constraint(s)}}$
A $QoS_{constraint}$ is defined on of a set of conditions that need to be met once there are resources that can deliver the level of QoS that is specified by the request for $QoS_{Parameters}$. These constraints, in BGQoS, which for example include *"The latest time that all tasks within the application MUST be completed"*, are delivered from the GRC to the GRP's resources as opposed to $QoS_{Parameters}$ which are delivered *via* the resource description that meets the $QoS_{Parameters}$ of the GRC.
$\mathbf{QoS_{metric(s)}}$
$\mathbf{QoS_{metrics}}$ are defined as the measurement criteria or units of measurement for $QoS_{Parameters}$.
$\mathbf{QoS_{characteristic(s)}}$
$QoS_{characteristics}$ are defined simply as the equivalent of $QoS_{parameters}$ which are provided by resources. Not all resource $QoS_{characteristics}$ are input by the GRP. BGQoS supports, dynamic calculation of specific characteristics, which are updated according to information retrieved dynamically throughout while the resource is available for allocation.
$\mathbf{QoS_{offer(s)}}$
A QoS_{offer} is defined as a response to the input QoS request. These offers are a set of resources that fulfil the requirements input by the GRC. A single offer is part of the negotiation process.
$\mathbf{QoS_{description(s)}}$
A $\mathrm{QoS_{description}}$ includes all the information that is required for the complete operation from GRC requirements specification to resource allocation. This includes GRC, GRP, Resource and operational information.

The operational steps within BGQoS include resource discovery and selection by matching the QoS requirements submitted by the GRC with resource characteristics associated with resources available to the GRC. This matchmaking process may produce a number of resources that can meet these requirements. The QoS related components within BGQoS use $QoS_{parameter}$ as an input and the output is a list of resources that match against the $QoS_{parameters}$. This provides a high-level abstraction in which the matchmaking process is carried out by BGQoS using a set of $QoS_{parameters} \rightarrow \{\mathrm{QP_1}, \ldots, \mathrm{QP_n}\}$ and a set of resources providing the attributes represented as $Resource_{characteristics} \rightarrow \{\mathrm{Ch_1}, \ldots, \mathrm{Ch_n}\}$ meeting the requested parameters as output $\rightarrow \{\mathrm{Set_1}, \ldots\ \mathrm{Set_n}\}$.

The simplification of the matchmaking process is an aim of BGQoS, presenting the GRC with the option of attaining the best suited set of resources while hiding the complexities of the infrastructure and differences in definitions in their requirements and the resource characteristics. If we consider an GRC attempting to carry out a number of tasks by requesting to utilise computational resources, with the following $\mathrm{QoS}_{\mathrm{parameters}}$: $\mathrm{number\ of\ CPUs} = \mathrm{QP}_1$, $\mathrm{average\ CPU\ power} = \mathrm{QP}_2$, and $\mathrm{reliability} = \mathrm{QP}_3$, then BGQoS uses the information extracted from the submitted request, which includes the required $\mathrm{QoS}_{\mathrm{parameters}}$and maps them to suitable resources according to their characteristics. There are two types of characteristics, static and dynamic. Static characteristics such as number of CPUs are submitted by the GRP within the $\mathrm{Resource}_{\mathrm{description}}$ in the advertisement phase. These characteristics remain the same while the resource is made available by the GRP and is expected to deliver the QoS specified accordingly. Dynamic characteristics such as resource reliability are updated at specific time intervals, dynamically and according to current information retrieved from monitoring [31] the available resource. The updated information replaces the previous information, while a historical record is kept, providing the model with access to the current state of the resource in relation to specific parameters. The historical record is used for calculating dynamic information such as reliability and availability. BGQoS uses these characteristics to map the requested parameters from the GRC with those provided by different resources available.

3.3. BGQoS Flexibility

In the presence of more than one domain and multiple types of resources that may overlap in terms of ownership and description, it is important that the implemented QoS model is capable of distinguishing between different resources, different requests and different types of resources. This is supported and taken into consideration in BGQoS which provides a standard model that recognises these types in which a resource can be assigned a type and associated with a set of specific characteristics and resource information that are identified by a unique ID.

Different types of GRCs are also supported with an expandable multi-tier model [32] in which different assignments can be made according to different domains and different requirements. Moreover, these requests can be designed to reflect each tier, accordingly.

3.3.1. Component Separation

The independence and separation of components from each other enhances the flexibility of BGQoS. In practice this means, that each domain or administrative authority can tailor specific components according to their specification or requirements without affecting any of the other components or the operational functionality of BGQoS. For example, the independence of GRC tier specification and interface design enables the model to accommodate different tier interfaces and the specification of different $\mathrm{QoS}_{\mathrm{description}}$ generated. This allows the GRC to input their QoS requests using different implementations of interfaces each mapping to the tier that GRC belongs to without altering the operation of functionality of other components. Component separation allows multiple tears to be added with multiple interface designs associated with them, seamlessly and independent of other components within BGQoS. In BGQoS GRCs are divided into groups according to privileges and constraints. In this paper, Tier A GRCs are at the highest level and have the most privileges while Tiers B and C have a reduced set and more stringent constraints. The flexibility in defining categories for different types of GRCs is in line with BGQoS design approach where flexibility is necessary. Figure 1 illustrates this component separation within the Grid's layered architecture.

3.3.2. Symmetric QoS Model

BGQoS employs a symmetric QoS model. Let S be the multi-dimensional space representing the QoS parameters that a set of resources available to the GRC can provide, and let R be the requested $\mathrm{QoS}_{\mathrm{parameters}}$ by the GRC, which in turn represent a subspace in S. Traditionally, a request (α) has been defined as a subspace in S. An offer (O) is viewed as a point in space S. However, in this symmetric model an offer is considered as a subspace in S just as requests, representing the range of QoS values that a resource is going to supply. In this case O conforms to α if its subspace is within the subspace for α.

