J Grid Computing
DOI 10.1007/s10723-016-9378-7

# A Meta-Brokering Framework for Science Gateways

**Krisztian Karoczkai · Attila Kertesz · Peter Kacsuk**

Received: 20 January 2016 / Accepted: 7 September 2016
© Springer Science+Business Media Dordrecht 2016

**Abstract** Recently scientific communities produce a growing number of computation-intensive applications, which calls for the interoperation of distributed infrastructures including Clouds, Grids and private clusters. The European SHIWA and ER-flow projects have enabled the combination of heterogeneous scientific workflows, and their execution in a large-scale system consisting of multiple Distributed Computing Infrastructures. One of the resource management challenges of these projects is called parameter study job scheduling. A parameter study job of a workflow generally has a large number of input files to be consumed by independent job instances. In this paper we propose a meta-brokering framework for science gateways to support the execution of such workflows. In order to cope with the high uncertainty and unpredictable load of the utilized distributed infrastructures, we introduce the so called resource priority services. These tools are capable of determining and dynamically updating priorities of the available infrastructures to be selected for job instances. Our evaluations show that this approach implies an efficient distribution of job instances among the available computing resources resulting in shorter makespan for parameter study workflows.

**Keywords** Meta-brokering · Interoperability · Distributed infrastructures · Workflows

K. Karoczkai · A. Kertesz (✉) · P. Kacsuk
Laboratory of Parallel and Distributed Systems,
MTA SZTAKI, Budapest, Hungary
e-mail: kertesz.attila@sztaki.mta.hu

K. Karoczkai
e-mail: krisztian.karoczkai@sztaki.mta.hu

P. Kacsuk
e-mail: kacsuk.peter@sztaki.mta.hu

## 1 Introduction

User communities worldwide use many kinds of workflow management and execution environments, which also raises interoperation problems among these working groups [27]. Workflow development, testing and validation are generally time-consuming processes and they require specific expertise to manage. These tasks hinder the growth of the number of available workflows and slow down the production of research results, so it is important to reuse them. Communities using similar workflow engines can benefit from sharing previously developed and tested applications, like using the myExperiment collaborative environment [1].

Besides sharing, interconnecting and combining these workflows can lead to further research gains and save time. Unfortunately, workflows developed for one workflow enactment system is most of the time not compatible with workflows of other systems. In the past, if two user communities using different workflow systems tried to collaborate, they had to redesign the application from scratch to the desired

workflow execution platform. To overcome these difficulties, new workflow interoperability technologies must be developed, which was the basic goal of the European SHIWA project [2]. By using the SHIWA technologies, publicly available workflows can be used by different research communities [25] working on different workflow systems, and are enabled to be executed on multiple distributed computing infrastructures. As a result, workflow communities are not locked any more to one workflow enactment system and its supported computing infrastructure. The main goal of the European ER-flow project [3] was to disseminate the achievements of the SHIWA project and use these achievements to build workflow user communities in Europe. It provided application support to research communities within and beyond the project consortium to develop, share and execute workflows with the SHIWA Simulation Platform. Both projects supported the execution of parameter study workflows over a growing number of DCIs, therefore it is crucial to develop such resource allocation algorithms that are capable of efficiently distributing a large number of simultaneously submitted parameter study jobs among resources of these heterogeneous infrastructures. These requirements also serve as a motivation for this work.

Managing these heterogeneous DCIs and scheduling the jobs of these scientific workflows are difficult problems and require sophisticated approaches to tackle. Many of these workflows apply the parameter study construct to examine large input sets with specific algorithms (e.g. [1]). A parameter study (PS) job of a workflow generally has high number of input files to be consumed by independent job instances. The state-of-the-art approaches for executing such parameter study jobs can be classified to two different strategies: (i) pull models try to avoid relying on outdated information on DCI load, therefore they start a pilot job in a DCI and pull jobs to it. Concerning the more traditional push approach (ii), generally all job instances are submitted simultaneously to the scheduling component of the workflow management system. This may cause significant overheads in service response time and bottleneck problems. Therefore additional information needs to be taken into account during workflow execution. For example, Fan et al. [10] applied an estimation of execution and wait time prediction based on Karnak prediction service in the SEAGrid science gateway using data mining techniques. In this paper we measure the waiting time instead of using job run time estimations.

To cope with these problems, in this paper we propose a meta-brokering framework for science gateways [26] to enable grouped DCI selection and submission for these instances. As job scheduling in these heterogeneous distributed systems is NP-hard [12], there is a need for solutions that apply some sort of heuristics and sophisticated approximations. Some of these methods are based on runtime estimates and the inaccuracy of these estimates is a perennial problem mentioned in the job scheduling literature. Even if users are required to provide these values, there is no substantial improvement in the overall average accuracy [9]. The background load of the utilized DCIs is also hard to estimate, and only some of them provide public information on dynamic resource load, which are usually outdated [9]. Our current approach tries to follow dynamicity in resource utilization with the help of so called resource priority services.

Therefore the main contributions of this paper are: (i) the design of meta-brokering framework for science gateways to enable automatic DCI selection for parameter study jobs of workflows, and (ii) the evaluation of this approach in a real-world science gateway solution called gUSE.

The remainder of this paper is as follows: Section 2 introduces the multi-DCI selection problem, and Sections 3 and 4 describe our proposed meta-brokering framework to support the efficient execution of parameter study workflows, including implementation details and evaluations. Section 5 presents improvement ideas to our current solution. Finally, Section 6 discusses related works, and the contributions are summarized in Section 7.

## 2 Multi-DCI Selection Problem

There are many user communities who need to access several DCIs in a transparent way, but they don't want to learn the peculiar features of these infrastructures. They want to concentrate on the execution of their scientific application. A science gateway [28] provides an interface between a scientist (or the research community) and the distributed computing infrastructures (DCIs). It can be imagined as a framework that provides a specific set of enabling technologies as well as frontend and backend services that together

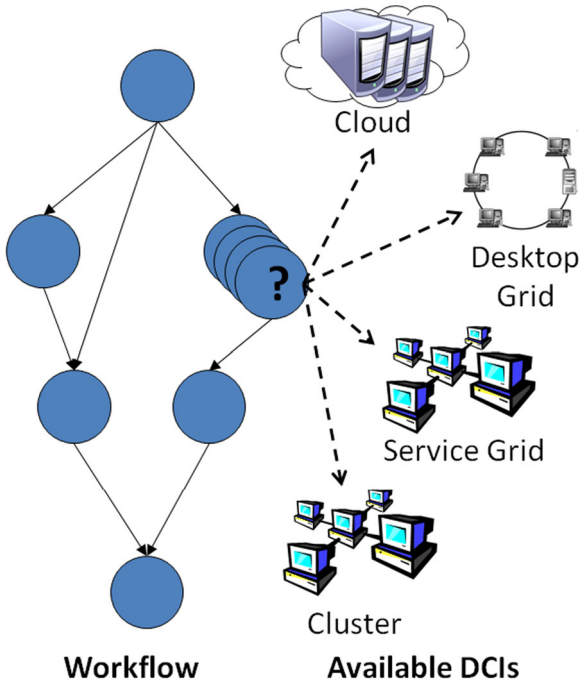


**Fig. 1** Problem of multi-DCI selection in science gateways

build a generic gateway. These frameworks are not specialized for a certain scientific area and hence scientists from many different areas can use them. Typical examples of such enabling technologies are: web application containers, portal or web application frameworks, database management systems, and workflow management systems. To efficiently manage multiple, heterogeneous distributed infrastructures within a science gateway, a meta-brokering approach is needed to coordinate, manage and efficiently distribute parameter study job instances over the available DCIs. Workflows applying the parameter study construct have multiple input sets to be executed with multiple job instances of the same program binary. Hence, a parameter study job has a large number of input files to be consumed by independent job instances.

Meta-brokering means a higher level brokering approach that schedules user jobs among various distributed infrastructures. This approach can also be regarded as a federation management solution. Current state-of-the-art works usually target one or two infrastructure types to form a federation. E.g. in Grid infrastructures, the InterGrid approach [18] promotes interlinking of Grid systems through peering agreements to enable inter-Grid resource sharing. Regarding Cloud systems, Buyya et al. [19] suggest a Cloud federation-oriented, opportunistic and scalable application services provisioning environment called InterCloud. They envision utility oriented federated IaaS systems that are able to predict application service behavior for intelligent down and up-scaling infrastructures. Some works investigated federating Grid and Cloud systems [20], but managing multiple infrastructures is rarely supported by multi-DCI brokering.

