Towards a scientific workflow-oriented computational World Wide Grid.

Peter K. Kacsuk[1]
Tamas Kiss[2]

[1] MZA SZTAKI, 1113 Kende u. 13, Budapest, Hungary
[2] School of Informatics, University of Westminster

This is a reproduction of CoreGRID Technical Report Number TR-0115, December 18, 2007 and is reprinted here with permission.

The report is available on the CoreGRID website, at:

http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0115.pdf

European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Towards a scientific workflow-oriented computational World Wide Grid

Peter Kacsuk
MTA SZTAKI 1111 Budapest, Kende u. 13
kacsuk@sztaki.hu
www.lpds.sztaki.hu

Tamas Kiss
Centre for Parallel Computing, University of Westminster
115 New Cavendish Street, London, W1W 6UW
kisst@wmin.ac.uk

CoreGRID Technical Report
Number TR-0115
December 18, 2007

Institute on Grid Systems, Tools and Environments

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Towards a scientific workflow-oriented computational World Wide Grid

Peter Kacsuk
MTA SZTAKI 1111 Budapest, Kende u. 13
kacsuk@sztaki.hu
www.lpds.sztaki.hu

Tamas Kiss
Centre for Parallel Computing, University of Westminster
115 New Cavendish Street, London, W1W 6UW
kisst@wmin.ac.uk

CoreGRID TR-0115
December 18, 2007

Abstract

This paper describes a potential roadmap establishing a scientific, workflow-oriented, computational World Wide Grid (WWG). In order to achieve such a WWG the paper suggests three major steps. First, create uniform meta-brokers and connect existing production Grids by them in order to form the WWG infrastructure where existing production Grids become interoperable at the job submission level. Second, create workflow-oriented advance Grid portals and connect them to the meta-brokers in order to exploit workflow level Grid interoperability. Finally, create workflow Grid services and their registry and repository in order to make the workflows developed by different workflow communities interoperable and shareable.

1 Introduction

The goal of this document is to assess where we are in the road of establishing a scientific, workflow-oriented, computational World Wide Grid (WWG) that is similar to the World Wide Web (WWW) in the sense that anyone can access and use its services according to his needs. If we look at the current trend of how and where Grid develops we can see that isolated production Grids have been created that are based on different Grid technologies that are not interoperable or only in a limited way. The primary aim of our research is to define how these isolated production Grids can be seamlessly connected to a world wide Grid architecture in a user transparent way. The WWG in this sense is the federation of the existing production Grid islands that give access to a much larger scale of resources then one particular Grid can. The requirements towards a generic WWG are derived from the features of the WWW. However, the implementation and the roadmap of the WWG are influenced by the current state of Grid development.

This paper is intended to show that we are not far from creating a WWG if we make reasonable assumptions how the users would like to use a WWG and reasonable restrictions how such a WWG can be used and for what purposes. The basic assumptions are as follows:

1. We restrict ourselves for a computational Grid type WWG

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

2. The goal of the user is to dynamically collect and use as many Grid resources as possible to accelerate his Grid application

3. The basic application type the user would run in the WWG is a complex workflow

4. The WWG will be used in the beginning by the scientific community

Assumption 1 says that we restrict the first version of the WWG to a computational Grid where the size of files used in the WWG are not too large and hence they can efficiently be moved between computational Grid resources. It does not exclude the usage of large files in the WWG but it significantly reduces the overall performance of the WWG if many users try to use large files. Obviously, in a longer term the efficient management of large files should also be solved in the WWG but since our goal is to show that a computational WWG is already feasible we neglect this problem in this document. A follow-up document should be written to outline the possible solutions for a data-oriented WWG.

Assumption 2 requires that all possible parallelisms of an application should be exploited in the WWG. Users go for the WWG to access and use many resources in parallel to speed up the execution of their complex applications. Section 1 will classify the types of parallelisms that can be achieved in the WWG and later it will be shown how such parallelisms can actually be utilized in the WWG.

Assumption 3 might require some more explanations then the first two assumptions. Why workflow applications are so interesting for the WWG? The workflow concept abstracts a collection of services (tasks) that can be executed in a partially ordered way and hence it is general enough to contain as a special case any other types of applications. So workflows could be considered as the most general type of applications including as special cases any other types.

Assumption 4 is based on the current usage scenario of large e-science Grids. In order to establish a WWG that is used by the population and for business it requires a significant improvement in the development of a Grid market model. If only the scientific community will use the first WWG, then the Grid market model could be much simpler than a real commercial one. Since our interest is to establish a WWG as soon as possible, it is better to start with a scientific WWG and later extend it to other directions.

At this point many readers could say that these assumptions are too restrictive and it is not worth defining and creating a WWG that can support only these kinds of applications and goals. We would argue that we cannot create at once the ultimate WWG. The WWW is much more complex today as it was in its initial stages and in the beginning it was used only for scientific purposes and only later it was extended towards the commercial world. Once we established an infrastructure that is useful and works afterwards there are plenty of opportunities to improve and extend that system in the future. Even this restricted version of the WWG that is suggested in this document can be used to support much more applications than we can dream today. The establishment of such a WWG could tremendously widen the user community of the Grid and would significantly accelerate the take-up of Grid technology world-wide and would lead later to a WWG usable for the whole population including commercial services as well.

Before starting to explain the technical details of creating such a WWG, it is important to compare the WWW concept and the proposed WWG concept. First of all, let's see why the WWW concept is so popular and successful. There are five main aspects that make WWW so attractive:

1. Services

2. User interface

3. Web search engines

4. Security

5. Interest to use

The original WWW concept was based on the web page services. The idea is that anyone can create and publish web pages and anyone from any client machine can access these published web pages. Another important concept here is that web pages can be linked together and this significantly facilitates the creation and usage of web pages.

The second appealing feature of the WWW is that its user interface is extremely simple and easy-to-use. A simple browser is sufficient to browse web pages no matter where these web pages were created. More than that these browsers are provided as part of the basic services of the client machines' operating systems and hence the user does

not have to install any complicated WWW access software, it comes together with the client machine. Web portals help the users to access structured information over a set of web pages.

Web search engines help the users to discover information in the Web. In fact the Web search engines provide the web information system by discovering relevant web page contents.

The HTTP and HTTPS protocols provide the necessary security mechanism. They require only a single port to open on the server machine and hence they can be securely managed. The security concept of the WWW is so reliable that even large banks trust this system and provide financial services through their web portals.

The final aspect of the WWW is the motivation of people to use it or to provide services by it. The WWW is an excellent way of creating communities, accessing information and special (e.g. financial) services and hence people are interested in using the WWW services. In the beginning when commercial exploitation of the WWW was not so apparent people were interested in creating web pages because in this way they could increase their or their company's visibility. Later when it became clear that there are several business models by which companies can make profit by providing WWW services, the usage of the WWW became even more popular.

After the overview of the main aspects of the WWW let's see how these aspects can be used to promote the WWG also as a success story. In the case of the WWG the services are Grid resources (computing services, data services, etc.) or higher level Grid services (for example, workflow execution service). It is important that anyone should be able to provide Grid services and anyone should be able to access these Grid services from any client machine through the WWG. The same way as web pages can be linked together Grid services should be linked together.

The user interface should be in the same way extremely simple and user-friendly as it is in the case of the WWW. Simple tools like a web browser should be available on every client machine in order to access the WWG services without installing any Grid software on the users' machines. However, in the WWG the main objective is to run applications and hence instead of a simple browser, rather a simple application builder is the right GUI. Grid portals should help the users to access the WWG services in a coordinated way releasing the user from the actual organization of resource collection and orchestration via the WWG.

Grid search engines similar to the Web search engines should help both the users and Grid services to discover relevant Grid services in the WWG.