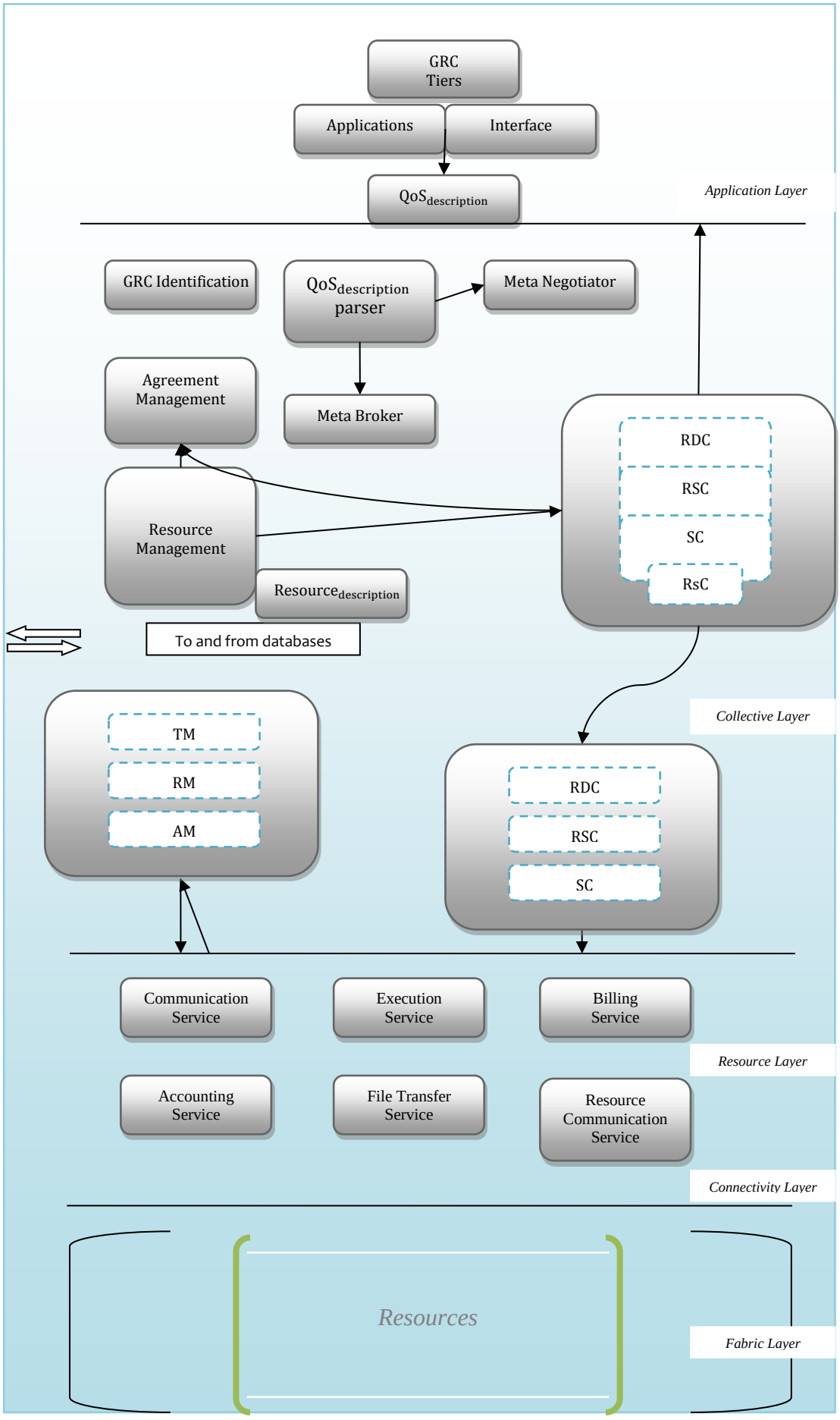

Figure 1: BGQoS within Grid Architecture

3.3.3. Standardising Request Inputs and Metric Unification

BGQoS is proposed for a multi-domain business context environment, with variable applications and variable GRC populations. It is therefore important that a method that would unify the high-level QoS metrics both for resource requirements and constraints, requested by different applications, is specified. To illustrate this, we use the constraints as an example. If a specific GRC inputs a requirement that states "all tasks must be completed before 1 PM on Wednesday and execution cost must not exceed 200", it might seem clear enough, however it raises a number of issues. Grid resources by definition are heterogeneous, geographically distributed, belong to different GRPs and adhere to policies within their own administrative domains. Therefore the constraint specification above is not sufficient and must be put in context. 1 PM in which time zone? 200 units of which currency? BGQoS uses templates which resolve this problem. The use of predefined units at the requirement specification phase reduces the chances of error or confusion.

3.3.4. Standardising Resource Information

Resource advertising is an important step in making resources discoverable, and this advertising process includes a description of the resource. A $\mathrm{Resource}_{\mathrm{description}}$ includes both static and dynamic characteristics that are required in identifying the resource's eligibility for selection in accordance with the input requirements by GRCs.

Resources could be a part of a candidate set of resources for selection and may have to be compared with other sets of resources through a filtering process, called resource ranking. Resource ranking is a process in which resource sets are compared in order to select the most suited. The success of this process relies on the ability to compare different resources against each other according to their characteristics.

4. QoS Capabilities

In this section we describe in more detail how BGQoS carries out matching and ranking resources according to consumer requirements.

4.1 Types of Match

A QoS GRC is capable of stating the $\mathrm{QoS}_{\mathrm{parameters}}$ they require [32]. BGQoS supports the allocation of resources that support a higher level than is initially required if the cost does not exceed the constraints set by the GRC. If we assume that an authorised GRC is requesting a computing resource in the shape of number of CPU cores, a computing resource in the shape of Memory in RAM and a storage resource, then one of three cases occurs within BGQoS. The first case is that of a perfect match, where the offer exactly matches the request in terms of type of resources and the level of QoS requested. The offer in this case if called a 'Perfect Match'.

Case 1: Perfect Match

$$\mathrm{QoS\ Parameters} = \mathrm{Resource\ Characteristics}$$

The formula is as follows:

{(CPU = Requested CPU) && (RAM = Requested RAM) && (Storage = Requested Storage)}

The second case occurs when the level of service provided by a resource is higher than the level of service requested by the GRC. The offer in this case if called 'Over Qualified'.

Case 2: Over Qualified

$$\mathrm{Resource\ Characteristics} > \mathrm{QoS\ Parameters}$$

The formula is as follows:

{(CPU ≥ Requested CPU) && (RAM ≥ Requested RAM) && (Storage ≥ Requested Storage) }

AND {(CPU > Requested CPU) OR (RAM >Requested RAM) OR (Storage > Requested Storage)}

In other words all three resource characteristics (CPU, RAM and Storage) must be at least equal to the requested QoS parameters and at least one of those three characteristics must exceed the matched requested QoS parameter (otherwise the case would be a perfect match rather than overqualified).

The third case occurs when the offer only partially meets the requested level of Service. The offer in this case if called 'Insufficient'.

Case 3: Insufficient

$$\text{Resource Characteristics} < \text{QoS Parameters}$$

The formula is as follows:

{(CPU<Requested CPU) OR (RAM<Requested RAM) OR (Storage<Requested Storage)}

If either case 1 ("Perfect Match") or case 2 ("Over Qualified") is true in relation to a set of proposed resources, then these resources can be considered for allocation. The set of potential resources that meet GRC requirements, are called candidate resources lists. BGQoS supports the negotiation process between the GRP and the GRC in this case. The model then filters the list for the most local and optimal solution meeting the GRC constraints using resource ranking. If case 3 ("Insufficient") occurs, then the resources are deemed unfit and are not considered as potential resource sets that could be allocated, initially.

The use of logical operators provides the option of introducing a variation of the BGQoS operational model where the GRC can specify a second requirement set in case the first one cannot be met. An OR operation is used in order to carry out this operation. For the same request, a GRC may wish to provide two descriptions: CPU_i, RAM_i and Storage_i as set of requirements (1) and CPU_h, RAM_h and Storage_h as set of requirements number (2). The first set is called a main request and is given a priority value over the second set.

$$\{(\text{CPU} \geq \text{Requested CPU}_\text{i}) \,\&\&\, (\text{RAM} \geq \text{Requested RAM}_\text{i}) \,\&\&\, (\text{Storage} \geq \text{Requested Storage}_\text{i})\}$$

OR

$$\{(\text{CPU} \geq \text{Requested CPU}_\text{h}) \,\&\&\, (\text{RAM} \geq \text{Requested RAM}_\text{h}) \,\&\&\, (\text{Storage} \geq \text{Requested Storage}_\text{h})\}$$

If both requirements can be met, the main request is used.

4.2 Time and Cost Constraints

When the GRC submits a $\text{QoS}_\text{description}$, they submit the number of tasks to be executed. The completion of the matchmaking process means that a set of resources has been selected for the tasks to be carried out. Using the information submitted by the GRC and the information available on the selected resources, time estimation and cost estimation can be carried out. These estimations are used for provision of live information during the run of the application, comparison between the delivered and expected level of QoS from the resources with tasks running and meeting the Time and Cost Constraints.

Time Estimation is a service which supplies the GRC with an estimated time of completion once their tasks are submitted and a request is met by a set of candidate resources has been implemented within BGQoS. The model uses the information provided by the GRC in the request and the allocated resource characteristics to calculate an estimated completion time that is returned to the GRC. The estimated time can be calculated as the sum of the following time components, if the tasks are carried out sequentially:

$$\text{eT} = \sum_{\text{n}=1}^{\text{k}} (\,\text{eQT}_\text{n} + \text{eET}_\text{n} + \text{eTT}_\text{n}\,)$$

eT is the total estimated time for an application with k Tasks. eET_n is the estimated execution time of a Task T_n, eQT_n is the queuing time for a task T_n and eTT_n is the estimated file transfer time for the same task, T_n. This service is referred to as the time estimator.

However, if the tasks are carried out in parallel, then the estimated time can be defined as the time at which the final task will be completed $\text{Time}_\text{finish}$ and can be defined as:

$$\text{Time}_\text{finish} = \max_{\text{n}=1,\text{k}} (\text{eQt}_\text{n} + \text{eET}_\text{n} + \text{eTT}_\text{n})$$

In addition to the time estimator, cost estimation can be requested by the GRC. The cost constraint specifies that a specific total cost should not be exceeded. Once a set of resources has been identified and selected, the cost of running the tasks could be calculated accordingly, using information on the price for using a resource per unit time $p(t)$ and the time the resource is expected to be occupying the resource until completion, or the estimated τ. Since the resource usage cost in BGQoS is calculated based on a time basis, i.e. the price is per unit time and the GRC is charged for the period of time during which they use the resource.

If an application has k tasks:

$$eC = \sum_{n=1}^{k} p_n(e\tau_n)$$

eC is the predicted cost for the entire application and $p\,(t)$ is the price of running task n on resource Ri for time t. eT_n is the estimated time it will require to complete task n on resource Ri. This service is called the cost estimator.

Candidate resources are discovered and ranked. Ranking candidate resources is accomplished via a multi-step filtering and ranking process that is initiated after accumulating the lists through matchmaking the $QoS_{descriptions}$ of the GRC with the $Resource_{descriptions}$. If the level of QoS available in the $Resource_{descriptions}$ and the $QoS_{descriptions}$ from the GRC produce a result of "Perfect Match" or "Over Qualified" then the resource is added to the list of potential resources. To achieve this, two questions are asked:

- *Is the GRC identified via the GRC_{ID} authorised to use the resources in the potential set? Do they have access?*
- *Is $QoS_{Available} \geq QoS_{Requested}$?*

The answer to both questions must be a "Yes" for the set to be accepted as a potential set and added to the initial list.

4.3 Filtering: Meeting the Constraints

The first filtering process occurs at this stage. The potential resources that are identified are checked against the two constraints input by the GRC as part of their QoS description, the time constraint and the cost constraint.