The problem for multi-DCI selection in science gateways is depicted in Fig. 1. The traditional approach for execution parameter study jobs is to submit all instances simultaneously (i.e. one-by-one) to the scheduling component of the workflow management system of a science gateway that forwards them to a predefined infrastructure set by the user. This may cause significant overheads in service response time and bottleneck problems, and it can also overload certain DCIs.

Our proposed default meta-brokering approach to be applied for parameter study jobs of a workflow distributes the load evenly among the connected DCIs based on their number of available resources. This approach is shown in Fig. 2. This job distribution

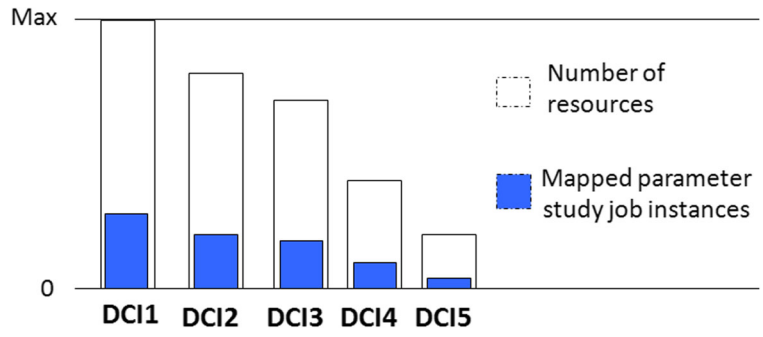


**Fig. 2** Default meta-brokering approach for DCI selection for parameter study job instances of a workflow

implies a load balancing that puts more job instances to DCIs having higher throughput, thus having more resources to execute a certain number of jobs.

Note that this approach does not take into account the background load of the managed DCIs (i.e. the already running or waiting jobs on the actual infrastructure). Specific DCIs like Grids may have monitoring components that provide such information through so-called Information Systems. Unfortunately the monitored data they publish is generally outdated, and cannot be used for exact schedules. Not to mention that usually the execution times of the background jobs are unknown, even user estimates are inaccurate [9] for their submitted jobs. This uncertainty makes job scheduling even harder in Grids, and of course, this is the case for multiple, heterogeneous DCIs. Some DCIs such as local clusters may provide more accurate load information, but for example commercial clouds inherently hide load information on the underlying infrastructure.

In an earlier work we have designed a meta-brokering approach [17] suitable for these needs having five major components (shown in Fig. 3). The Meta-Broker Core is responsible for managing the interaction with the other components and handling user interactions. The MatchMaker component performs the scheduling of the jobs by selecting a suitable broker. This decision making is based on aggregated static and dynamic data stored by the Information Collector component in a local database. The Information System Agent is responsible for regularly updating static and dynamic information on resource availability from the interconnected infrastructures. The Invoker component forwards the jobs to the selected broker and receives the results. Each job submitted for matchmaking is supplied with a standard description document containing their quality of service attributes. More information on these components and the utilized description language can be read in [17]. By developing our meta-brokering framework we took into account these theoretical guidelines, and created a concrete implementation to our DCI-Bridge tool to be detailed in the next subsection.

Various DCIs have different characteristics, which makes them hard to compare. Service Grids have active resources that continuously execute submitted jobs, while desktop Grids have volatile resources that can go offline any time a volunteer donor wants to use it. Clouds have virtual resources that have to be deployed before using them, which also affects the load and waiting time of the jobs. What common in these characteristics is that in all DCIs a certain amount of delay exists that affects the execution time of a submitted job. Note that once a DCI is selected by the meta-broker for a job (or group of jobs), the local scheduler of the appropriate DCI will perform the actual resource selection within the DCI.

In order to incorporate dynamic information to our meta-brokering approach we try to track and estimate this delay. This estimation will be incorporated to the former default job distribution approach to arrive to a more dynamic solution capable of avoiding overloading certain DCIs. This extended approach will be described in detail in the next section after introducing our reference science gateway architecture and the solution for extending it with the default meta-brokering approach.

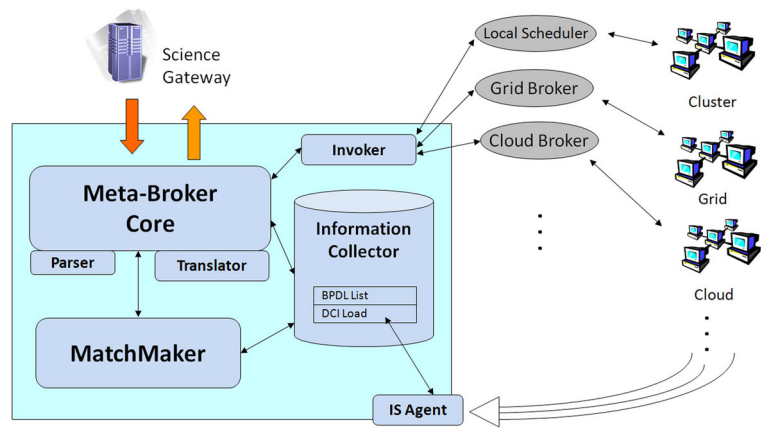


**Fig. 3** The architecture of GMBS

## 3 Static Meta-brokering for Science Gateways

Our proposed meta-brokering solution for parameter study workflows enables a randomized DCI selection and submission for these PS instances to better balance the load among them.

In order to achieve this in a science gateway, our approach proposes a meta-brokering component for the workflow managers. Traditionally each job of a workflow is assigned to a DCI. During workflow execution, the workflow manager uses the given DCI to submit and execute the actual job. With the help of a meta-brokering component, the user can assign a job to more than one DCI, and the final execution environment will be determined during runtime. The role of this new component is to select one of the candidate DCIs. The default algorithm can take into account the number of resources of each candidate DCI, and distribute parameter study job instances evenly among them (i.e. bigger DCIs should get more job instances). In this way the resource number represents a static weight for the actual DCI.

### 3.1 Implementation in gUSE

Our proposed meta-brokering solution for parameter study workflows is developed for a science gateway family called gUSE [8]. It provides necessary software stacks to develop science gateway frameworks and instances for various scientific communities (offering a simplified user interface that is highly tailored to the needs of the given community).

DCIs are managed by the so-called DCI-Bridge component in gUSE. It has been developed as an independent service for science gateways capable of submitting jobs of scientific workflows to different computing infrastructures. It hides the details of accessing these DCIs, and accepts standard job descriptions (in XML-based Job Submission Description Language format – JSDL) sent through a standard BES (i.e. Basic Execution Service) interface. DCI-Bridges can be chained together to enable job forwarding among them (with the help of so-called proxy DCI-Bridges).

There are different user roles within DCI Bridge (more details in [24]). First of all, (i) the system administrator is responsible for the definition of the main settings of the DCI Bridge objects (as shown in Fig. 4), for adding or removing a connector of a certain resource to (or from) an existing middleware group, further for the enabling or disabling and observing the flow of jobs to a certain resource. (ii) The common (power- or end-) user can browse among the resources which may be selected as the targets of the submissions to be defined during the workflow/job configuration phase. The common user may not change the actual settings of the DCI Bridge therefore he or she has a slightly modified user interface.