The current Grid security mechanism is built on the concept of Grid certificates and VOs. Although scientific papers always emphasize the importance of creating dynamic VOs in practice VOs are quite static and their creation is a long procedure. This static nature of VOs is one of the reasons why Grid developed towards the isolated Grids direction and not towards the WWG direction. The current certificate and VO scheme make the usage of the Grid much more complicated than the usage of the Web. As a consequence in this respect some revision is necessary if we want to make the WWG really easy-to-use and popular.

Finally, the motivation of the usage of WWG should be made tempting for large user and Grid service provider communities. Once the usage of WWG is simple enough it will be really attractive for a large user community to access Grid resources and services in an "unlimited" way. This will raise a new problem. If there is no limit of accessing resources and services the whole WWG will collapse due to the huge demand of resources and services. A WWG market model is therefore unavoidable and obligatory from the very beginning in order to attract resource and service providers and to restrict the eagerness of WWG consumers in acquiring resources and services for their applications. A kind of WWG credit system must be introduced where resource and service providers can earn WWG credits and then their community can use these credits to acquire WWG resources and services.

Table 1 summarizes and compares the five main features of the existing WWW and a potential WWG.

If there are so many similarities in the concept of WWW and WWG, then why we still miss the WWG as a working infrastructure and service? Unfortunately, there are some problems concerning with all the five required features of the WWG.

Services and resources

There are many different production Grids based on different Grid technologies and middlewares and they can not interoperate. As a result the services are not accessible by anyone from anywhere as it would be needed by the WWG concept proposed above. If a user is registered for the VOX of GridA then he cannot use the resources and services of VOY of GridB. Chapter 1 of this study will show that based on assumptions 1-3 we can easily solve this problem and create a WWG that satisfies the required criteria.

User interface

	WWW	WWG
Services Anyone can create and publish anyone can access	Web pages, web services	Grid resources, computing services data services, etc.
User interface	Web browsers Web portals	Grid application builder Grid portals
Information discovery mechanism	Web search engines	Grid search engines
Security	HTTP and HTTPS protocols	Revised dynamic VO concept
Interest to join	User: access information and services Provider: increase own visibility, make money	User: use Grid resources (by Grid credit) Provider: collect Grid credits (later make money)

Table 1: Summarizes and compares the five main features of the existing WWW and a potential WWG.

The Grid user interface is currently too complicated. In most cases production Grids neglect the problem of user interface. They provide only a special command line interface and programming API. It means that there is no user interface standard like the web browser in case of the Web, different Grid middlewares require the usage of different command line interfaces and programming APIs. It means that a user who wants to use several Grids (an obvious assumption of the WWG concept) has to learn several Grid user interfaces. If the user wants to port a Grid application from GridA to GridB he has to re-engineer the application according to the programming API of GridB.

Information discovery mechanism

The Grid information system and discovery mechanism is not mature enough, the concept of Grid search engine is missing in the current Grids. The usage of the information system is very limited in the current production Grids.

Security

The Grid security mechanism is suitable for the rather static VO concept of the current production Grids but not for the WWG where VOs should be formulated dynamically on demand. As a consequence the concept of VOs should be revised in the framework of WWG.

Interest to join

Since the current usage of Grid lacks the market concept (everyone can get what he needs without payment) a WWG would lead to the tremendous overload of resources. At least the introduction of a simplified Grid market concept would be necessary to establish a scientific WWG.

The following sections of this study will describe in detail these features and will show the possible solutions that can be used in order to establish a scientific computational WWG. Obviously, when the goal is to create a WWG as soon as possible any proposed solution should be built according to the current situation. It should be accepted as a fact that production Grids already exist and they are not willing to give up their freedom of developing their own roadmap and direction. Therefore a WWG concept should be built on the autonomicity and collaboration of these existing production Grids.

2 Services and Grid middleware support

As we have seen in the Introduction Assumption 2 says that the goal of the user is to

1. dynamically collect and use as many Grid resources as possible
2. in order to accelerate his Grid application.

Condition 1. means that the user needs a WWG where resources can be accessed from any production Grids that are connected to the WWG no matter what type of Grid middleware they built on. As a result current production Grids

should be used in an interoperable way even if they are based on different Grid middlewares. When this problem is solved any user from any Grid can access all the Grid resources and services that are connected to the WWG.

Condition 2. requires that both the application and the WWG should be able to support the largest possible classes of parallel execution.

In this section first we examine the impact of parallelism on the requirements of the Grid middleware and then investigate the problem of Grid interoperability. We shall see that both conditions can be fulfilled by the introduction of interoperable Grid brokers or by the introduction of a meta-broker.

2.1 Parallelism in the WWG

There are two classes of parallelisms achievable in the WWG:

1. Grid architecture parallelism
2. Grid application parallelism

The Grid architecture parallelism can be further divided into two classes according to the usage of various Grid resources:

- **Inter-Resource (IrR) parallelism** (parallel usage of several resources) within which we can distinguish:

  Inter-Grid (IrG) parallelism (parallel usage of several resources within several Grids)

  Intra-Grid (IaG) parallelism (parallel usage of several resources in the same Grid)

- **Intra-Resource (IaR) parallelism** (usage of parallelism in a single resource having parallel architecture, e.g. cluster)

The current Grids typically enable the exploitation of the Intra-Grid and Intra-Resource parallelism but they do not support Inter-Grid parallelism. Even worst, within the same Grid, users are restricted to use the resources only of a certain VO where they are accepted as members. The current concept of VOs is strongly against the nature of WWG.

We can distinguish four types of Grid application parallelism according to the granularity of tasks to be solved in the Grid:

- Single job/service level (SJ)
- Parameter Sweep at job/service level (PSJ)
- Workflow level (WF)
- Parameter Sweep at workflow level (PSWF)

Intra-Resource parallelism can be applied for any types of Grid application parallelism if the resource is a multi-processor one and the local job manager is able to distribute the parallel components of the application among the processors of the resource.

Single job level parallelism (SJ) can be exploited if the application consists of one job and this job is a parallel (e.g. MPI) one. In this case we can explore process parallelism that comes from the parallel execution of processes of the parallel application. Although there are some research projects aiming at the exploitation of Inter-Resource parallelism, SJ parallelism still best fits to the Intra-Resource parallelism where processes of a parallel application can be distributed among the nodes of a parallel resource. If there are N processes inside a parallel job, then the achievable parallelism is O(N).

PS job level parallelism (PSJ) can be exploited if the application consists of one job and this job should be executed with many different parameter sets (this is called job instance parallelism). PSJ parallelism fits both to Intra-Resource and Inter-Resource parallelism no matter whether the job is a sequential or parallel one. In the case of a sequential job Intra-Resource parallelism can be exploited by allocating many instances of the same job to a multiprocessor resource and the different job instances are simultaneously executed by different processors of the

resource. If a sequential job is to be executed with M parameters, then the achievable parallelism is O(M). If the job is a parallel application with N processes and this should be executed with M parameters, then both process parallelism and job instance parallelism can be exploited and the achievable parallelism is O(MxN). Since both N and M can be in the range of hundreds or thousands PSJ parallelism can require several thousands of resources and hence it really needs the large number of resources available in the WWG.

Workflow level parallelism (WF) can be exploited if there are parallel branches in the workflow. This is called **workflow branch parallelism** and it fits both to Intra-Resource and Inter-Resource parallelism. However, the amount of parallelism is typically not as big as in case of the PSJ parallelism so in many cases Intra-Grid parallelism is enough to handle it. If some of the jobs in the workflow are parallel (e.g. MPI) jobs, then two levels of parallelism can be exploited simultaneously: process parallelism and workflow branch parallelism. Different MPI jobs of the different branches of the workflow can be simultaneously executed on different resources (Inter-Resource parallelism) and on each such resource Intra-Resource parallelism can be applied for the parallel execution of the processes of the MPI jobs. If the maximum number of parallel branches in the workflow is B and in every branch there is a parallel application with N processes, then the achievable parallelism is O(BxN).