For a resource set S_i containing resources $\{R_1, \ldots\ldots, R_n\}$ selected as a solution to a $QoS_{descriptions}$ for an application App_i containing n Tasks then S_i meets the constraints if and only if:

$$\sum_{s=1}^{n} pR_s(t) \leq C \;\&\&\; \max_{t=1,n} \left(time_{finish_t}\right) < T$$

$pR_s(t)$ is the price of R_s for the time it was allocated to the GRC, $time_{finish_t}$ is the time at which the final task is completed. C, is the cost constraint and T, is the time constraint.

Once this is done, the lists of resource sets that do not meet the constraints set by the GRC are removed and the rest of the sets retained and included in a new list. This list is passed on to the next ranking stage.

4.4 Constraints Minimisation

In BGQoS an application is defined as a collection of connected tasks. We have also modelled the Grid as a collection of variable types of distributed heterogeneous resources belonging to different owners that can be pooled together to execute the tasks that comprise an application. These resources can include, in BGQoS, computing and storage resources, so a Grid can include a set of resources = $\{R_1, R_2, R_{3,\ldots\ldots} R_i\}$.

The GRC may wish to select a constraint to be minimised if the option existed. Consider time minimisation. Let us assume that there are more than one set s of potential resources that could execute n tasks submitted by a GRC according to their requirements and that $\max_{t=1,n} \left(time_{finish_s}\right)$ is the expected completion time for final task on

set s. If we consider the cost of executing the submitted tasks on resource s as c_s , then the aim is to choose the set of potential resources with the earliest $\max_{t=1,n}\ (\text{time}_{\text{finish}_s})$ while the following conditions are true:

$$c_s < C$$

And

$$\max_{t=1,n}\ (\text{time}_{\text{finish}_s}) < T$$

Where C is the cost constraint and T is the time constraint.

4.5 Rank According to the Proximity to QoS description

If we assume that the requirements are as follows:

$$R > 80\%$$
$$C = 150 \text{ units}$$
$$T = 450 \text{ units}$$

Where C is the cost constraint and T is the time constraint and R is the reliability constraint.

And we assume the potential list of resource sets returned is as shown in Table 2.

Table 2: Potential Resource Sets

Set_{ID}	eC	eT	R
S_1	180.56	300.05	96.8
S_2	172.5	400.25	90.5
S_3	165.	442.85	87.8
S_4	178.9	381.72	93.50
S_5	177.75	320.65	92.7
S_6	172.9	338.7	82.9

For the purposes of this example the GRC has chosen to minimise the cost while meeting the other requirements and constraints. In other words, the only time consideration would be that $eT < T$ without taking time minimisation into account. On the other hand, in terms of cost then $eC < C$ is taken with the objective of minimising the cost. In terms of reliability, it needs to be over 80%, $R > 80$. The sets returned meet these criteria and must be ranked, and the ranking is as shown in Table 3.

Table 3: Swap Decisions

Sets	C_{diff}	eT_{diff}	R_{diff}	Swap Ranks
S_1, S_2	-8.06	100.2	-6.3	Yes
S_2, S_3	-6.92	42.6	-2.7	Yes
S_3, S_4	13.32	-61.13	5.7	No
S_3, S_5	12.17	-122.2	4.9	No
S_3, S_6	7.32	-104.15	-4.9	No

The minus signs in the table above represent the differences for the second set in the comparison. Therefore, S_3 is selected as the highest ranked resource set solution

A ranking process combining the two ranking steps is available within BGQoS. For example, cost or time minimisation is carried out and the top f sets ranked $S_1 \rightarrow S_f$ are rearranged according to the proximity to $\mathrm{QoS}_{\mathrm{Requested}}$. Combining two ranking steps provides a more reliable, accurate and relevant selection method, however, it does incur more overhead for the resource selection phase. Moreover, the minimisation preference of the GRC must be included in the $\mathrm{QoS}_{\mathrm{description}}$.

5 BGQoS Extensibility

Matchmaking is the core operation of BGQoS. A flexible, accurate, simple and expandable approach to matchmaking based on the $\mathrm{QoS}_{\mathrm{descriptions}}$ submitted by GRC is implemented. These $\mathrm{QoS}_{\mathrm{descriptions}}$ outline the high-level requirements for their tasks. Moreover, descriptions are submitted in accordance to policies defined within a multi-tier user grouping model. The environment for which BGQoS is proposed itself includes multiple domains and many applications that differ in terms of their requirements and the way they may use the resources provided by different Grids. It is therefore very important that BGQoS is easily expandable to these variable domains. It is at this stage of active resource discovery, resource selection and matchmaking that BGQoS provides the basic functionality that can be built upon via other components to tailor to the different domains. This functionality includes the multi-interface model and resource broker selection.

5.1 The Multi-interface Model

In BGQoS a multi-interface model is implemented corresponding to the multi-tier GRC architecture introduced previously [32]. It provides the GRC with the ability to specify high-level QoS requirements that form the core of the $\mathrm{QoS}_{\mathrm{description}}$ directly. The interfaces also serve as the entry point to BGQoS and session initiators. The design and the components of the interfaces might differ according to domain or specific administrative requirements. Every GRC in our model belongs to one specific tier depending on the necessary specificity of the QoS parameters to be submitted. This user model can be expanded to accommodate as many tiers as every administrative domain sees necessary. Different interfaces have been developed for each tier with the objective of allowing each GRC access to the resources to which they have permission, as well as, allowing each GRC to specify their QoS requirements and constraints as specified in the definition of their level. The design of the interface at each tier will vary according to the expertise expected of the user and the type of QoS parameters they need to submit. These might be very detailed in terms of type of resource characteristics or they could be very general and high-level, for instance just specifying a deadline by which the task should be completed. Both the QoS parameters, the number and the type of tiers are extensible which means that the GRC requirements submission can be tightly tailored to the domain needs of the business. The matchmaking process is initiated on submission of the GRC application execution request and a description of the requirements in terms of QoS parameters, constraints and other relevant information that is specific to each domain.

5.2 Partner and Global Access to Resources through Brokers

In Grid environments, Grid users must rely on resource brokers to discover the appropriate resources that meet their requirements, which has led to this surge in the development of different resource brokers after the initial efforts of the Globus Grid Resource Broker. However, these different resource brokers have no clear way of communicating between each other, or a clear protocol for communication between a GRC and a GRP through a set of resource brokers. Moreover, in BGQoS the top level GRCs are allowed access to different resources on different Grids.

In BGQoS once GRCs submit their execution requests and $\mathrm{QoS}_{\mathrm{descriptions}}$, they expect that the appropriate resources be located and selected. In addition to the methods that have been introduced to achieve this goal, a component has been implemented in BGQoS, which operates as a high-level communication platform between different Grids and deciding which Grids to communicate with in case local resources cannot meet the GRCs requirements.

The first step is to populate a list at each location with a list of potential brokers that can be communicated with. In the following step the ranking technique is tailor implemented and applied by BGQoS, by implementing broker ranking that follows the following criteria:

- Priority: The first filtering phase is based on whether mutual agreements exist between different, Grids, organisations and domains. If these agreements exist, only brokers included in these agreements are considered and the rest are discarded. The newly populated list of potential brokers proceeds to the next filtering step.

- Location: The proximity of the location of the broker is important and taken into consideration when selecting the right broker, in keeping with BGQoS' objective to shorten distances between GRCs and the resources they require, for more efficient and smoother allocation.

- The resources available and the number of successful tasks that have finished successfully over a period of time, which can be calculated using up-to-date information.

- The load of the Grid that the broker is a part of.

$$GB_i \xrightarrow{rank} 1$$
$$GB_n \xrightarrow{rank} n$$

An outline of the process is shown in Figure 2. The algorithm shown selects the closest broker that meets the QoS requirements.

Step 1: Partner Brokers

Contact the Broker Repository and get a set of Partner Brokers, PBrokerSet

For all Brokers pj $\in$ PBrokerSet, calculate d(pi, pj), where d is the distance

Select p_{min} with minimum distance to pi

Contact partner broker

$S(i) \leftarrow S(p_{min})$

Repeat for all Brokers in PBrokerSet if resources do not meet QoS requirements

Step 2: Global Brokers

Contact the Broker Register and get a set of Global Brokers, GBrokerSet

For all Brokers gj $\in$ GBrokerSet, calculate d(gi, gj), where d is distance

Select g_{min} with minimum distance to gi

Contact global broker

Repeat for all Brokers in GBrokerSet if resources do not meet QoS requirements

Figure 2: Broker Ranking Algorithm

6 Reallocation

BGQoS' aim at assigning the right tasks to the right resources by a matchmaking process driven by $QoS_{description}$ submitted by GRCs and using available up-to-date information on resource characteristics and QoS levels has been explained. Sustaining this level of QoS is a major objective of BGQoS. The combination of appropriate resource selection and the sustainability of the level of QoS provided by the selected resources provide the guarantee to the GRC that their QoS requirements are met and maintained until the completion of the tasks submitted. The premise that both parties will adhere to what they agree upon is documented in a contract that is initiated by the GRC, received by the BGQoS and offered by the GRP. If there is a potential violation of the contract, which might occur for multiple reasons, including resource failure and performance degradation, then the reallocation components in BGQoS are activated.