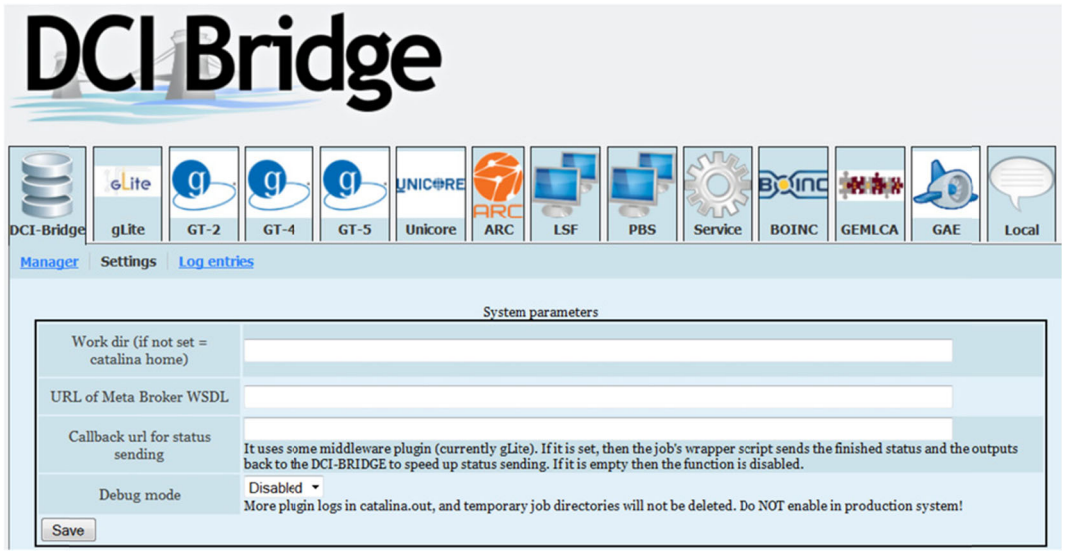


**Fig. 4** DCI-Bridge configuration

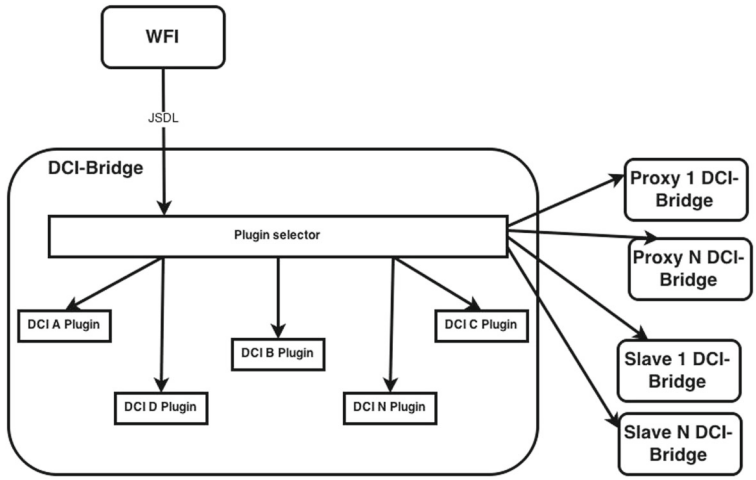


**Fig. 5** DCI-Bridge architecture with DCI plugins

The DCI Bridge has three main important properties. Firstly, it supplies the user interface, where the references about the resources of job executions are defined. Consequently only those resources are visible during workflow/job configuration, which have been defined by the Administrator user on DCI Bridge. Secondly, it contains the base routing parameters to remote resources, thus the actual job submissions can be controlled and observed via the DCI Bridge. Thirdly, which is most important for this paper, the

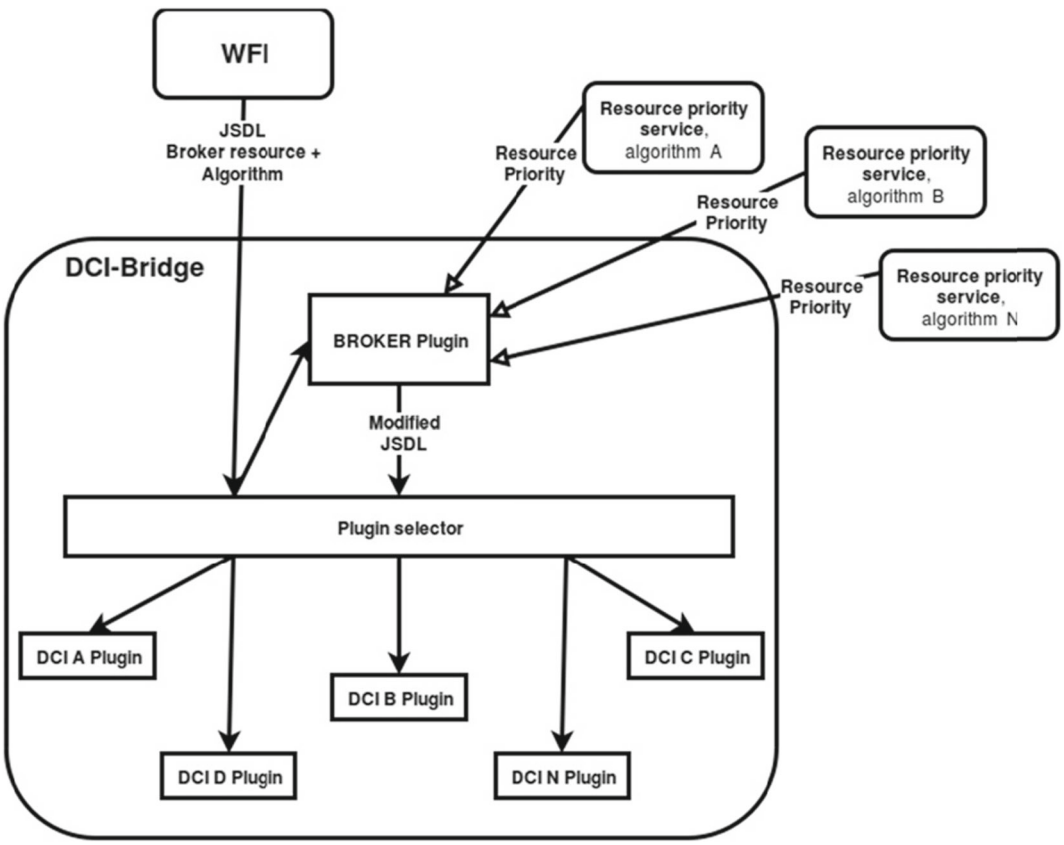


**Fig. 6** The extended DCI-Bridge with the Broker plugin

single interface of the DCI Bridge is an ideal insertion point for meta-brokering. In the background, with the call of a special subroutine-like service it performs the eventual late binding of jobs to resources of the available DCIs. A DCI-Bridge separates management operations of a concrete infrastructure into a plugin (as shown in Fig. 5). Therefore each plugin is responsible for storing jobs in its waiting queue (using the First Come First Served strategy - FCFS), submitting jobs to its DCI, monitoring their states and handling possible failures. The so-called Plugin selector component is used to determine which plugin should be selected to an incoming job request, based on its JSDL content.

We decided to develop a new plugin for our meta-brokering solution. This BROKER plugin is not connected to any DCIs, its role is to determine which DCI plugin should be contacted to manage an actual job submission. This decision is made on the information available in the actual job description (JSDL), and on the properties of the available DCIs. The JSDL contains the possible list of target infrastructures, and a meta-brokering algorithm changes this list to a concrete DCI (by modifying the JSDL) after selecting one of them. Finally the modified JSDL is sent back to the Plugin Selector. The extended DCI-Bridge architecture is shown in Fig. 6 (Resource Priority Services will be introduced in the next section).

By default, our proposed meta-brokering approach uses the static resource number of the available DCIs to determine the submission environment. In the DCI-Bridge we represent the throughput of DCIs by priority numbers, which are initially set to their static resource number by the system administrator of a science gateway. This number represents a weight for a DCI, and the BROKER plugin uses these weights to perform weighted randomized DCI selection for parameter study jobs of a workflow. This means that DCIs are selected randomly, but a DCI having twice higher weight then another gets selected with doubled probability.

This also means that in a science gateway using this approach users need to specify a list of DCIs and an algorithm name for all jobs of a workflow, if meta-brokering is needed to execute the workflow. When such a workflow is submitted, the JSDL of an actual job is sent to the DCI-Bridge, and it is given to the BROKER plugin. First it checks, if the user is allowed to utilize the named DCIs (i.e. has valid proxies to execute jobs on it). This step could be done in the science gateway itself, but since the DCI-Bridge has these capabilities, it is reasonable to perform it here.