PS workflow level parallelism (PSWF) can be exploited if the application consists of a workflow and this workflow should be executed with many different parameter sets (this is called workflow instance parallelism). In such case three levels of application parallelism can be exploited:

- Workflow instance parallelism (among the several instances of the workflow)
- Workflow branch parallelism (among the branches of every workflow instance)
- Process parallelism (among the processes of a parallel node of a workflow instance)

Notice that job instance parallelism is a special case of workflow instance parallelism when the workflow consists of a single job. From another point of view workflow instance parallelism is a sum of achievable job instance parallelism when the component jobs of a workflow are executed with many parameters sets.

In order to exploit this three levels of parallelism, PSWF parallelism can require thousands or even millions of resources and hence it really needs the large number of resources available in the WWG. PSWF parallelism fits to the Inter-Resource parallelism (both IaG and IrG) and as a result can advantageously be used in the WWG. If the maximum number of parallel branches in the workflow is B and in every branch there is a parallel application with N processes, and the workflow should be executed with M different parameter sets, then the achievable parallelism is O(MxBxN). This is clearly the most demanding type of parallel application concerning the number of required Grid resources.

2.2 Resource selection to achieve the highest possible parallelism

After seeing that many resources should be used in a PSJ or PSWF application the next question is how to select the required resources in order to achieve the highest possible parallelism. First we consider the single Grid models and then we generalize the concept for the WWG.

2.2.1 Single Grid model

The single Grid model means that the user can access the resources and services only in a single Grid. Within a single Grid there are two options to select a resource:

- User selected Resource (UR)
- Broker selected Resource (BR)

In the UR model it is the user who has to select the required and best fitting resources and services for running his application in the Grid. This requires a certain skill from the user and a great care. If the resources and services are selected in a wrong way it can result in significant performance degradation. Unfortunately, there are production Grids where brokers are not available and the user should follow this model. Lets' suppose that the users are smart enough to

	User selected Resource (UR)	Broker selected Resource (BR)
SJ level process parall.	IaR, unbalanced	IaR, balanced
PSJ level Instance parall Process parall	IaR unbalanced	IaR and IrR, balanced
WF level Branch parallelism Process parall.	IaR and IrR, unbalanced	IaR and IrR, balanced (if there is WF-scheduling)
PSWF level Instance parall. Branch parall. Process parall.	IaR and IrR, unbalanced	IaR and IrR, balanced

Table 2: Achievable Grid architecture parallelism within a single Grid

select the most optimal resource for their applications. Even in that case many users can select the same resource at the same time and this can result in an unbalanced usage of Grid resource leading to the overall performance degradation of the Grid.

The BR model significantly facilitates the usage of the Grid by taking over the responsibility of selecting the right Grid resources and services. The broker accesses the Grid information system and based on the stored information it can take smart decisions to select the optimal resources and services for a particular application in a particular time. It means that both the static characteristics of the resources and their dynamic load and behaviour can be taken into consideration by the broker. Therefore using the BR model has several main advantages compared to the UR model:

1. the task of the user is significantly reduced
2. the resource selection algorithm could be much more optimal than the user driven one
3. it can result in a well balanced load distribution among the Grid resources and services

Table 2 summarizes how the UR and BR models can support the various forms of Grid architecture parallelism in the case of the different Grid application parallelism types. It also shows whether balanced resource usage can be achieved. Notice that in all cases the IrR (inter-resource) parallelism realized by the class of IaG (Intra-Grid) parallelism since there is only a single Grid. Now let us examine in detail the support of SJ, PSJ, WF and PSWF parallelism by the UR and BR model.

SJ parallelism
No matter if the UR or BR method is used no Inter-Resource (IrR) parallelism can be achieved. The only difference is that in case of the BR method a smart broker can manage a well balanced usage of the resources. Intra-Resource (IaR) parallelism can be achieved in both the UR and BR cases if the job is a parallel application. In such case process parallelism can be exploited on multiprocessor resources.

PSJ parallelism
In case of the UR model no Inter-Resource (IrR) parallelism can be achieved since the user can not select resources individually for each input parameter set. Whatever resource is selected by the user for the job it will be used all the time for every job instance. (Of course the user can develop a PS job submitter script that somehow distributes the job among the Grid resources but it means that the user should develop Grid middleware.) In the BR case Inter-Resource (IrR) parallelism automatically provided by the broker. In principle all resources of the Grid can be used in parallel and in a balanced way. Intra-Resource (IaR) parallelism can be achieved in both the UR and BR cases even if the job is a sequential one provided that the resource (e.g. a cluster) has got multiple nodes and the local job manager can distribute the incoming job instances among the nodes of the resource. If the job is a parallel application, then process parallelism can be easily exploited on the multiprocessor resources.

WF parallelism
In case of the UR model the user can easily achieve Inter-Resource (IrR) parallelism by allocating different resources to the nodes of the workflow that are placed on parallel branches (exploitation of workflow branch-parallelism). In the BR case the broker requires workflow level scheduling support (which is not available in most cases) in order to allocate different resources for the different branches of the workflow. Intra-Resource (IaR) parallelism can be achieved in both the UR and BR case either if the job is a parallel one (exploiting process parallelism) or if several branch parallel jobs are allocated to the different nodes of the same resource.

PSWF parallelism
In case of the UR model the user can exploit the branch-parallelism but can not exploit the workflow instance parallelism since the same resource allocation will be applied for every workflow instance no matter which input parameter set they use. As a result the Grid resource usage will be very unbalanced and the achievable parallelism will be significantly limited. In the BR case the broker can easily exploit the workflow instance parallelism. More than that during the PSWF execution workflow level scheduling support is not needed for the broker since balanced execution can be achieved via the parallel workflow instance execution. Intra-Resource (IaR) parallelism can be achieved in both the UR and BR cases as shown for the PSJ and WF parallelism.

2.2.2 WWG model

Now let's see how resource/service selection can be supported in the case of using many interconnected Grids, i.e., in the case of the WWG system. In such system four models can be distinguished:

- User selected Grid User selected Resource (UGUR)
- User selected Grid Broker selected Resource (UGBR)
- Broker selected Grid User selected Resource (BGUR)
- Broker selected Grid Broker selected Resource (BGBR)

All these models raise the problem of Grid interoperability and Grid info system is needed both at the Inter-Grid and Intra-Grid level in all the four models. Notice that the BGUR model makes no sense to use: if a broker selected a Grid, this broker or another broker should be able to select the resources, too. Therefore we do not study further the BGUR case.

The UGUR model means that the users can access several Grids simultaneously and these Grids have no broker. The user should select first a Grid and then a resource inside that Grid for every job or service as shown in Fig. 1.

The UGBR model means that the users can access several Grids simultaneously and these Grids have got broker. The user should select the Grid but not the resources inside the selected Grid. The user submits the jobs through the Grid broker that selects the right resources according to the user specification given by a resource specification or job description language like RSL [1], JDL [2] or JSDL [3] depending on the actual Grid. The architecture of the UGBR model can be seen in Fig. 2.

The BGBR model means that the users can access several Grids simultaneously and these Grids have got broker. However, the user is not connected directly to a Grid broker rather to a new level of brokers called as meta-broker. It is the task of the meta-broker to select the right Grid according to the user requirements that is described by the Broker Property Description Language (BPDL) [4]. This is similar to the JSDL, but it provides metadata about brokers (Grids) and not about resources. Once the Grid is selected the meta-broker passes the job and the job description language (RSL, JDL or JSDL depending on the actual Grid) to the selected Grid broker and it will be the task of this broker to select the best resource according to the requirements specified in the job description language (or sometimes called resource specification language). The architecture of the BGBR model can be seen in Fig. 3.