6.1 Reallocation: Issues and Considerations

The decision to reallocate, from the GRCs point of view must be put into the context of whether it is beneficial or not. There are a number of issues to consider:

- The percentage of tasks that have already been completed.
- Whether there are available resources that could be allocated immediately while still meeting the GRC requirements.
- Whether the total cost, including the cost of moving the tasks from one resource to another is viable and within the constraints.

If the conditions are met, then reallocation is viable and beneficial, the rescheduling and reallocation components of the BGQoS are triggered. Our model performs rescheduling and reallocation in two ways: the first is to migrate to a different set of resources for guaranteed QoS GRCs; the second performs resource swapping for BE GRCs, if there are resources available. Figure 3 provides an overview of the structure of the BGQoS Monitoring and Reallocation system.

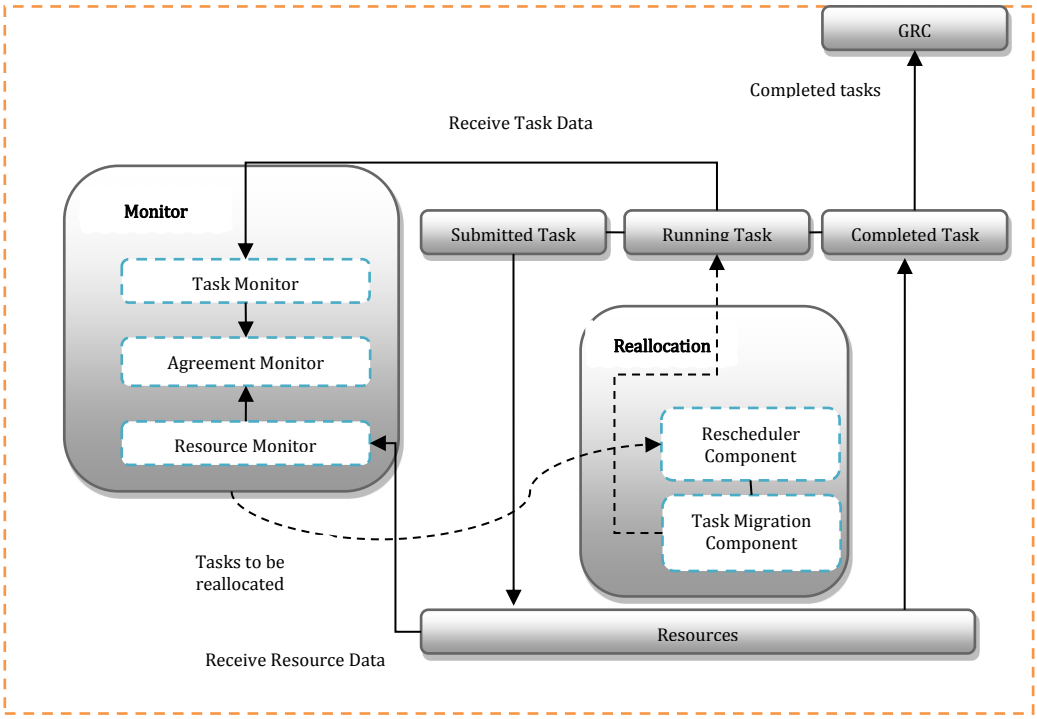

Figure 3: Monitoring and Reallocation of Tasks

6.2 Tolerance Ratio

It is necessary to specify the points at which reallocation is viable, numerically. For this a QoS ratio is introduced, calculated at specific time intervals, as the ratio between the values of the expected QoS to be delivered at time i, $\mathrm{QoS}_{\mathrm{expected_i}}$ and the actual QoS delivered at that point, $\mathrm{QoS}_{\mathrm{delivered_i}}$, effectively calculating the percentage of QoS delivered in relation to the original requested and agreed upon QoS. A Tolerance Ratio TR is specified by the GRC, expressing the percentage to be tolerated if the $\mathrm{QoS}_{\mathrm{delivered_i}} < QoS_{\mathrm{expected_i}}$. If none is set by the GRC, a default value is referenced. If $\mathrm{QoS}_{\mathrm{delivered_i}} < \mathrm{QoS}_{\mathrm{expected_i}}$ then the actual Delivered Ratio DR as a percentage is calculated as:

$$\mathrm{DR} = \frac{\mathrm{QoS}_{\mathrm{delivered_i}}}{\mathrm{QoS}_{\mathrm{expected_i}}} \times 100$$

Case 1:

If $\mathrm{DR} \geq \mathrm{TR}$ no migration is necessary and the application continues its operation normally until the next specified time for calculating a new DR, where the process is repeated.

Case 2:

If $\mathrm{DR} < \mathrm{TR}$, the rescheduling and reallocation components of the BGQoS check how much of the application has been completed. This process is related to the issues discussed in section 6.1, concerning whether there is any benefit in migrating the application at that specific point in its execution cycle.

$$\text{Migration decision: } \begin{cases} \text{Migration, if } \mathrm{DR} < \mathrm{TR}, \\ \text{No Migration , if } \mathrm{DR} \geq \mathrm{TR} \end{cases}$$

At this point, there are a number of issues to consider, including:

- The size of an application
- The QoS requirements
- The Cost constraints specified by the GRC
- The Time constraints specified by the GRC

The first issue is solved by proposing a percentage of tasks completed. This both eliminates the size of the application as parameter as well as maintaining the GRCs control over the reallocation procedure. This parameter is called Completion Ratio (CR). The percentage of actual tasks completed can be calculated as Actual Completion Ratio (ACR). The formula is as follows:

$$\text{Actual Completion Ratio} = \frac{\text{Number of completed tasks}}{\text{Total number of tasks}} \times 100$$

If $\mathrm{ACR} \geq \mathrm{CR}$ then no migration is carried out and the penalties are incurred, if $\mathrm{ACR} < CR$ then the third issue is to be considered. In general:

$$\text{Migration decision: } \begin{cases} \text{Migrate, if } \mathrm{DR} < \mathrm{TR} \text{ and } \mathrm{ACR} < \mathrm{CR} \\ \text{No Migration , if } \mathrm{RDR} \geq \mathrm{TR} \text{ or } \mathrm{ACR} \geq \mathrm{CR} \end{cases}$$

The third issue is whether the cost of migration is within the constraints of the GRC. If the first two conditions are met in (1) and (2) then the cost of migrating the resources is calculated as Migration Cost ($\mathrm{C_{migration}}$):

$$\mathrm{C_{migration}} = \mathrm{Cost}\,(\mathrm{set_{1_i}}) + \mathrm{Cost}\,(\mathrm{Migration}) + \mathrm{eC_{set_{2_j}}} - \text{Penalties imposed on GRP providing } \mathrm{set_1}$$

The condition to be met is:

$$\mathrm{C_{migration}} < \mathrm{C}$$

The Time constraint is the final issue to be considered. While the resources lists have fulfilled the initial Time Constraint requirements, the migration time must be added to the total execution time, introducing a newly calculated estimated time of Completion, $\mathrm{T_{migration}}$:

$$\mathrm{T_{total}} = \mathrm{T_{set_1}} + \mathrm{eT_{set_{2_j}}} + \text{Time required for Migration operation.}$$

The condition to be met is:

$$\mathrm{T_{total}} < \mathrm{T}$$

If the conditions are met, then migration is initiated. The migration operation specifies that under specific conditions, which are presented above, it is possible that tasks may migrate to a different resource before the final execution result is returned to the GRC.

7 BGQoS Evaluation

The experiments in this section have been designed in order to investigate the behaviour and performance of the separate operations and components within BGQoS and to evaluate specific functionalities that have been explained in this paper. Finally, the results are used to identify the operations' validity, and whether they achieve their goals.

7.1 Experimental Design

The experiments in this section each have a specific setup; therefore, each sub-section includes an explanation of the environment, the components and the context within which the experiment is carried out. This is followed by the results obtained and an explanation of the results. Moreover, the metrics used for obtaining the results are explained for each experiment. Table 4 outlines the experiments that have been included in this section.