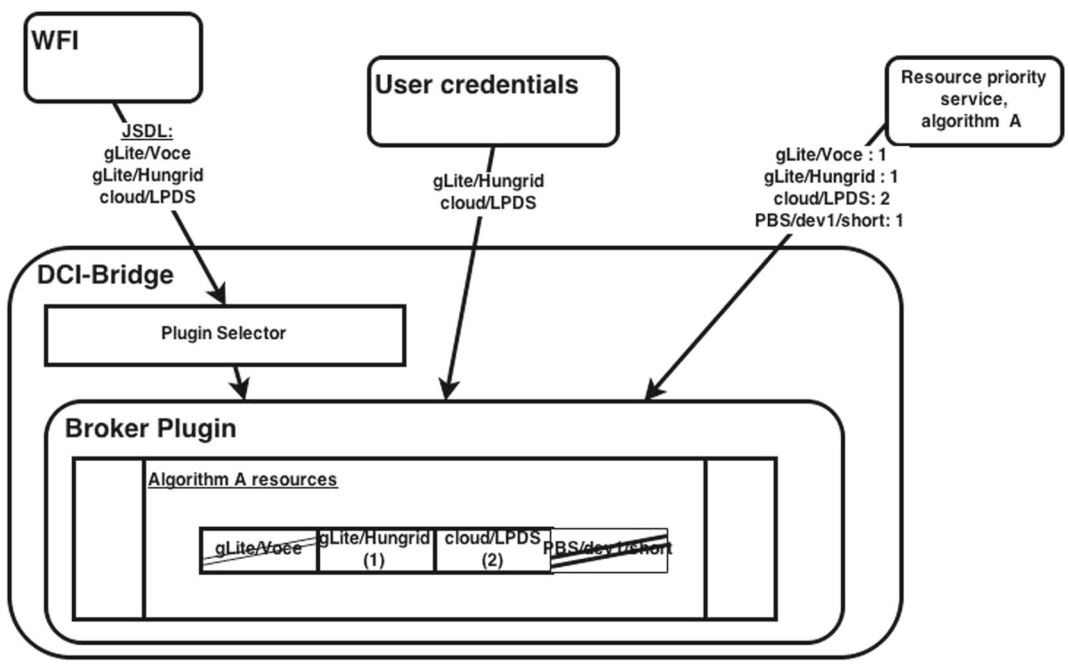


**Fig. 7** The extended DCI-Bridge with the Broker plugin

If this checking process does not find a valid proxy to a DCI listed in the JSDL of a job, it will exclude it from the potential list of DCIs to be used for the selection process. This step is exemplified in Fig. 7. In this case the user listed three DCIs, and only have proxies (i.e. credentials) for two of them (namely to HunGrid and LPDS Cloud), therefore these two DCIs will be the candidates for executing instances of the actual parameter study job.

Once the list of potential DCIs are determined, the BROKER plugin takes their priority values and performs a randomized selection weighted with the priorities. Such a selection can be imagined for all job instances as: all the candidate DCIs are put into a hat with occurrences given by their priority values, and one candidate is taken out of the hat randomly. The selected DCI will be written to the JSDL of the actual job instance, and it will be sent back to the Plugin selector, then forwarded to the named DCI plugin to perform the actual submission.

In order to use the proposed meta-brokering feature in gUSE, first the administrator of a DCI-Bridge service needs to set the priority weights for each DCI – as shown in Fig. 8. Weights are numerical values that typically reflect the capacity of a given resource (e.g., the number of CPUs available).

During submitting a job of a scientific workflow, the brokering service in gUSE will choose one of the actual computing resources with a distribution corresponding to the weights (by using weighted random distribution). Resources with higher weights take part with higher priority in the resource selection process, for example, brokering composed of three computing resources (local resource, gLite, cloud) and weights: 1 (local), 2 (gLite), and 3 (cloud), will submit about 16 % (local), 33 % (gLite), and 50 % (cloud) of the submitted jobs to the related resources. In case of static brokering, these weights are basically constants; the distribution of the submitted jobs over the resources does not change in time.

"0" weight means that no jobs are brokered to that DCI (e.g., LPDS OCCI), while DCIs having a positive integer value are potential candidates for the job execution. Once these weights have been set and saved, the jobs of a workflow can use the MetaBroker service of gUSE by selecting resource type: "broker" for the actual job. This selection possibility is shown in Fig. 9. Users should have valid certificates to all

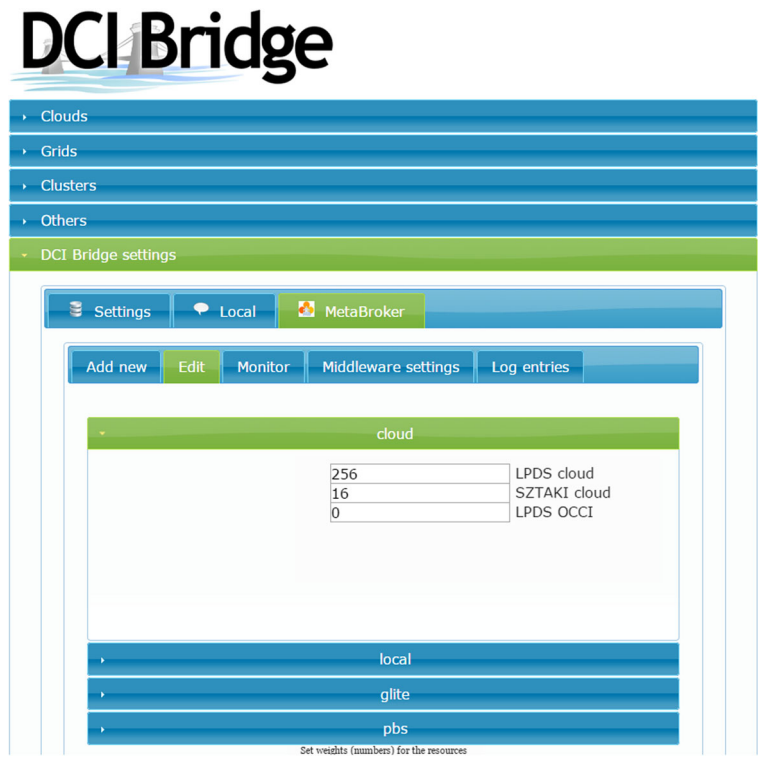


**Fig. 8** Defining DCI weights for the MetaBroker in DCI-Bridge

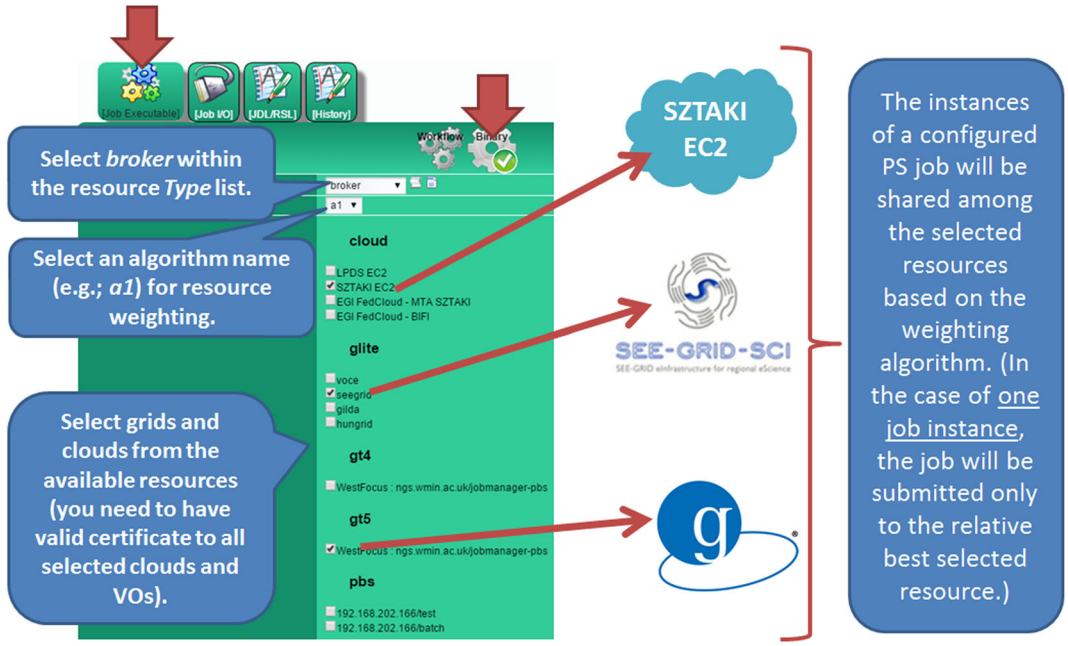


**Fig. 9** MetaBroker configuration in gUSE

selected DCIs, and the selected algorithm will determine the actual DCI selection based on the predefined weights.