The proposed Meta-Broker architecture is described in detail in [4]. In order to make the concept of meta-broker clear a short overview of its architecture concept is given here. Fig. 4 introduces the proposed architecture of the Meta-Broker that enables the users to access resources of different Grids through their own brokers. The Translator components are responsible for translating the resource specification language of the user (JSDL) to the language of the selected resource broker. Once a broker will be capable of supporting the JSDL standard, the corresponding Translator can be removed from the Meta-Broker. The Invokers are broker-specific components. They communicate with the interconnected brokers, invoking them with job requests and collecting the results. Data handling is also

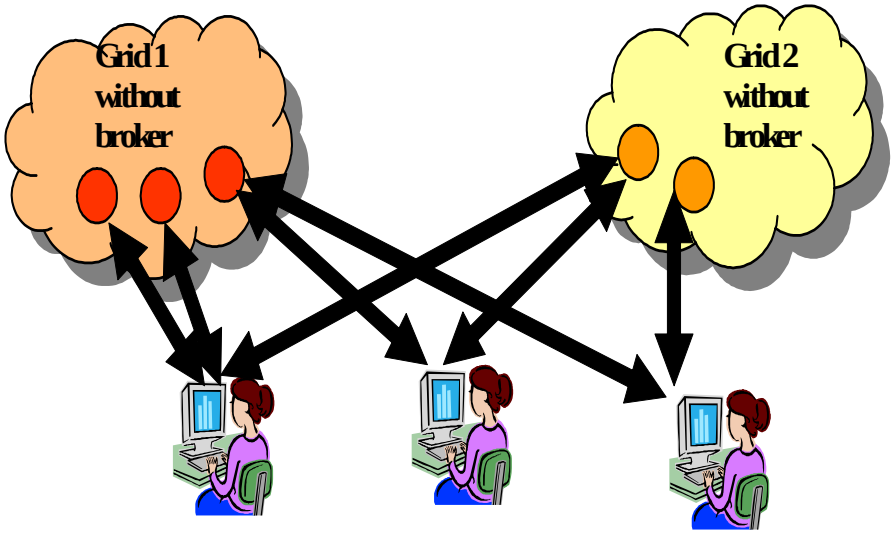

Figure 1: UGUR model

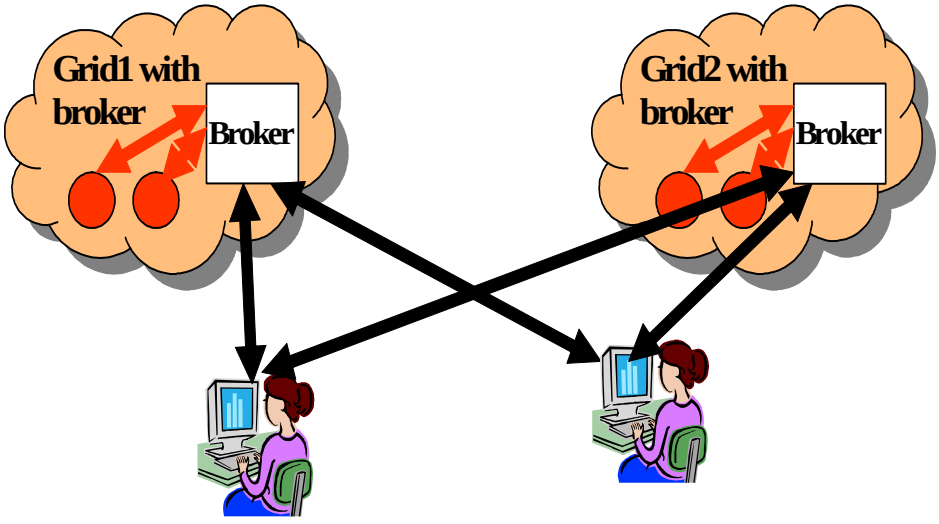

Figure 2: UGBR model

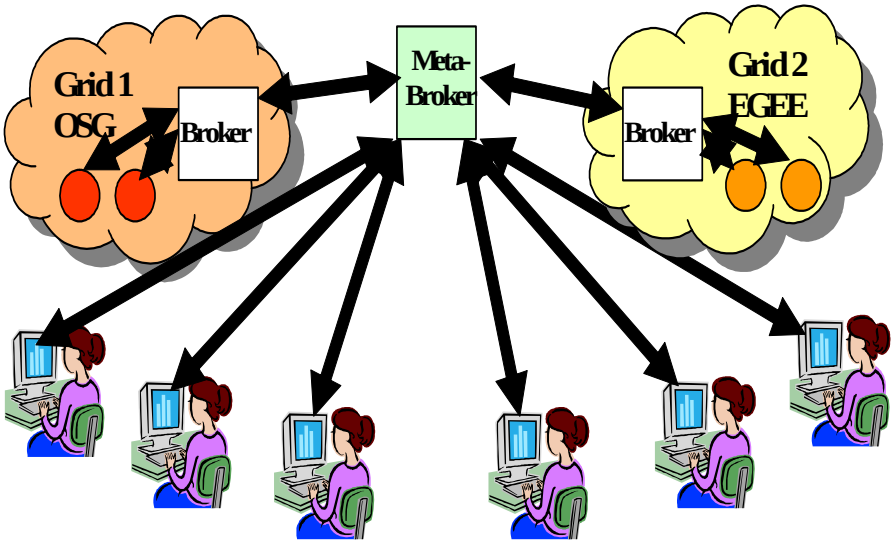

Figure 3: BGBR model

an important task of this component. After the user uploaded the job, proxy and input files to the Meta-Broker, the Matchmaker component tries to find a proper broker for the request. If no good broker was found, the request is rejected, otherwise the JSDL is translated to the language of the selected broker. The responsible Invoker takes care of transferring the necessary files to the selected Grid environment. After job submission it stages back the output files, and upgrades the historical data stored in the Information Collector with the log of the utilized broker. The core component of the Meta-Broker is responsible for managing the communication (information and data exchange) among the other components. The communication to the outer world is also done by this part through its web-service interface.

Table 3 summarizes how the UGUR, UGBR and BGBR models can support the various forms of Grid architecture parallelism in the case of the different Grid application parallelism types. It also shows whether balanced resource usage can be achieved. Now let us examine in detail the support of SJ, PSJ, WF and PSWF parallelism by the UR and BR model.

SJ level parallelism

UGUR case

The UGUR case is the same as the UR case in single Grids except that first the user should select a Grid.Obviously the chance for load-balancing between Grids and inside Grids is much lower than in the broker selected case.

UGBR case

Inside a Grid the UGBR case is the same as the BR case in single Grids but first the user should select a Grid. Therefore there is no load-balancing between Grids but inside a Grid load-balancing can automatically be achieved by the broker.

BGBR case

In the BGBR case both the Grids and the resources are selected by brokers and hence load-balancing can be achieved between Grids and inside Grids.