Table 4: Experiments Overview

Experiment Name	Experiment Focus	Metrics
Overhead and Performance	The overhead incurred in milliseconds from the three main resource operations	$\text{Overhead}_1 = \text{Time}_{\text{Query}}$ $\text{Overhead}_2 = \text{Time}_{\text{Select}}$ $\text{Overhead}_3 = \text{Time}_{\text{ranking}}$
Resource Selection	The resources selected to complete that tasks according to GRC requirements	The rank of the resources selected.
GRC Access Authorisation	The differences in the success rate of task completion with relation to which resources the GRC has access to	$\frac{\text{Successful tasks}}{\text{Total Tasks}} \times 100\ \%$
Effect of the Number of QoS Parameters Requested	The effect of the number of parameters submitted on the resource operations and success rate.	
Reallocation and Migration	Average increase in successful completion according to QoS level with reallocation with increasing number of tasks submitted per minute, measured by total number of Tasks Completed	$\frac{\text{T}_{\text{reallocation}} - \text{T}_{\text{noreallocation}}}{\text{T}_{\text{Total}}} \times 100\%$
Violations	examines the number of violations and the number of successful executions within an experiment	The number of requests granted without violations The number of violated executions The number of violations outside tolerance ration The number of requests granted without violations The number of violated executions The number of violations outside tolerance ration
Reallocation with Ratio	examines the variation of ratios and parameters with reallocation operations	Migration decision (from): $\begin{cases} \text{Migrate, if DR} < TR \text{ and } ACR < AR \\ \text{No Migration , if RDR} \geq \text{TR or ACR} \geq \text{CR} \end{cases}$
Further comparison with FCFS	Comparison showing that BGQoS delivers a level of QoS that is requested by the GRC but also executes tasks in an efficient manner for a large number of tasks submitted over a long period of time	The number of tasks completed per day over the period of 50 days.

7.2 Overhead

Overall overhead can be broken down into specific overhead measurements, each reflecting the overhead at a specific stage within the operation of BGQoS. The following paragraphs identify the main overheads that can be calculated and identifies them as metrics for the measurement of overall overhead.

Resource Information Retrieval from Resource Repositories (RR)

Informational retrieval is carried out in response to a GRC execution request, through querying the appropriate databases containing resource information. Since the query is sent to a database, it simplifies the resource information retrieval process by eliminating the process of querying individual resources.

Resource information within BGQoS is updated at regular intervals and therefore is assumed to be up-to-date and reflecting the current state of the resources. The query returns a list of resources that fulfil the GRC requirements stated in the $\mathrm{QoS}_{\mathrm{description}}$.

The overhead from this operation is calculated as follows:

$$\mathrm{Overhead}_1 = \mathrm{Time}_{\mathrm{Query}}$$

$\mathrm{Time}_{\mathrm{Query}}$ is the time required to search the information in the database. This time varies depending on the size of the database, the number of resources and the number of accessible RRs.

Resource Selection According to $QoS_{description}$

Resource selection is the operation which concludes with the selection of resources that meet GRC requirements from the resource information retrieved process which are in turn extracted from the appropriate databases. The overhead incurred at this step is directly related to the time required to compare resources and their characteristics and whether they fulfil the requirements and constraints requested by the GRC.

$$\mathrm{Overhead}_2 = \mathrm{Time}_{\mathrm{Select}}$$

Where $\mathrm{Time}_{\mathrm{Select}}$ is the time required to complete the selection operations introduced in earlier chapters in order to confirm whether that:

$$\mathrm{QoS}_{\mathrm{requested}} \leq \mathrm{QoS}_{\mathrm{offered}}$$

Resource Ranking

Resource ranking uses the information fed in through the previous steps in order to rank the resources according to specific criteria. The overhead is calculated as the time required for completing ranking operations of candidate resources:

$$\mathrm{Overhead}_3 = \mathrm{Time}_{\mathrm{ranking}}$$

$\mathrm{Time}_{\mathrm{ranking}}$ is the time required to rank a list of resources within the candidate resource stack in an order that meets the GRC requirements. $\mathrm{Time}_{\mathrm{ranking}}$ is directly related to the number of candidate resources to be ranked.

7.2.1 Experiment Setup

The experiment is set up as three GRPs each providing a dedicated 10 CPU cluster for task execution. For this experiment, the CPUs in terms of computational power are equal. The GRC selects the number of CPUs it requires, the length of time it requires them for and sets a cost constraint that must not be exceeded. Between 3-30 tasks were submitted in the experiments.

7.2.2 Results

Figure 4 represents the overhead incurred in milliseconds from the three main resource operations; Resource Information Retrieval, Resource Selection and Resource Ranking.

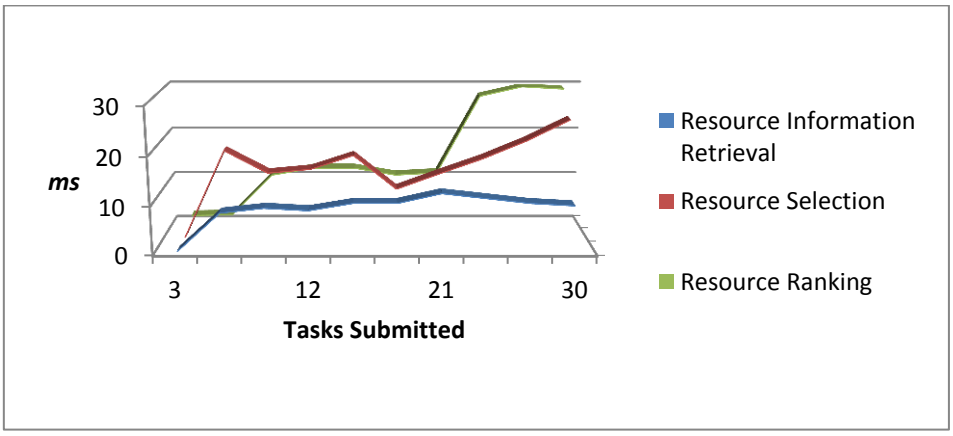

Figure 4: Overhead in *ms*

Resource information retrieval does not depend on the number of tasks to be submitted by the GRC, with resource information retrieval accruing in a stable manner; the small variations in Figure 4 are negligible. However, the overhead itself cannot be neglected, if predicted and expected. Within this experiment, it was around 10 ms.

The average overhead from resource selection was 15.5 milliseconds. This was followed by a similar overhead for the resource ranking operations which averaged at 19.1 ms resulting in an average of 34.6 ms of overhead (15.5 ms + 19.1 ms) after the information has been retrieved. This overhead is to be expected and because of the main objective of BGQoS, it can be tolerated. Knowledge of overhead related to usage patterns is important so that configuration or methodological decisions can be taken concerning caching of information, frequency of resource ranking, or even the use of independent agents to maintain and supply ranking information quickly.

7.3 Resource Selection

The resource selection process concludes with each resource being assigned a rank. The highest ranked resource set is to be selected, however, in real scenarios; this may not be the case. Many factors such as unexpected failures, resource degradation, dynamic availability information and policy mismanagement can result in that the top ranked list is not selected. This experiment presents the simulation carried out for evaluating this process and identifying the rank of the resources selected by the BGQoS to execute the tasks.

7.3.1 Experimental setup

- GRPs and Resources

The experiment is set up with 9 GRPs each providing a dedicated 10 CPU cluster for task execution. For this experiment the CPUs in terms of computational power are equal. A QoS GRC may select the number of CPUs they require, the length of time they require them for, a memory requirement and set a cost constraint that must not be exceeded, while a BE GRC must rely on BGQoS BE resource allocation with a predefined cost constraint.

- The GRC and Tasks

The results and comments below are related to single QoS GRC submitting 100-1300 tasks submitting multiples of 200 tasks i.e. 100, 300 ,500 ,700,…,1300 tasks. The tasks vary in size and the number of CPUs they require.

7.3.2 Results

While most successful requests have been met with resources of rank 1, there are a significant number of resources ranked 2nd, 3rd and 4th selected, due to random resource failures implemented within the experiment. More importantly, BGQoS has managed to maintain successful QoS driven selection throughout, by selecting the highest ranking, most appropriate and available resource set.

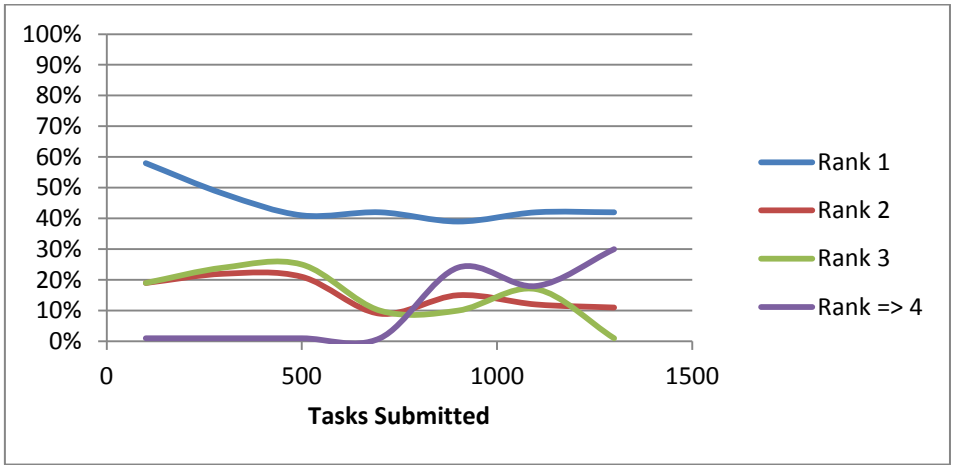

Figure 5: Rank Selected Percentage

Figure 5 also shows that the BGQoS ranking criteria provide an alternative if the highest ranked set of resources cannot be selected or is not the most preferred according to the policy matchmaking process between the GRC and GRP, for example. BGQoS has performed that successfully, while delivering the requested level of QoS.