### 3.2 Evaluation

In order to evaluate our proposed meta-brokering framework for science gateways, we have designed an artificial workflow that helped us to examine the introduced brokering capabilities of our solution. This workflow is shown in Fig. 10. It contains two generation phases to create parameter study job instances: the first job (*Gen10*) generates 10 output files which implies the creation of 10 instances for the next 3 jobs in both paths, and the third jobs in these paths (labeled *Gen5A* and *Gen5B*) generates 5 output files each to result in 5 dynamically created instances for the remaining jobs sorted to four paths. These generator jobs are marked with red circles in Fig. 10. The final jobs of the workflow collects the results of all parameter study job instances. In this way the execution of the workflow requires running 2062 job instances, which would take around 3 hours with a fully parallel execution (theoretically). In practice data transfers, latencies of the applied middleware and queue waiting times introduce additional execution time.

Our evaluation environment consisted of a gUSE-based science gateway with a DCI-Bridge connected to four DCIs: two PBS clusters located at Italy and Hungary: INAF-PBS (with 200 nodes) and LPDS-PBS

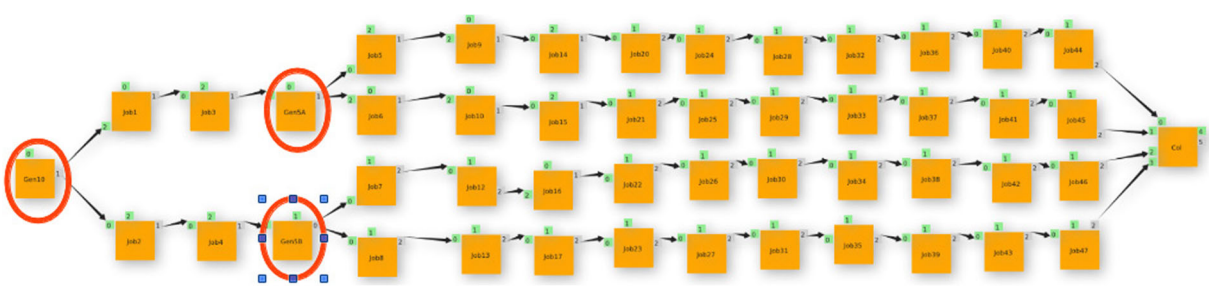

**Fig. 10** Test workflow for evaluation

(with 20 nodes) respectively, and two private clouds located at different regions of Budapest, Hungary: LPDS-Cloud (with 20 VMs) and SZTAKI-Cloud (with 30 VMs). In the first evaluation phase we executed the test workflow separately in all DCIs. We performed the measurements three times, and counted their average shown in Table 1. Only the INAF PBS cluster had enough resources to perform a fully parallel execution (meaning no queue waiting times occurred, only data transfers and communication processes prolonged the theoretical execution time). Therefore in the following measurements we exclude this cluster from the evaluation, and use only the remaining three DCIs.

In the second evaluation phase, we used the earlier introduced static meta-brokering capability of the DCI-Bridge service. In this case we configured the Broker plugin with the same priorities to the available three DCIs, which means that the parameter study job instances are distributed equally among the DCIs during workflow execution. The result is shown in Table 2. Since we used all DCIs this time to execute the parameter study job instances, we achieved some performance gain compared to two previous single executions. Nevertheless we experienced some VM failures in the cloud infrastructures, therefore some instances had to be reexecuted.

In the third evaluation phase we set the priorities according to the actual computing capacities (i.e. resource numbers) and previous performance values of the DCIs. The result is shown in Table II. In this phase we experienced additional performance gains. Though we also had some VM failures in this case, the DCIs having more computing capacities received more job instances than the others (i.e. the job distribution was weighted by the priorities), which resulted in additional speedup. The fourth row of the table shows the number of executed instances per DCIs.

In order to evaluate our proposed meta-brokering framework in a more dynamic environment, we repeated the third phase of Section 3.2 with increased background load on the strongest DCIs. Therefore we generated additional load with additional jobs on certain VMs of the SZTAKI Cloud to simulate a real world utilization. In this case we experienced longer makespan for the test workflow, hence the result was *5h 57min*. The instance distribution was also modified in this case: our new algorithm distributed the jobs according to the predefined static weights.

Figure 11 shows the total number of jobs submitted to all DCIs. This experiment proved our initial hypothesis that dynamic changes in the background load can really affect our brokered execution. As a result we need to react to these changes by modifying the static priorities. This modification can be performed by our proposed Resource Priority Service (to be discussed in the next section), but we have to mention that the initial priorities for the DCIs and the normalization function for refreshing the priorities should be chosen carefully with possible pretesting measurements in order to achieve additional performance gains.

**Table 1** Results of the first evaluation phase

| | DCIs of the science gateway | | | |
|---|---|---|---|---|
| | INAF-PBS | LPDS-PBS | SZTAKI-Cloud | LPDS-Cloud |
| Exec. time | 3h 30min | 13h 22min | 4h 46min | 6h 1min |

**Table 2** Results of the second and third evaluation phase

| | DCIs of the science gateway | | |
|---|---|---|---|
| | LPDS-PBS | SZTAKI-Cloud | LPDS-Cloud |
| 2nd phase Priorities | 1 | 1 | 1 |
| 2nd phase Exec. time | 7h 26min | | |
| 3rd phase Priorities | 1 | 3 | 2 |
| 3rd phase Distribution | 339 | 979 | 744 |
| 3rd phase Exec. time | 4h 1min | | |

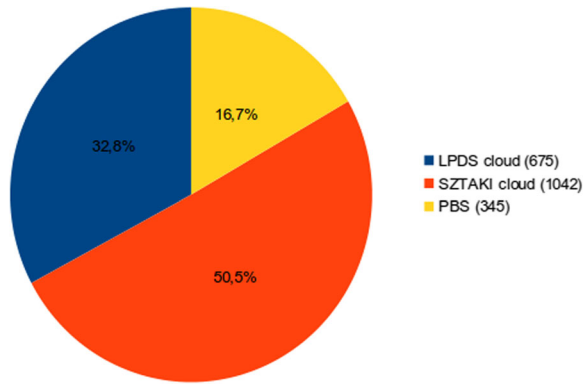


**Fig. 11** The applied job instance distribution in the third evaluation phase

## 4 Dynamic Meta-brokering for Science Gateways

As we introduced in Section 2, generally there is a need to change these DCI weights dynamically to avoid overloading an infrastructure, thus to reflect the background load in the weights (not only the size of a DCI). Next we detail, how DCI weights can be updated, and present a proof-of-concept implementation with a new component called a Resource Priority Service (RPS). These components are independent from DCI-Bridges, and can use any logic and algorithm to determine and refresh these priorities.

### 4.1 Implementation in gUSE

In order to modify DCI weights based on their dynamic background load, we propose to regularly measure the waiting time on these infrastructures. Our reference implementation for a Resource Priority Service uses so-called robot or probe jobs to determine this latency for a DCI that reflects its background load. To transform the waiting time to weights (i.e. priorities as we call them later), we use this latency to decrease the predefined number of available resources that serves as a base for determining DCI weights.

This task can be done by using some kind of normalization function. In general, the higher the measured waiting time is, the lower the weight the actual DCI should have. To achieve this, in our reference implementation of the Resource Priority Service in [11] we apply $f(x) = 1-(\log(x)/5)$ for the waiting time in minutes for getting a weight multiplier value between 0 and 1. This value will be multiplied by the predefined resource number in order to get lower weights for higher waiting times. This normalization function is depicted in Fig. 12, where the x axis represents the measured waiting time, and the y axis shows the multiplier values. For a concrete science gateway the applied normalization function may be different, since the average waiting times can differ for various DCIs, types of workflows and on the size of the active scientific community.

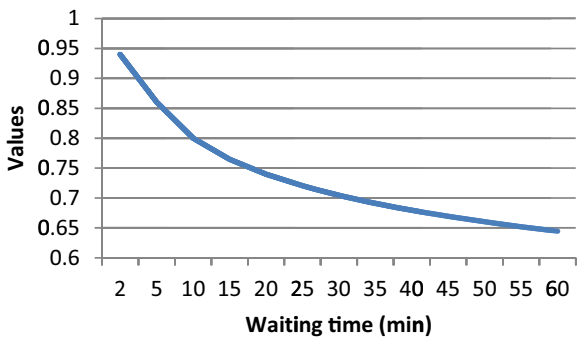


**Fig. 12** The applied function for weight normalization

An actual priority update from an RPS is stored in the DCI-Bridge under a unique algorithm name. This means that in this solution the priorities of any DCI set can be managed and updated by an RPS under a special algorithm name. A reference proof-of-concept implementation of an RPS service [11] is already available. This version uses the previously introduced and discussed weight normalization to determine DCI priorities. This service is implemented as a web service that can be deployed in a web server (e.g. a Tomcat server that hosts gUSE components and services).