PSJ level parallelism

UGUR case

Since it is the task of the user to select a Grid and resource all the job instances are executed on the same resource and hence no Inter-Resource (IrR) parallelism can be exploited. Intra-Resource (IaR) parallelism can be achieved in the

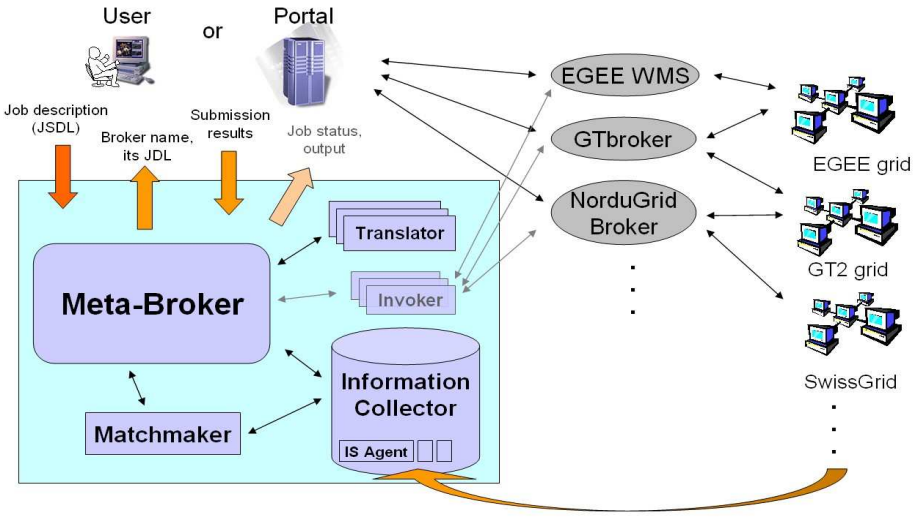

Figure 4: The Meta-Broker Architecture

	UGUR	UGBR	BGBR
SJ level process parall.	IaR; unbalanced between Grids and inside Grids	IaR; unbalanced between Grids balanced inside Grids	IaR; balanced between Grids and inside Grids
PSJ level instance parall. process parall.	IaR, unbalanced; NO IaG No IrG	IaR and IrR; balanced IaG No IrG	IaR and IrR; balanced IaG balanced IrG
WF level Branch parallelism Process parall.	IaR and IrR; unbalanced IaG unbalanced IrG	IaR and IrR, balanced IaG (if there is WF scheduling), unbalanced IrG;	IaR and IrR; balanced IaG balanced IrG (if there is WF scheduling)
PSWF level Instance parall. Branch parall. Process parall.	IaR and IrR; unbalanced IaG unbalanced IrG; No WF instance parallelism	IaR and IrR; balanced IaG, unbalanced IrG; WF instance parallelism at IaG	IaR and IrR; balanced IrG balanced IaG ; WF instance parallelism at IrG and IaG

Table 3: Achievable parallelism by differentwithin a single Grid

same way as in the UR case within a single Grid.

UGBR case

Since it is the task of the user to select a Grid only one Grid can be used and hence there is no Inter-Grid (IrG) parallelism. (Of course, user developed job submitter script is possible to exploit Inter-Grid parallelism but it means that the user should develop Grid middleware.) Resource selection is done by a broker inside the selected Grid and hence inter-resource (IrR) Intra-Grid (IaG) parallelism is automatically provided in a balanced way by the broker.

BGBR case

Both Grid and resource selection is done by a broker and hence both Inter-Grid (IrG) and Intra-Grid (IaG) parallelism are automatically provided by brokers. In principle all resources of all connected Grids can be used in parallel and in a balanced way.

WF level parallelism

UGUR case

Grid selection is the task of the user and hence it is the user's responsibility to explore the Inter-Grid (IrG) parallelism through the workflow branch-parallelism. Similarly, resource selection is done by the user and hence it is also the user's responsibility to explore Intra-Grid (IaG) parallelism through workflow branch-parallelism and to explore Intra-Resource (IaR) parallelism by assigning parallel jobs of the workflow to parallel Grid resources. Since many users can select the same Grid and the same resources at the same time that can result in the unbalanced usage of Grids and resources and can lead large response time for the individual users.

UGBR case

Grid selection is the task of the user and hence it is the user's responsibility to explore the Inter-Grid (IrG) parallelism through the workflow branch-parallelism. Inside the user selected Grid resource selection is the task of the broker and hence it is the broker's responsibility to explore Intra-Grid (IaG) parallelism through workflow branch-parallelism and to explore Intra-Resource (IaR) parallelism by assigning parallel jobs of the workflow to parallel Grid resources. Notice that workflow scheduler support is needed for the broker in order to be able to exploit workflow branch-parallelism at IaG level. Since many users can select the same Grid at the same time this can result in the unbalanced usage of Grids and can lead large response time for the individual users.

BGBR case

Grid selection is the task of the broker and hence it is the broker's responsibility to explore the Inter-Grid (IrG) parallelism through the workflow branch-parallelism. Resource selection is also the task of the broker and hence it is the broker's responsibility to explore Intra-Grid (IaG) parallelism through workflow branch-parallelism and to explore Intra-Resource (IaR) parallelism by assigning parallel jobs of the workflow to parallel Grid resources. Since both the Grid and the resources are selected by brokers, load balancing can be achieved between Grids and inside Grids. However, workflow scheduler support is needed for the broker to exploit workflow branch-parallelism at IrG and IaG level.

PSWF level parallelism

UGUR case

Grid selection is the task of the user and hence it is the user's responsibility to explore the Inter-Grid (IrG) parallelism through the workflow branch-parallelism. Another way of exploiting Inter-Grid parallelism is through workflow instance-parallelism. Unfortunately, this is not possible without broker support so the UGUR model can not support this kind of Inter-Grid parallelism. The same is true for exploiting Intra-Grid parallelism inside the selected Grid where the resources should be selected by the user: workflow branch-parallelism can be achieved but workflow instance-parallelism can not in the UGUR model. Intra-Resource (IaR) parallelism can be exploited by the user assigning parallel jobs of the workflow to parallel Grid resources.

The main drawbacks of the UGUR model are:

- Many users can select the same Grid and the same resources at the same time that can result in the unbalanced usage of Grids and resources, and can lead to large response time for the individual users.

- Workflow instance-parallelism can be exploited neither at IrG nor at IaG level.

UGBR case

Grid selection is the task of the user and hence it is the user's responsibility to explore the Inter-Grid (IrG) parallelism through the workflow branch-parallelism. The exploitation of Inter-Grid (IrG) parallelism through workflow instance-parallelism is not possible (see the UGUR model). Intra-Grid parallelism can be exploited both through the workflow branch-parallelism and workflow instance-parallelism by the broker. Intra-Resource (IaR) parallelism can also be provided by the broker assigning parallel jobs of the workflow to parallel Grid resources.

The main drawbacks of the UGBR model are:

- Many users can select the same Grid at the same time and this can result in the unbalanced usage of Grids, and can lead to large response time for the individual users.
- Workflow instance-parallelism can not be exploited at the IrG level.

Advantages of the UGBR model are:

- Workflow instance-parallelism can be exploited at the IaG level.
- Workflow level scheduling support is not needed for the broker at the IaG level since balanced execution can be achieved via the parallel workflow instance execution

BGBR case

Both Grid and resource selections are the tasks of the broker and hence both Inter-Grid (IrG) and Intra-Grid (IaG) parallelism can be exploited both through the workflow branch-parallelism and workflow instance-parallelism by the broker. Intra-Resource (IaR) parallelism can also be achieved as described in the single Grid BR model.

Advantages of the BGBR model are:

- Workflow instance-parallelism can be exploited both at IrG and IaG level.
- Workflow level scheduling support is not needed for the broker either at IrG or IaG level.
- Balanced usage of Grids and resources are provided by the broker.

As a summary one can say that the most advantageous model to exploit every possible parallelism in a well balanced way is the BGBR model. So if we want to establish an efficient and powerful WWG it should be based on the concept of the BGBR model. Unfortunately, current Grid middlewares do not support any of the UGUR, UGBR or BGBR models. There is an on-going effort in the GIN VO of OGF to solve this problem but they concentrate at the moment on the SJ level. Within the framework of the CoreGrid project SZTAKI and the Barcelona Supercomputing Centre work on to create a meta-broker that can efficiently support the BGBR model.

2.3 Alternative middleware approach for the BGBR model

It has to be mentioned that the BGBR approach is not the only concept to solve Grid interoperability at the broker level. Within the CoreGrid Institute on Resource Management and Scheduling and the OGF Grid Scheduling Architecture RG researchers work on another concept in which brokers of different Grids are directly connected. This model can be called as the User-Transparent Grid, Broker selected Resource (UTGBR) model. UTGBR can substitute the BGBR model, i.e. everything that was written for the BGBR model is valid for the UTGBR model, too. The only difference is in the implementation approach as shown in Fig. 5.