7.4 GRC Access Authorisation

This experiment presents the results of an experiment carried out in order to examine the differences in the success rate of task completion with relation to which resources the GRC has access to; local, partner or global.

7.4.1 Experiment Setup

- GRCs:

Two types of GRCs are represented, QoS GRCs and BE GRCs split into a 40-60 percentage ratio. This is done in order to give BE GRCs a numerical advantage and explore the effects of that on the task completion ratio of the more privileged GRCs.

The QoS GRC within this experiment reflects the Tier A GRC. The BE GRC represents a Tier C GRC. The QoS GRC may select the number of CPUs they require, the length of time they require them for, a memory requirement and set a cost constraint that must not be exceeded, while the BE GRC must rely on BGQoS BE resource allocation with a predefined cost constraint.

The experiment uses twenty GRCs (12 QoS GRCs and 8 BE GRCs). Each submits 10 to 1300 tasks incremented by 10 at each run. Only QoS GRCs have access to non-local resources.

- Resources:

The experiments have been carried out using a populated database of 120 local resources, 120 partner resources and 120 global resources. Each of these resources is assigned random computational power equivalent to between 2.0 GHz and 3.2 GHz and Memory between 256 KB and 2GB of RAM.

7.4.2 Results

Figure 6 illustrates the results obtained:

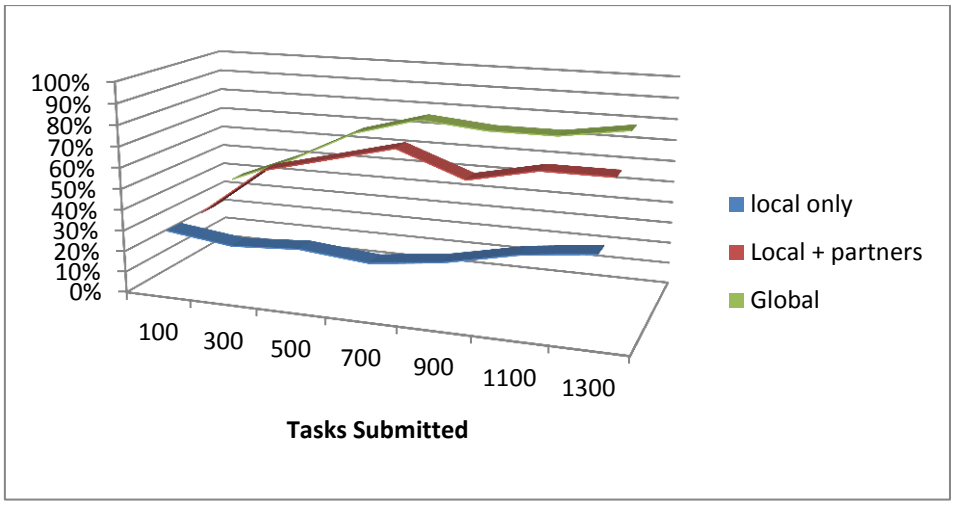

Figure 6: Successful Task Percentage - Local Access, Partner Access, Global Access

Where there is a significant difference between case 1 (local access only) and case 2 (local + partner) and case 3 (local + partner +global), the difference between case 2 and case 3 is not as significant, even though access to global resources represents a significant rise in the number of resources available. These results further emphasise the importance of a multi-tier GRC architecture where resource access is managed and facilitated while maintaining a balance between access and effectiveness.

7.5 Effect of the Number of QoS Parameters Requested

The number of parameters submitted by the GRC with the execution request in the $\mathrm{QoS}_{\mathrm{description}}$ vary. These parameters are used to locate the appropriate resources. This experiment examines the effect of the number of parameters submitted on the resource operations and success rate.

7.5.1 Experiment Setup

A database has been populated with a set of resources with variable reliability information. The resources had been used to carry out 500 mock tasks in which the failure rate has been random. This allowed the resource information to be updated and therefore, the information associated with the resource stored within the database represents a simulated real time information model. A single GRC submits 100 tasks with different sizes.

The experiments have been carried out using a populated database of 10 to 120 resource incremented by 10 at each run. Each of these resources is assigned random computational power equivalent to one of the following values 2.0, 2.4 and 3.2 GHz and Memory between 256 MB, 512 MB and 2GB of RAM.

- GRC requests:

The experiments ran with 1, 2, 3 and 4 QoS parameters requested by the GRC. The required parameters were as follows:

- 1 Requirement → CPU 2.4 GHz.
- 2 Requirements →CPU 2.4 GHz, RAM 512 MB
- 3 Requirements →CPU 2.4 GHz, RAM 512 MB, Reliability 80%
- 4 Requirements →CPU 2.4 GHz, RAM 512 MB, Reliability 80 %, Cost 350 units.

7.5.2 Results

Figure 7 shows the results from a run of the experiment that we have performed and the successfully completed tasks in each run:

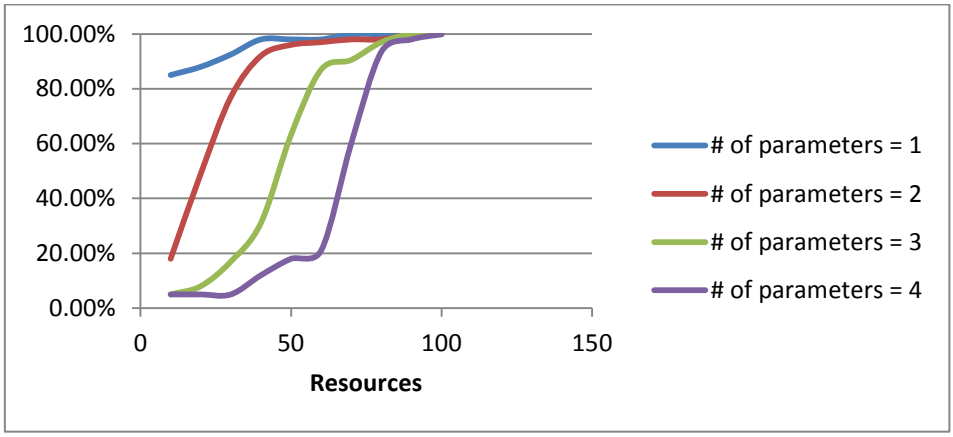

Figure 7: Successful Tasks - # of Parameters Requested

The results show that submitting fewer QoS parameters enables successful completion of more tasks. However as more resources become available the rate of successful tasks improves regardless of the number of QoS parameters submitted. This experiment illustrates the importance of providing resources according to the likely QoS pattern and also the relevance of QoS flexibility, the more flexible the requirements the more tasks can be successfully completed. BGQoS supports both these aspects by firstly supporting access to multiple brokers in BGQoS, thus increasing likelihood of availability of resources and secondly using logical operators in the multi-tier interface such that flexibility in QoS requirements can be specified.

7.6 Reallocation and Migration

Rescheduling and migration within BGQoS is carried out using a set of criteria that depends on a tolerance ratio set by the GRC, which in the case of this experiment is at 75 %. The tolerance ratio is a percentage within which a customer will accept deviation from specified resource characteristics. For instance a customer may have specified a certain amount of processing power but, in the case that the specified level is not available, will accept a job being run within a deviation of up to 75%. This experiment examines the effect of success and completion ratios with and without reallocation. It also shows the effect of increasing the number of submitted tasks per minute on the level of QoS delivered with reallocation.

7.6.1 Experimental Setup

Within this experiment, 20 simulation runs were carried out, each producing a random generated number of tasks as the initial number, increasing the number of tasks submitted by a specific percentage every simulated hour.

7.6.2 Results

Table 5 shows the effect of increasing the task submission rate on the percentage of tasks completed.

Table 5: Effect of Task Submission Rate vs Increase in number of Tasks Completed

#of tasks per minute	percentage of increase in tasks completed with reallocation
+5%	0.0
+10%	24%
+20%	12%
+30%	46%
+40%	37%
+50%	35%
+60%	11%

In all of the above cases, the number of tasks completed according to the GRC requirements and within the constraints has increased with reallocation employed as opposed to considering the execution as a failure when no reallocation is available.

Table 6 shows the effect of increasing the task submission rate on QoS level. The QoS reduction is calculated as an aggregate average of the percentage reduction of each job from specified requirement to actual allocation. In all cases the allocations were in line with the tolerance parameters specified.

Table 6: Effect of Task Submission Rate vs Decrease in QoS

#of tasks per minute	Percentage of QoS Reduction
+5%	0.0
+10%	-23%
+20%	-11%
+30%	-36%
+40%	1%
+50%	-18%
+60%	-4%

The QoS degradation associated with the increase in the number of tasks submitted per unit time is expected because of the competition that results from a larger number of tasks competing for the same number of resources in the available population, as well as the availability of resources providing a lower level QoS which are selected in order to meet the constraints submitted by the GRC. However, this degradation is tolerable and further enhanced with resource reallocation in case of failures. At one point an increase has been achieved, in this instance, the reallocated tasks were executed on resources providing a higher level of QoS than the original resources on which they would have been expected to be allocated. However, the normal situation would be for the resources providing a high level of QoS to be reserved or not available for running tasks, therefore it would be more difficult to achieve the same level or a higher level of QoS with a larger task submission rate.