Figure 13 illustrates, how robot jobs, also called as probe jobs, are used to refresh DCI (also called as Middleware) weights. We have two DCIs in this case: Middleware A and Middleware B, both having one-one resource. The following steps are needed to determine or update the weights of these DCIs:

1. The RPS should initiate the execution of a probe job on Middleware A and B (this can also be done separately)
2. a) The DCI-Bridge submits a probe job to Middleware A b) The DCI-Bridge submits a probe job to Middleware B
3. The appropriate DCIs execute the probe jobs
4. Once a probe job is finished, the DCI-Bridge gets notified
5. The DCI-Bridge sends a notification to the RPS of finishing the probe job (separately for the two DCIs)
6. The RPS service calculates and sends an updated DCI weight to the DCI-Bridge (as a reply for each notification)

Before deploying an RPS service, a system administrator needs to configure it with the WSDL of a DCI-Bridge to be connected to it. After deploying it, the service should be further configured with the managed algorithm name, with the JSDLs of robot jobs specifying their DCIs and with the initial priorities of these DCIs. As a prerequisite, the repository of the

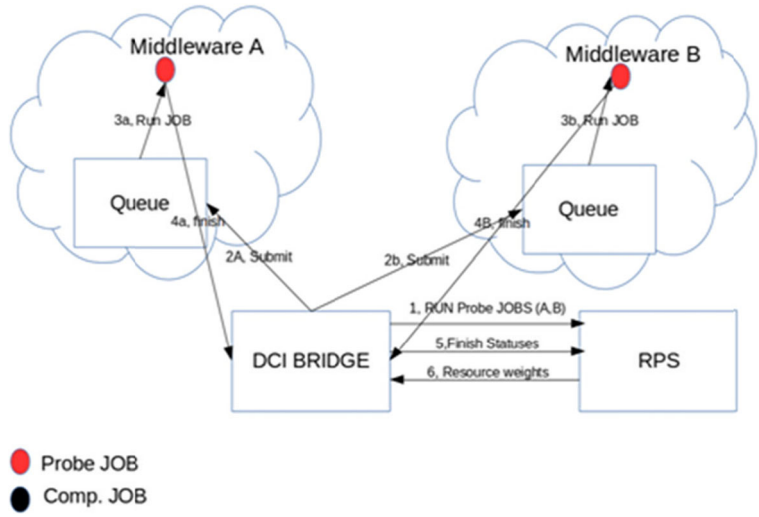


**Fig. 13** Load prediction with probe jobs

gUSE portal using the appropriate DCI-Bridge must contain these robot jobs as single workflows.

This approach also enables that different algorithms (i.e. priorities) can be used by the users for jobs of a specific workflow, as shown in Fig. 14. This figure depicts a workflow having generator and collector jobs used to manage parameter study job instances. Between these jobs the workflow contains compute-intensive and data-intensive PS instances. By using different algorithms for these two instance categories, more efficient workflow execution could be achieved. Therefore various scientific communities can develop their own RPS to better represent and manage certain DCIs in their science gateways.

Note that random selection is an important feature for selecting the submission environment. It could also be possible to choose the DCI with the highest priority for all job instances, but it could result in overloading the actual DCI. Randomized selection avoids this and also selects DCIs with less priority to better balance the load among the potential infrastructures.

## 5 Improvements

Though the presented meta-brokering framework can provide significant performance gains for parameter study workflows in science gateways (SG), our real-world experiences show that further improvements can be useful. As we have seen in Section 4, we use robot jobs, also called as probe jobs to determine the background load of a DCI.

If a DCI has a small number of computational resources or nodes, these probe jobs can waste significant execution time. Nevertheless, if a DCI has heterogeneous nodes (one is faster than the other), the measured load on a slow or overloaded node may rule out a generally well performing DCI from the selection process. These issues can hinder the performance gains of the meta-brokering framework, therefore gathered some improvement suggestions in this section.

Figure 15 depicts a situation for 12 PS job instances to be executed, where a DCI (Middleware B) has a

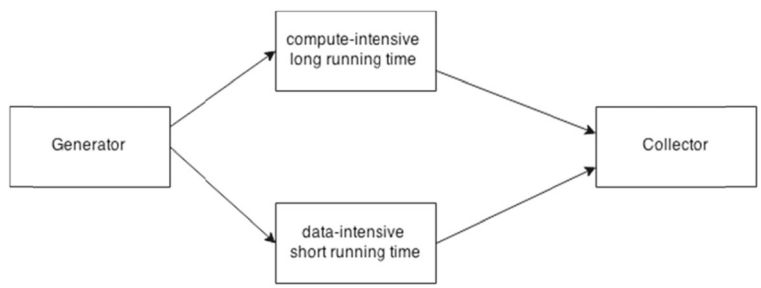


**Fig. 14** A sample workflow using different DCI priorities

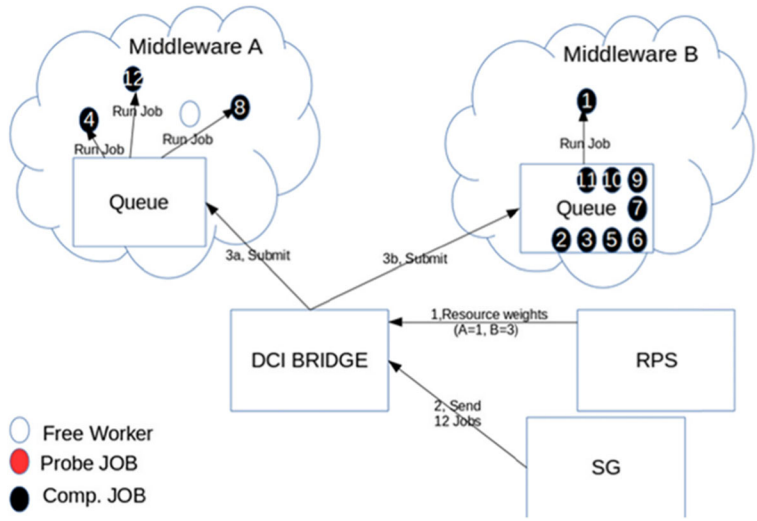


**Fig. 15** Load distribution for different DCIs

higher weight (i.e. 3, meaning that it is three times faster) than the other (Middleware A with weight 1), but it has only one resource behind a waiting queue. In this case we experience performance loss, because Middleware A could execute more jobs than assigned (because it has a higher number of resources to parallelize job executions) according to the measured weight by probe jobs. This situation is further detailed in Fig. 16, by exemplifying parallel job executions over time.

The optimal job distribution in this case would be to submit more jobs to Middleware A, thus the weight determination should incorporate the number of available nodes on a DCI, i.e. the level of parallelization achievable on a DCI. Such an optimal distribution is shown in Fig. 17. According to this finding, the RPS algorithm, taking into account only resource speeds measured by probe jobs, can provide significant performance gain only for DCIs with similar parallelization capabilities (i.e. similar resource number). As a result, we can improve the job distribution by replacing the default RPS algorithm to count the weight value by *measured speed * parallelization capability*. Thus the new weight for the situation depicted in Fig. 17 will change from (B-3, A-1) to (B-3, A-4).