The user connects to one Grid broker and submits the job with the job description language (RSL, JDL or JSDL depending on the actual Grid). It is the task of the Grid broker to decide whether the job is to be executed in the current Grid or it is better to pass the job and the job description language to another Grid. As a result the Grid selection is completely transparent for the user (just like in the case of the BGBR model).

If we compare the two models there are pros and contras for both. The advantage of the UTGBR model is that there is no need to develop a new broker type, the meta-broker. However, it requires the development of an internationally accepted standard protocol by which all the existing brokers can communicate to each other. It means that if such a standard is created each existing broker should be reengineered in order to extend them with this protocol. This

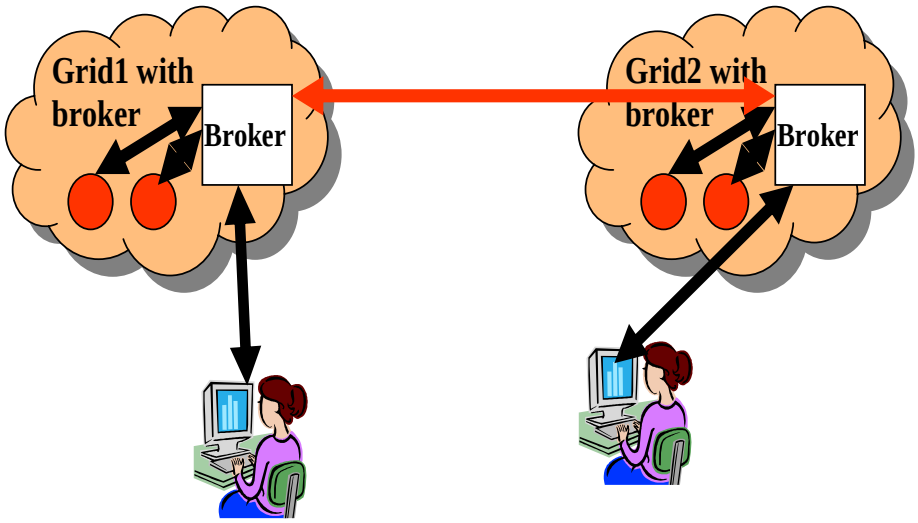

Figure 5: UTGBR Model

seems to be the smaller problem, the establishment of the standard would require a longer time. Although such a standard would be very welcomed in the case of the meta-broker architecture, too it is not unconditionally necessary. The creator of the meta-broker can handle individually the major broker types creating the necessary Translator and Invoker modules as described in [4]. Once the standard is accepted and the brokers gradually start to use the standard protocol these modules can be simply omitted from the meta-broker. So from the point of view of the fast establishment of WWG the meta-broker model seems to be more feasible.

An important drawback of the meta-broker model is that obviously a single meta-broker would be a bottle-neck in the WWG. In order to avoid this problem, large number of meta-brokers should be used in the WWG that can communicate to each other and to some Grid brokers that are directly connected to them. In such a WWG system any user can connect to the closest meta-broker as shown in Fig. 6.

In the case of the WWG the number of meta-brokers can be very large. P2P networks proved to be very efficient for creating large networks of identical networking units and hence it is likely that creating a P2P network of meta-brokers would be a suitable solution to establish the WWG. (However it obviously requires some further research to justify this claim.)

There is another important advantage of the meta-broker model compared to the UTGBR model from the point of view of accessing the WWG. Since in the BGBR model all the users are connected in the same way to one of the meta-brokers, this model is much better fits to the concept of WWG where a standardized access to the WWG is as important as the standardized access to the WWW. If all the users would access the WWG via meta-brokers, then only one uniform user interface mechanism would be needed. Any client could install exactly the same WWG client software which in this case could be part of the operating systems.

2.4 Conclusions

We have seen that the highest level of parallelism can be exploited in the WWG if workflows are executed as parameter sweep applications. In order to exploit the largest possible parallelism the most advanced architecture concept of the WWG is based on the BGBR and/or UTGBR model. These are equivalent from the point of view exploiting parallelism at the largest possible extent of load-balancing. It was shown that there are several advantages of using the BGBR model compared with the UTGBR model.

In the BGBR model every client can be connected in a standard way to one of the uniform meta-brokers. As a result if we want to build the WWG, the only thing we have to do is to define and implement:

1. The functionality of the meta-brokers

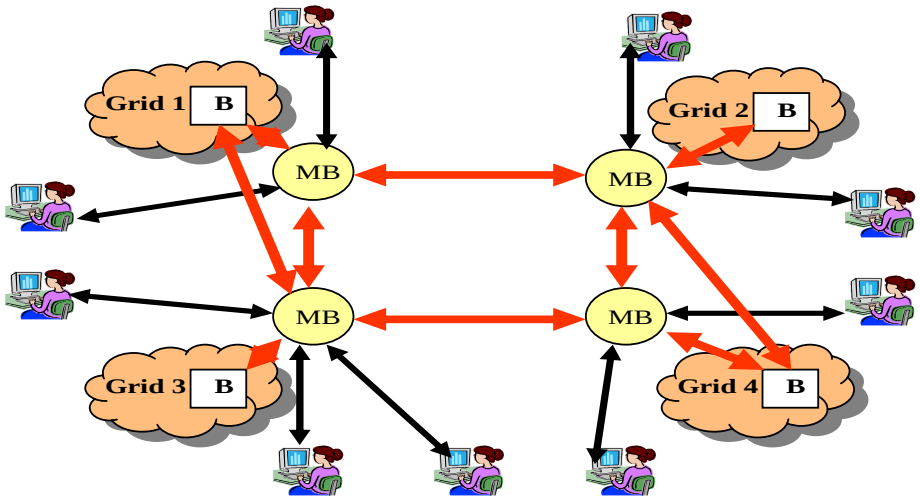

Figure 6: The architecture of the WWG based on the BGBR model

2. The intercommunication protocol of meta-brokers
3. The communication protocol of clients and meta-brokers
4. The standard intercommunication protocol of meta-brokers and Grid brokers

The solution for requirements 1 and 3 are already designed and described in [4]. Work is needed for requirement 2. The BGBR model concept can be used even if requirement 4 is not fulfilled but it would be advantages.

Once these requirements are defined and implemented, any existing Grid can be connected to the WWG provided that the broker of the given Grid realizes the standard intercommunication protocol of meta-brokers and Grid brokers. Even if requirement 4 is not fulfilled, those Grids for which the Translator and Invoker module of the meta-broker is already available can be connected to the WWG. The meta-broker should be implemented as an open source Grid service in order that any Grid could extend it with the necessary Translator and Invoker module by which the given Grid could be accessed by the meta-broker.

So overall, there is no real technical obstacle to create a scientific computational World Wide Grid where complex parameter sweep workflow applications could run and exploit the largest possible parallelism. It means that technically the WWG can be established by simply introducing the meta-broker concept and using a network of uniform meta-brokers to connect the existing production Grids. Obviously, the meta-broker itself does not help in managing workflows, only to submit nodes (jobs or service requests) of the workflow in a Grid transparent way. To assist the workflow execution in the WWG a workflow manager is needed. This will be investegated in the next chapter.

3 Grid user support

After defining how to technically create the WWG, it is time to see what we can provide at the user support level. The WWW became popular because

- its user interfaces (web browsers) are extremely easy-to-use
- anyone can access the WWW from any client machine without installing and maintaining a complicated client software.

If we want to make WWG popular, we need

- a similarly easy-to-use user interface

- an access method whereby anyone can access the WWG from any client machine without installing and maintaining a complicated client software.

These criteria of success were tremendously overlooked in the first decade of Grid computing. In the current Grid systems even accessing a single Grid raises many problems:

1. The user has to learn the low-level command-line interface of the Grid middleware that is far not user-friendly.
2. The learning curve for the Grid is quite steep.
3. The definition and execution of PS jobs, workflows and PS workflows are not standard and hence in many cases require the user's own development.