7.7 Violations

This experiment examines the number of violations and the number of successful executions within an experiment. The purpose is to show how many violations occur in systems, where a violation is a level of QoS that does not meet the tolerance ratio set by the GRC and how BGQoS handles these violations.

7.7.1 Experiment Setup

- GRC and Task requests
  A single QoS GRC submits 500 tasks for execution with 1, 2, 3 and 4 parameters. The GRC is associated with a QoS delivery tolerance ratio of 85%.

- GRPs
  The experiment is setup of three GRPs each providing a dedicated 10 CPU cluster for task execution. For this experiment the CPUs in terms of computational power are equal.

- Resources
  A database has been populated with a set of resources with variable reliability information. The resources had been used to carry out 80 mock tasks in which the failure rate has been random. This allowed the resource information to be updated and therefore, the information associated with the resource stored within the database represents a simulated real time information model.

7.7.2 Results

The results obtained from the experiment above are presented in Table 7, where the number of executions without violations, the number of violations and the violations that required action are shown, as well as the number of granted GRC requests.

Table 7: Violations

# of Parameters	# of executions without violations	# of violated executions	# of violations outside tolerance ratio	# of requests granted
1	14	62	9	76
2	16	57	12	73
3	10	58	16	68
4	3	52	25	55

The number of violations according to the GRC ratio was surprisingly large; however, BGQoS has managed these violations well and performed a set of successful reallocation operations according to ratios, which the next experiment examines in more detail.

7.8 Reallocation with Ratio

This experiment examines the variation of ratios and parameters with reallocation operations.

7.8.1 Experiment Setup

- GRC:
  A single QoS GRC submits 500 tasks with the following requests for the complete execution:
  - Average CPU Power
  - Average Memory (RAM)
  - Storage

- GRPs and Resources:
  12 Clusters are simulated within this experiment. 10 are CPU clusters containing Computational resources, while 2 are Storage Clusters containing storage resources. The resource characteristics are in Table 8.

Table 8: Experimental Resource Characteristics

Type	Minimum	Maximum
CPU	2.4 GHz	3.8 GHz
RAM	128 MB	2048 MB
Storage	1GB	1024 GB

7.8.2 Results

Table 9 represents the results obtained from running 15 experiments with varying requirements on a set of resources, explained above. The table also illustrates the number of experiments in which violations beyond the ratio have occurred. Where the violations are within the ratio then the QoS requirements are still considered as met, while if the violations go beyond the tolerance ration, then reallocation is triggered as explained in section 6.1. Experiment 13 is an example where a violation has occurred. Average CPU requested was 6.4 but amount allocated was 2.6. The tolerance ratio is 90% but this situation falls below the tolerance level. In this case reallocation would be triggered.

Table 9: Meeting the Ratio QoS Demand

Exp	Ratio %	Average CPU Requested	Actual Average CPU	Storage Requested	Actual Storage	Average Memory Requested	Actual Memory
1	97	7.4	7.1	20	19	256	256
2	94	8.3	7.9	20	19	256	256
3	95	24.2	24.5	20	19	256	256
4	98	33.7	32.5	150	180.3	1024	1024
5	95	31.3	28.2	150	178.8	1024	1024
6	95	32.5	29.9	280	148.5	1024	1280
7	93	186	182.3	200	231.2	1024	1280
8	86	71	80.2	25	23	256	128
9	70	112	115.9	25	26	256	256
10	60	2.2	2.4	25	26.3	256	256
11	55	8.3	8.9	25	25	256	256
12	95	5.3	7.2	25	25.8	256	1024
13	90	6.4	2.6	25	28.9	256	256
14	80	2.5	8.9	25	29	256	256
15	85	8.5	6.7	25	25	256	256

The table shows the results from the 15 experiments run. It shows the actual resources delivered against the resources requested. The table also shows that the resources allocated delivered the GRC requirements successfully and within the ratio set by the GRC, often outperforming it while meeting the constraints.

7.9 Further Comparison with FCFS

In this experiment, further comparison with FCFS (First Come First Serve) is provided. The value of this comparison is that it shows not only that BGQoS delivers a level of QoS that is requested by the GRC but also executes tasks in an efficient manner for a large number of tasks submitted over a long period of time. This period has been chosen as a 50 day simulation period for this experiment.

7.9.1 Experimental Setup

- GRCs
  The experiment was carried out using two sets of GRCs; the first represented thirteen QoS GRCs (Tier A) submitting up to 2200 requests over a period of 50 days. The second was carried out with thirteen FCFS users submitting the same number of requests over 50 days.

- Resources
  12 Clusters are simulated within this experiment. 10 are CPU clusters containing computational resources, while 2 are Storage Clusters containing storage resources. The resource characteristics are in Table 8.

7.9.2 Results

Figure 8 illustrates the results obtained from both runs which show completions and the time it took for the process to be completed. Each line in Figure 8 represents a resource and the y-axis shows the percentage of the resource that was active at any time.

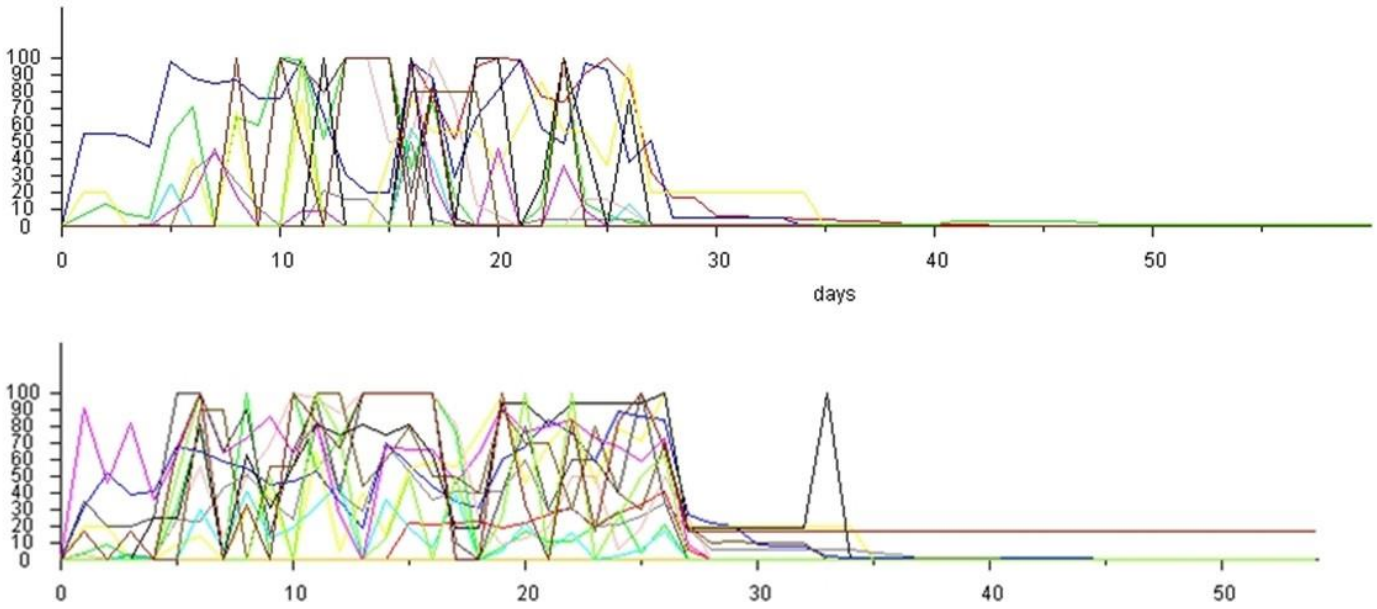

Figure 8: Comparison between BGQoS and FCFS over 50 days

The top half of Figure 8 shows the number of requests completed every day over a 50 days simulated period for BGQoS and the bottom half shows the same for FCFS. It is clear from the figure that all GRCs completed their tasks over the required period when BGQoS was used. This was not the case when using FCFS where there were tasks still running at the end of the simulated 50 day period. Moreover, we can see that task execution was more uniform and better organised under BGQoS whereas task completion appears less so using FCFS. It can therefore be concluded that BGQoS performed its resource operations, task execution and task completion driven by GRC requirements in an efficient manner illustrated in this comparison with FCFS.

8 Conclusion and future work

This paper has introduced the BGQoS operation relative to it's QoS and rescheduling capabilities. The evaluation of different components and operations through a simulated environment showed BGQoS' capabilities. In general, BGQoS has been able to deliver the successful results and has been able to carry out the operations it has been designed for.