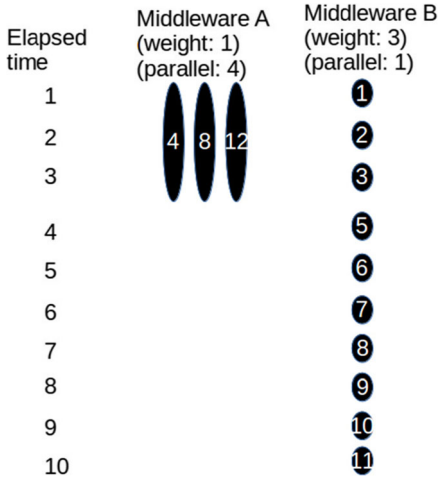


**Fig. 16** A possible unequal job distribution corresponding to Fig. 15

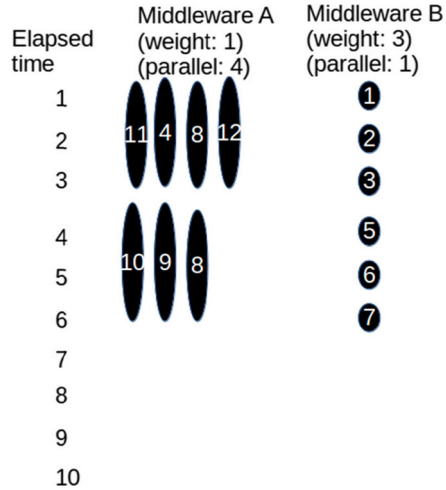


**Fig. 17** Equal job distribution corresponding to Fig. 17

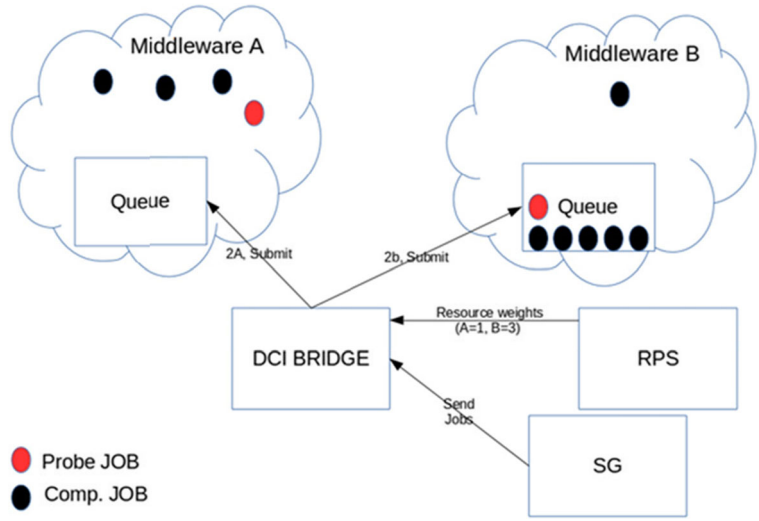


**Fig. 18** Probe jobs in waiting queues

Nevertheless we can still find cases resulting in an inefficient distribution. Therefore we can further improve the distribution by revising the weight calculation with the following equations for $n$ DCIs:

$$\mathtt{weight}_a = (\mathtt{speed}_a / \Sigma\mathtt{speed}) * (\mathtt{parrarel}_a / \Sigma\mathtt{parrarel}$$
$$\mathtt{weight}_b = (\mathtt{speed}_b / \Sigma\mathtt{speed}) * (\mathtt{parrarel}_b / \Sigma\mathtt{parrarel}$$
$$\ldots$$
$$\mathtt{weight}_n = (\mathtt{speed}_n / \Sigma\mathtt{speed}) * (\mathtt{parrarel}_n / \Sigma\mathtt{parrarel}$$

The second problem we experienced is that the probe jobs used for determining the speed of DCI resources can get stuck in waiting queues and reserve free resources causing additional waiting of user jobs as shown in Fig. 18. In such cases a new weight calculation for a group of jobs is done by taking the latest measured *speed* value of a DCI, which is definitely better than the one will be measured for the actually waiting probe job. A good solution for this would be to modify the measurement process, and track the execution process of these probe jobs, and mark the elapsed time since submitting the probe job. We can enable this by modifying the DCI-Bridge, not to wait for weight updates from the RPS, but to trigger weights for DCIs right before a meta-brokering phase. This modified version is shown in Fig. 19.

The third problem we found is the presence of third party jobs in DCIs. This is one of the main reasons

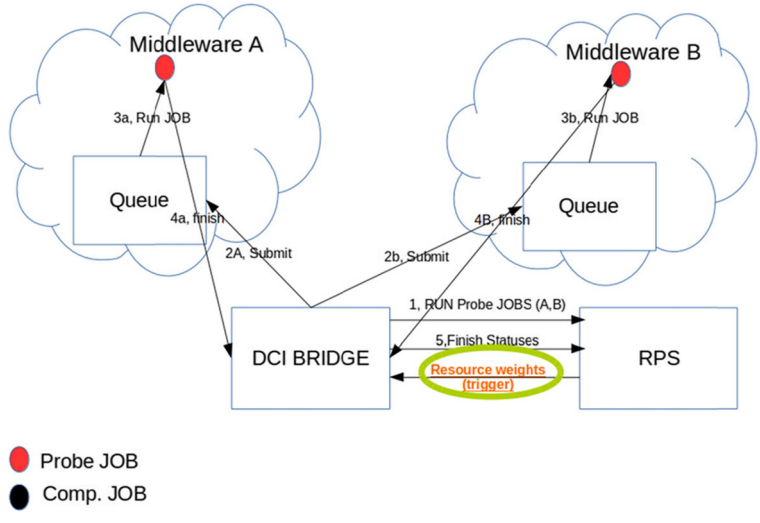


**Fig. 19** Triggered weight calculation by DCI-Bridge

uncertainty is preset in distributed systems. Some DCIs, such as Grids or clusters may allow job submissions and executions from different portals or gateways. In this situation the science gateway we are using is unaware that there are other jobs in the system than the gateway submitted. This case is shown in Fig. 20.

According to the calculated weights the system expect that a certain number of jobs can be run in parallel (e.g. 4 jobs in Middleware A of Fig. 20), but only 2 of them can be executed after submission, since 2 other jobs (marked with grey color) occupy the rest of the nodes. This means that optimal job distribution (and the highest performance gain) can be achieved only for such DCIs, for which the actual science gateway has full control. Typically cloud infrastructures can provide this. For others the meta-brokering algorithms have to be developed by taking into such uncertainty. We have also researched solutions for these cases by applying fuzzy approaches [21].

## 6 Related Works

Scheduling in Grid systems, which is one of the tasks of Grid resource managers, become even more complicated with multi-organizational shared resources, therefore Grid scheduling is also NP-hard [12]. Schwiegelshohn et al. [13] showed that the performance of Garey and Graham's list scheduling algorithm is significantly worse in Grids than in general multiprocessor systems.

While the principle of workflows is easy to conceive, their execution is very complex and no general solution exists. In an earlier joint work by Bacso et al. [23] we examined the resource allocation challenges of parameter study jobs in distributed computing infrastructures, and proposed a series of job allocation models that helps refining and simplifying the problem complexity. We examined some special cases that are polynomial and showed how more complex scenarios can be reduced to these models. The state-of-the-art approaches for executing parameter study workflows show high diversity based on their application area and execution environments. Hirales-Carbajal et al. [5] present an experimental study of 22 deterministic non-preemptive multiple workflow scheduling strategies in Grids. While their objective is to schedule and execute the whole workflow, and minimize its makespan, we restrict ourselves to parameter study jobs of such workflows. Oprescu et al. [6] propose a budget constraint-based resource selection approach for cloud applications, which can schedule bags of tasks onto multiple clouds with different CPU performance and cost, minimizing completion time with maximized budget. Their scheduler learns to estimate task completion times at run time. GridBot [7] represents an approach for execution of bags-of-tasks on multiple Grids, clusters, and volunteer computing Grids. It has a Workload Manager component that is responsible for brokering among these environments, which is similar to our approach, but they focus on tasks more suitable for volunteer Grids.

Casanova et al. [14] focused on the same problem of scheduling parameter sweep applications on Grids, with particular attention to file transfers and network performance. Also, their motivation is in alignment with most of the papers in this area. Their approach is modifying existing heuristics so that they are adaptive in a dynamic heterogeneous environment. The core of the scheduling is a Gantt chart that is created and updated periodically and keeps track of job and resource assignments. Assignments are governed by a heuristics called sufferage, where a task is assigned to a host if the task would suffer the most if done otherwise. Maheswaran et al. [15] addresses the issue of mapping independent tasks onto heterogeneous computing systems. They apply heuristics aiming at optimizing for throughput, i.e. increasing the finished task per time unit ratio. J. L. Lucas-Simarro et al. [16] proposed different scheduling strategies for optimal deployment of services across multiple clouds based on various optimization criteria. The examined scheduling policies include budget, performance, load balancing and other dynamic conditions.

The execution of parameter study jobs can be classified to two strategies: the push and pull models. Pull models try to avoid relying on outdated information on DCI load, therefore they start a pilot job in a DCI and pull jobs to it, i.e. feed the actual resource with own jobs. For example DIRAC and GWpilot [4] use this approach. Concerning the more traditional push approach, generally all job instances are submitted simultaneously to the scheduling component of the workflow management system. This may cause significant overheads in service response time and bottleneck problems.