   Using a multi-Grid system like the WWG will increase the number of problems:

4. Using another Grid requires the learning of low-level command-line interface of the other Grid, too.
5. Porting applications between Grids based on different Grid technology requires substantial re-engineering of the Grid application.
6. The user should be able to use several Grids simultaneously.

In order to solve Problems 1-2 and 4 the user needs a high-level workflow concept that would be implemented on top of any kind of Grid middleware and hence would hide the low level details of the actual Grid middleware. Since the workflow application would run on top of any kind of Grid middleware, porting the workflow application from one Grid to another would not require any modification of the application code and in this way Problem 5 could easily be solved. The WWG concept requires exactly this approach since in this case it does not matter which Grid is selected by the meta-broker the workflow application could run on top of the selected Grid as well. To support the development and execution of such high-level Grid applications we need workflow-oriented application hosting environments that can manage the workflows and coordinate the Grid resources on behalf of the user. Such an application hosting environment should be able to support the simultaneous usage of several Grids and orchestrating Grid resources even if they belong to different Grid systems. In this way Problem 6 could be solved by the application hosting environment concept.

In fact the application hosting environment can be implemented in two ways:

- based on the thick client concept
- based on a portal and the thin client concept

If Grid portals are available, the user does not have to install and maintain the application hosting environment on his machine. He can use the Grid from anywhere in the world through a simple web browser in the same way as the WWW can be accessed from any client machine. According to this Grid portals are more and more popular in scientific communities.

Problem 3 is a difficult issue. There is already a large number of workflow concepts (BPEL [5], Taverna [6], Triana [7], P-GRADE [8], Pegasus [9], GridAnt [10]) that are accepted by different communities and obviously all these communities would insist on their own workflow system. It would be extremely difficult to agree on a common standard. However, it is not unconditionally necessary. The same way as many different programming concepts and languages are used for programming the individual computers, many different workflow concepts could exist for the WWG. It is the task of the compilers to make sure that the code of a certain programming language be executable on many different machines. The same way it is the task of the hosting environments to make sure that a certain workflow system be executable on many different Grid systems. Another important aspect is that like web pages can be linked together in the Web, the same way different workflows should be linked together in the WWG. In this case anyone can use his favourite workflow concept but at the same time at any position of his workflow he can link existing workflows developed by someone else even if it is constructed based on another workflow concept. This interoperability of workflows would be as important as the interoperability of the Grid middlewares if we want to create a widely used WWG system.

We have seen that the WWG can be established technically by the introduction of meta-brokers and the users can access the WWG via such meta-brokers (see Fig. 6). Though it would simplify and make uniform the access of the WWG by eliminating Problems 4 and 5, still many of the problems mentioned above would remain. So as a conclusion we can say that in order to solve Problems 1-6 we need

- Portals
- Interoperable workflows and workflow concepts
- Workflow application hosting environments that provide workflow management

The portals should be able to provide the necessary user interface for the workflow concepts and for the workflow application hosting environments. So the three components integrated together will solve the user interface problems of the scientific computational WWG. Notice that a Grid portal that integrates these three aspects of the Grid user interface is much more than usually people mean by a portal. In order to distinguish such integrated portals from the generally used simple job submission portals we shall call them Advance Grid Portals (AGP). The advantages of Advance Grid Portals are as follows:

- The user does not need to install anything but can use the Grid immediately without any installation effort.
- The portal can be used from anywhere, so the user is free to move between client machines without causing it any trouble of accessing the Grid.
- The portal hides any changes in the Grid middleware. It is the portal administrator and not the user who has to update the user interface according to the new middleware releases.
- An AGP provides a much higher level user interface than the original middleware user interface. In many cases the low-level command-line interface is replaced with high-level graphical interface making the usage of the Grid much more user-friendly.
- The learning curve for the Grid is much less steep than in the case of the command line interface of the original middleware.
- AGPs can even connect the user to several different Grids that could be built on different Grid middlewares.
- AGPs provide the necessary application hosting environment and Grid orchestration mechanisms to handle PS jobs, workflows and PS workflows.

The overall solution would be to access the meta-brokers via portals, i.e., AGPs should be connected to the meta-brokers and the collaboration of AGPs and meta-brokers would result in the ultimate user access mechanism for the scientific computational WWG as shown in Fig. 7.

The next issue to be discussed is how to support the various application level parallelisms by such a user access mechanism and what are the main requirements for the AGPs to support all levels of application parallelism. In order to understand how far we are from creating such an AGP we show the current status of Grid portal technology.

3.1 Supporting application parallelism by portals in a single Grid environment

Different Grid portals support different levels of application parallelism and they are typically tailored to one particular Grid. An AGP should support all the four types of parallelism both in the UR and BR model within a single Grid.

SJ parallelism

Practically every Grid portal (not only AGPs) can support this feature in case of the UR model. However, most of the job submission portals are typically tailored for one particular Grid. Required functionalities from the portal are:

- User proxy management
- Job submission management

There are much less Grid portals that can support this feature in case of the BR model. It is because in case of the BR model one more requirement appears:

- Handling the job description language of the connected broker

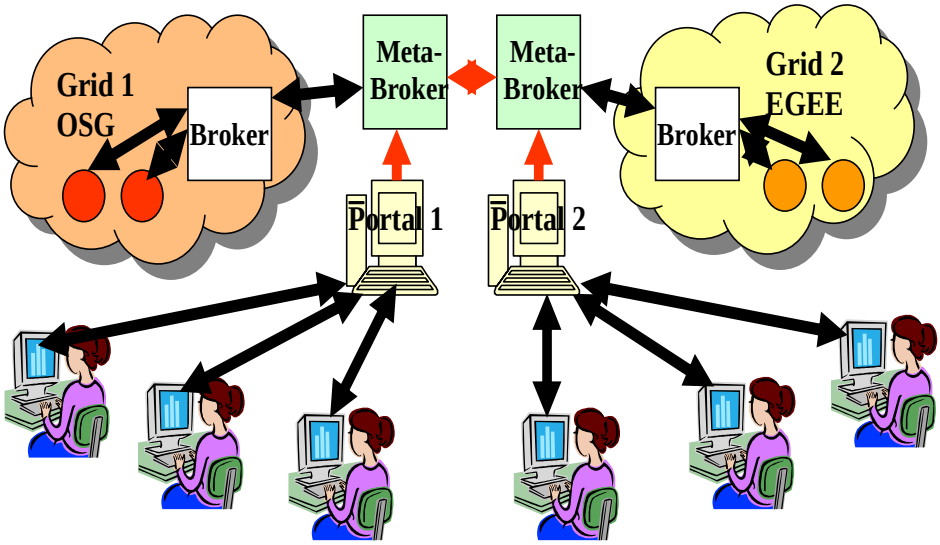

Figure 7: User interface for the WWG

Even if a portal can support this feature it is typically tailored for one particular Grid. An AGP should provide this features in a Grid neutral way. Unfortunately, there is only very few portals that fulfill this requirement.

PSJ parallelism

There are even less Grid portals that can support this feature. Still most of them are tailored for one particular Grid. The required functionalities from the portal in case of the UR model are:

- User proxy management
- Job submission management
- Repeated submission of the same job with many different parameter sets
- Support for the user to create these parameter sets in a convenient way
- To assign different resources for the different job instance executions even if the connected Grid has no broker

In case of the BR model one more requirement appears:

- Handling the job description language of the connected broker

WF parallelism

Grid portals that can support this feature are typically tailored for one particular Grid and for one particular WF concept. Required functionalities from the portal in case of the UR model:

- User proxy management
- Job submission management
- Workflow description language support (wf editor, wf interpreter, etc.) - this is sometimes a graphical support
- Workflow management support (wf enactor, wf job submission, file transfer support between wf nodes, etc.)