From the evaluation, it is evident the BGQoS operates with a higher efficiency when there is a large population of GRCs and resources, efficiently meeting the GRC requests by using resource information and which has been successfully and dynamically maintained and updated. Moreover, BGQoS has been able to successfully carry out it's operations both in QoS driven mode and in BE driven mode, both in terms of resource operations and GRC access and limitations.

BGQoS presents many advantages including the simplicity it provides both GRC and GRP, effectiveness in combining the information from the GRC request and the resource information to locate and allocate the appropriate resources and most importantly its flexibility which allows it to accommodate different domains, GRCs and resources. QoS driven resource discovery and selection has been implemented to its intended effect, in which the resources are selected according to the $\mathrm{QoS}_{\mathrm{description}}$ submitted by the GRC. The GRC can therefore expect that the resources selected adhere to their requirements in terms of types and meeting the level of QoS required. Driven by these GRC requests and QoS specification, resource discovery employs a search mechanism that retrieves resource information and matches them with the GRC request in order to select the most appropriate resource for selection. The selection process is carried out using a ranking process which is flexible, configurable and expandable in order to accommodate different types of GRC, resources and domains.

Recovery, reallocation and migration operations are implemented in case of violation and error, complementing the monitoring process and guaranteeing the level of QoS delivered, while hiding the complexities of migration and reallocation from the GRC. This process is carried out automatically within the execution phase of an application by BGQoS.

An important area to expand on is the number of QoS requirements and the types of requirements that could be set according to a standard method of communication and metric unification between resource and service consumers

and resource and service providers. This would include a dynamic pricing method relative to the level of QoS provided and adhering to different agreements set in place between different parties.

Moreover, with the reduction of overhead and improving performance there could be a case for using real-time information on resources that is updated more frequently. In addition, a method could be employed that brokers inform GRCs of resource availability that they could utilise using the real-time information on resources; this provides a more business-oriented environment and opportunity for GRPs and provides the GRC with the opportunity of utilising available resources should they wish to.

The availability of information and the direct distinction between the different types of GRCs and their locality currently allows BGQoS to carry out resource selection operations using information on global resources, however, a more frequent update on resource information on a larger set of characteristics can allow resource operations to be carried out much more effectively.

As part of the future direction, we aim to tailor BGQoS for emerging Cloud Computing fields and expand them to enhance the QoS support within Cloud Computing environments. The rapid growth of these environments and their adoption by major organisations and corporations such as Amazon has added to the significance and importance of Cloud Computing and therefore, applying BGQoS would be beneficiary.

The elasticity and scalability characteristics of clouds means that there must be a solution that is capable of providing the user with the ability to specify resources and specific QoS without concern to the underlying infrastructure, keeping in line with one of the main objectives of Clouds. BGQoS could provide that solution and can help in user and request management.

9 References

[1] R. Al-Ali, A. Shaikh Ali, O. F. Rana, and D. W. Walker, Supporting QoS-Based Discovery in Service-Oriented Grids, in: Proceedings of the 17th International Symposium on Parallel and Distributed Processing , IEEE Computer Society, Washington, DC, USA, 2003, p. 102.1

[2] I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of High Performance Computing Applications, 15 (3) (2001) 200-222

[3] S. Krishnamurthy, W. H. Sanders, and M. Cukier, A Dynamic Replica Selection Algorithm for Tolerating Timing Faults. In: Proceedings of the International Conference on Dependable Systems and Networks, 2001, pp 107-116

[4] R. Albodour, A. E. James, N. Yaacob, A. N. Godwin, QoS Requirements for a Medical Application on the Grid, in : Computer Supported Cooperative Work in Design IV, Lecture Notes in Computer Science Volume 5236, Springer, 2008, pp.316-330

[5] M. Kluge, S. Simms, T. Williams, R. Hensel, A. Georgi, C. Meyer, M. S. Mueller, C. A. Stewart, W. Wuensch, W.E. Nagel, Performance and quality of service of data and video movement over a 100 Gbps testbed, Future Generation Computer Systems, 29 (1) (2013) 230-240

[6] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leyman, Service-Oriented Computing: A Research Roadmap, International Journal of Cooperative Information Systems 17 (2) (2008) 223–225

[7] R. D. Stevens, A. J. Robinson and C. A. Goble, myGrid: personalised bioinformatics on the information grid, Bioinformatics, 19 (supplement 1) (2003) pp. i302-i304

[8] F. Buccafurri, P. D. Me, M. G. Fudging, R. Fernery, A. Goy, G. Lax, P. Lops, S. Modifier, B. Penrice, D. Red avid, G. Demerara and D. Ursine, Analysis of QoS in cooperative service for real time applications, Data and Knowledge Engineering 67(3) (2008) 463-484

[9] K. S. Golconda and F. Ozguner, A comparison of static QoS based scheduling heuristics for a meta-task with multiple QoS dimensions in heterogeneous computing. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, IEEE Computer Society, Washington, DC, USA, 2004, pp

[10] S. Venugopal, C. Xingchen, and R. Buyya, A Negotiation Mechanism for Advance resource Reservations Using the Alternate Offers Protocol, Proceedings of the 16th International Workshop on Quality of Service, University of Twente, Netherlands, 2008, pp. 40-49

[11] A. Roy A. and V. Sander, GARA: a uniform quality of service architecture, in J. Nabrzyski, J. M. Schopf, and J. Weglarz (Eds.),. Grid resource management, 2004, pp. 377-394

[12] T. Yu, and K. J. Lin, Service selection algorithms for composing complex service with multiple QoS constraints, in: Proceedings of the Third International Conference on Service Oriented Computing, Amsterdam, Netherlands, 2005, pp. 130-143

[13] M. Wieczoreka, A. Hoheisel and R. Prodan, Towards a general model of the multi-criteria workflow scheduling on the Grid Marek, Future Generation Computer Systems 25 (3) (2009) 237-256

[14] Scale Out Software, Distributed Data Grids for the Enterprise [online] available from <http://www.scaleoutsoftware.com/> [accessed 9th May 2013]

[15] IBM Platform Computing, IBM Platform LSF [online] available from: <http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/> [accessed 9th May 2013]

[16] Grid Engine [online] available from <http://gridscheduler.sourceforge.net/> [accessed 9th May 2014]

[17] HT Condor® Project [online] available from <http://www.cs.wisc.edu/condor/> [accessed 9th May 2014]

[18] L. Pinedo, Planning and scheduling in manufacturing and service, Springer Science and Business Media, New York, 2005

[19] M. Hovestadt, O. Kao, A. Keller, A. Streit, Scheduling in HPC resource management systems: Queuing vs. planning, in: D. G. Feitelson, L. Rudolph and U. Schwiegelshohn (Eds.) The 9th International Workshop on Job Scheduling Strategies for Parallel Processing, 2003, pp 1–20

[20] W. Suß, W. Jakob, A. Quinte, and K-U Stucky, GORBA: A global optimising resource broker embedded in a Grid resource management system, in: Proceedings of the International Conference on Parallel and Distributed Computing Systems, 2005, pp.19–24

[21] I. Taylor, Triana generations, in: Scientific Workflows and Business Workflow Standards in e-Science, Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, 2006, p. 143

[22] Askalon, Distributed and Parallel Systems Group, University of Innsbruck [online] available from <http://dps.uibk.ac.at/projects/askalon/> [accessed 9th May 2014]

[23] C. Pautasso, JOpera Visual Composition of Grid Service, ERCIM News, No. 59, October 2004

[24] A. Hoheisel, User Tools and Languages for Graph-based Grid Workflows, Special Issue of Concurrency and Computation: Practice and Experience, 18 (10) (2004) 1101-1113

[25] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, and D. S. Katz, Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Journal of Scientific Programming 13(3) (2005)219-237

[26] Taverna Workflow Management System [online] available from <http://www.taverna.org.uk/> [accessed 9th May 2014]

[27] myGrid [online] available from <http://www.mygrid.org.uk/> [accessed 9th May 2014]

[28] E. Deelman, D. Gannon, M. Shields and I. Taylor, Workflows and e-Science:An overview of workflow system features and capabilities, Future Generation Computer Systems, 25 (5) (2009) 528-540

[29] F. P. Junqueira and K. Marzullo, The Virtue of Dependent Failures in Multi-site Systems, in: Proceedings of the 1st conference on hot hopics in system dependability, 2005, p.1

[30] G. Da Costa, M. D. Dikaiakos, and S. Orlando, Nine Months in the Life of EGEE: a Look from the South, in: Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, held in Washington, DC, USA, IEEE Computer Society, 2007 pp. 281-287

[31] T. Ropars, E. Jeanvoine, and C. Morin (2006),Providing QoS in a Grid Application Monitoring Service, Inria Research Report No. 6070 [online] available from <http://hal.inria.fr/docs/00/12/12/79/PDF/RR-6070.pdf> [accessed 9th May 2014]

[32] R. Albodour, A.E. James, N. Yaacob, High level QoS-driven model for Grid applications in a simulated environment, Future Generation Computer Systems, 28(7) (2012) 1133-1144