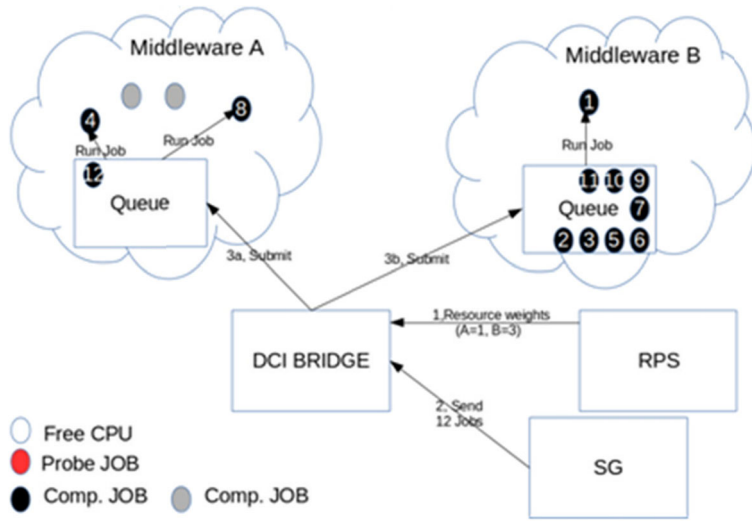


**Fig. 20** Presence of third party jobs

Since science gateways inherently work with various infrastructures, previous works are hard to apply. Scheduling among these diverse infrastructures need to understand resource utilization and handle varying resource load. In this work we propose a solution that is capable of executing many parameter study job instances over several diverse DCIs.

## 7 Conclusion

Managing heterogeneous DCIs and scheduling parameter study jobs of scientific workflows are also difficult problems and require sophisticated approaches. In this paper we proposed a meta-brokering framework for science gateways that use the push model for job submission in order to support the efficient execution of parameter sweep jobs and workflows containing this type of nodes. We apply a dynamic, weighted job instance distribution among the available infrastructures, and rely on resource priority services to cope with the high uncertainty and unpredictable load of the utilized infrastructures. Our evaluation showed that this approach implies more efficient distribution of job instances among the available computing resources than the random DCI selection algorithm resulting in shorter makespan for parameter study workflows.

Nevertheless there is still room for further improvements. Various science gateways and user communities may have different needs, therefore own metrics could be defined for custom resource priority services. In this approach we only took into account the measured waiting times by the priority service, which works well for DCIs having homogeneous resources. We encourage the development of additional methodologies and algorithms to this framework by interested research groups addressing more heterogeneous DCIs and specific workflows.

Notice that the proposed solution can be used for any science gateway that uses the OGF BES job submission standard. Such gateways can easily be extended with the DCI Bridge and the resource priority services. Since DCI Bridge supports every major DCI types and the internal resource selection algorithm of the resource priority services can easily be changed the joint usage of these two services provide a very flexible framework to distribute parameter sweep jobs among various infrastructures using the major DCI types.

In order to demonstrate the practical usage of these principals, WS-PGRADE/gUSE, one of the most widely used science gateway frameworks, was extended with this concept. From gUSE version 3.7.1, all the WS-PGRADE/gUSE gateways contain the meta-broker support. Meta-brokering is further investigated in the VIALACTEA project where PBS cluster with different capacity provide the underlying computational infrastructure for the project.

**Acknowledgments** The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 607380 (VIALACTEA), no. 312579 (ER-FLOW) and no. 608886 (CloudSME). This paper is a significantly revised and extended version of the workshop paper [22].

## References


1. Wiggins, A.: Success-Abandonment-Classification workflow at myExperiment. Online: http://www.myexperiment.org/workflows/140.html (2012)
2. SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs (SHIWA) EU FP7 project.Online: http://www.shiwa-workflow.eu/ (2012)
3. Building a European Research Community through Interoperable Workflows and Data (ER-flow) Eu FP7 project. Online: http://www.erflow.eu/ (2013)
4. Rubio-Montero, A.J., Huedo, E., Castejon, F., Mayo-Garcia, R.: GWpilot: Enabling multi-level scheduling in distributed infrastructures with GridWay and pilot jobs. Fut. Gener. Comput. Syst. **45**, 25–52 (2015)
5. Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., Gonzalez-Garcia, J.L., Roblitz, T., Ramirez-Alcaraz, J.M.: Multiple workflow scheduling strategies with user run time estimates on a grid. Journal of Grid Computing (2012)
6. Oprescu, A., Kielmann, T.: Bag-of-tasks scheduling under budget constraints. CloudCom, 351–359 (2010)
7. Silberstein, M., Sharov, A., Geiger, D., Schuster, A.: GridBot, execution of bags of tasks in multiple grids. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC '09) (2009)
8. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, Á., Karóczkai, K., Márton, I.: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities. J. Grid Comput. **10**(4), 601–630 (2012)
9. Lee, C.B., Schwartzman, Y., Hardy, J., Snavely, A.: Are user runtime estimates inherently inaccurate? Springer LNCS **3277**, 253–263 (2005)
10. Fan, Y., Pamidighantam, S., Smith, W.: Incorporating job predictions into the SEAGrid science gateway. ACM, NY, USA (2014)
11. Resource Priority Service for gUSE. Online: http://sourceforge.net/projects/priorityservice.guse.p/ (2015)
12. Ullman, J.D.: NP-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975)
13. Schwiegelshohn, U., Tchernykh, A., Yahyapour, R.: Online scheduling in grids. 22nd IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008), pp. 1–10 (2008)
14. Casanova, H., et al.: Heuristics for scheduling parameter sweep applications in grid environments. Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th. IEEE (2000)
15. Muthucumaru, M., Ali, S., Siegal, H.J., Hensgen, D., Freund, R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Heterogeneous Computing Workshop, 1999. (HCW'99) Proceedings, pp. 30–44. IEEE (1999)
16. Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Scheduling strategies for optimal service deployment across multiple clouds. Future Generation Computer Systems, doi:10.1016/j.future.2012.01.007 (2012)
17. Kertesz, A., Kacsuk, P.: GMBS: A new middleware service for making grids interoperable. Fut. Gener. Comput. Syst. **16**, 542–553 (2010)
18. Assuncao, M.D., Buyya, R., Venugopal, S.: InterGrid: A case for internetworking islands of grids. Concurrency and Computation: Practice and Experience (CCPE) (2007)
19. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Lect. Notes Comput. Sci. Algorithm. Architectures Parallel Process. **6081** (2010)
20. Buyya, R., Ranjan, R.: Special section: Federated resource management in grid and cloud computing systems. Fut. Gener. Comput. Syst. **26**, 1189–1191 (2010)
21. Kertesz, A., Maros, G., Dombi, J.D.: Multi-job meta-brokering in distributed computing infrastructures using pliant logic. In: Proceedings of the 22th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP'14), pp. 138–145. IEEE CS, Turin, Italy (2014)
22. Karoczkai, K., Kertesz, A., Kacsuk, P.: Brokering solution for science gateways using multiple distributed computing infrastructures. In: 7th International Workshop on Science Gateways (IWSG), pp. 28-33, 3–5 (2015). doi:10.1109/IWSG.2015.12
23. Bacso, G., Kis, T., Visegradi, A., Kertesz, A., Nemeth, Z.S.: A set of successive job allocation models in distributed computing infrastructures (2015). doi:10.1007/s10723-015-9347-6
24. DCI-Bridge User Manual. http://sourceforge.net/projects/dcibridge/files/DCI-Bridge-3.5.2/Documentation/DCIBridgeManualv3.5.2.pdf (2015)
25. Korkhov, V., Krefting, D., Kukla, T., et al.: Exploring workflow interoperability for neuroimage analysis on the SHIWA platform. J Grid Comput. **11**, 505 (2013). doi:10.1007/s10723-013-9262-7
26. Kiss, T.: Science gateways for the broader take-up of distributed computing infrastructures. J Grid Comput. **10**, 599 (2012). doi:10.1007/s10723-012-9245-0
27. Liu, J., Pacitti, E., Valduriez, P., et al.: A survey of data-intensive scientific workflow management. J Grid Comput. **13**, 457 (2015). doi:10.1007/s10723-015-9329-8
28. Kacsuk, P. (ed.): Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities, Springer. pp. 301 (2014)