- Support for the user to map WF nodes to Grid resources

In case of the BR model there are some additionally required functionalities:

- Handling the job description language of the connected broker
- Workflow level scheduling support for the broker to optimize the WF node -¿ Grid resource mapping decision.

PSWF parallelism

Required functionalities from the portal in the UR model:

- All the functions that are needed for PS job execution support
- All the functions that are needed for UR workflow support
- Extension of the workflow management to handle large number of workflows in parallel
- Extension of the workflow execution visualization to give a comprehensive view about all the running workflows

Required functionalities from the portal in the BR model:

- All the functions that are needed for broker support
- All the functions that are needed for PS job execution support
- All the functions that are needed for BR workflow support
- Extension of the workflow management to handle large number of workflows in parallel
- Extension of the workflow execution visualization to give a comprehensive view about all the running workflows

As far as we know there is only very few portals that can support these features both in the UR and BR model (P-GRADE portal [11], LEAD portal [38]).

3.2 Supporting application parallelism by AGPs in the WWG environment

As we have seen in the previous section some of the portals can already support PSJ, WF and PSWF parallelism. It can be predicted that soon in the future there will be much more such portals. The interesting point here, that once a portal can support all these parallelisms within a single Grid based on the BR model after connecting it to the meta-broker it will be able to support the BGBR model nearly immediately.

The only additional feature that is needed for the exploitation of the BGBR model is the capability of handling several Grid certificates per users. It is needed according to the current VO isolation of Grid systems. In several cases, if a user would like to run a job in two VOs of two different Grids he needs two certificates that are accepted by these VOs/Grids. Once these certificates are available their proxys should be passed to the portal before the job execution is started. In this case the portal can notify the meta-broker which are those VOs and Grids to which the user has valid certificate proxys and the meta-broker can select Grids/VOs according to these conditions.

The management of several certificates per user is typically a missing feature in the current portals because they are tailored to one particular Grid. One exception is the P-GRADE portal that was designed from the very beginning as a multi-Grid portal [8]. It was prepared for working according to the UGUR and UGBR model. For example, in case of the UGBR model the user could select a Grid for any node of the workflow and then the portal used the broker of that Grid to select the right resource. Once the user selects the Grid, the portal automatically transfers the corresponding user certificate proxy together with the job.

In order to support the UGBR model the portal should be able to handle the job description languages of all the connected brokers. The nice feature of the BGBR concept is that in this case it is enough for the portal to handle only one job description language. This should be the JSDL that is a standard language proposed by OGF and adapted by more and more Grids. In this way a BGBR portal could be much simpler than an UGBR portal. It means that once the meta-broker is available P-GRADE portal can immediately become a BGBR portal that can exploit all the four levels of application parallelism.

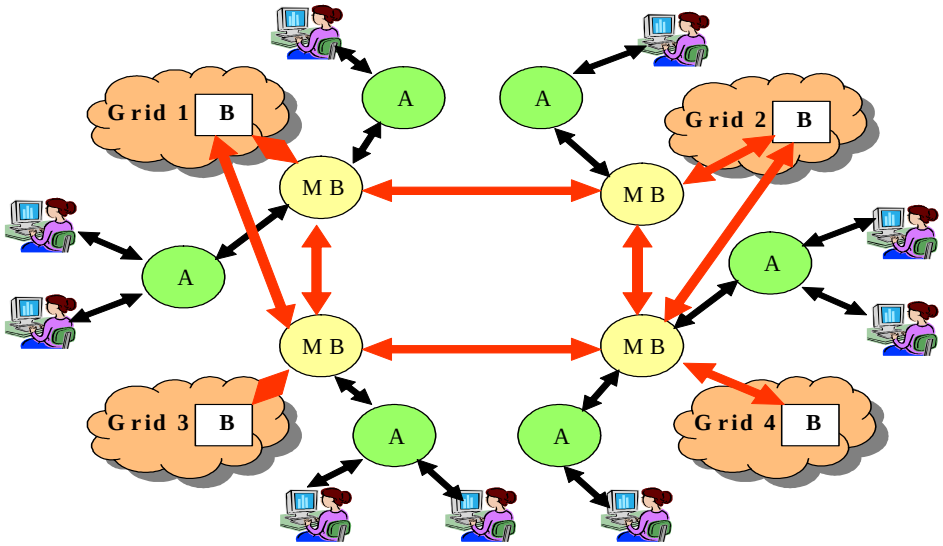

Figure 8: The architecture of the WWG based on meta-brokers and AGPs

It is obvious that if the meta-broker concept is accepted by the Grid community and meta-brokers will be available, many different portals will be connected to the meta-brokers in order to exploit all the four levels of application parallelism according to the BGBR model.

So where are we?

At this point we have a WWG infrastructure (corresponding to Fig. 8 where A=AGP, B=Broker, MB=Meta-Broker) that can execute PSWF applications in a dynamic, distributed and parallel way exploiting the connected production Grids and their resources. We have user interfaces (AGPs) that can be used to develop and execute workflows. This is now a very attractive WWG because it provides a very large number of resources that can be executed by anyone who collects the necessary Grid certificates and users can build and run their applications in a very convenient way.

The limitation of this WWG stage is that users can not use each other's results, i.e., an application workflow developed on a certain portal by user X can not be used by user Y on a different type of portal. Not mentioning that a workflow developed by user Y can not use as a workflow node another workflow developed by user X. In short, workflow interoperability is missing yet.

3.3 Network of AGPs to realize workflow interoperability

There are many different existing workflow models and the different AGPs will realize these models. It means that by the different AGPs different type of workflows can be developed even for the same type of applications. This would lead to a very redundant development of Grid applications. In order to avoid this waste of efforts creating Grid applications the interoperability of Grid workflows should be solved somehow.

To solve the workflow interoperability problem we have to extend the AGPs with the capability of

- Publishing the workflows as web services
- Uploading the workflows as web services to workflow repositories
- Using web service search engines to find the requested workflows
- Downloading the workflows from workflow repositories
- Insert the downloaded workflow as node into the higher level workflow
- Execute a workflow whose nodes were downloaded workflows

Before showing the solution we introduce several definitions. We will call a workflow application that can be run on any of the existing Grid systems as Workflow Grid Service (WFGS). A WFGS can be invoked as a usual web service and the same way as web services can invoke each other WFGSs should be able to invoke each other. In order to achieve this goal first we have to understand that a WFGS should have a type. The type of a WFGS would need at least three type fields:

- Type of the workflow showing that the WFGS was defined according to workflow concept type X (Taverna, Triana, P-GRADE, etc.)
- Type of the Grid where the WFGS can run (GT4, gLite, UNICORE, etc.)
- Type of the portal by which it was developed and can be managed (MyGrid [12], P-GRADE, etc.)

It also means that the workflow concepts, Grids and portals have types. The Grid community should agree which are these types (the most frequently used workflow concepts, Grids and portals should be considered here). This type definition should be flexible enough to be easily extended with new emerging types. The overall goal is that jobs (or services) of a WFGS should be executed by any Grid and the user should be able to initiate the execution it by any portal (his favourite portal) and get back the results to this portal.

In order to achieve the goal written above the following solution is proposed:

- Create a WFGS repository (WFREP) where correct executable workflows are stored as web services.
- Create a WFGS registry (WFREG) where the workflows stored in the WFREP are registered with their type. Web services use UDDI and a similar registry can be used here as well.
- A WFREG record should contain a web service interface description of the workflow extended with the workflow type

Once a user developed a workflow and would like to publish it for the whole user community he has to register the workflow as a WFGS providing the type of the workflow as defined above. Portals also should register themselves in a Portal Registry (PREG). A portal registration record defines the type of portal, the type of workflows it can execute and the locations of available portals belonging to this class.

Once a user through portal X would like to execute a WFGS he can browse the workflow registry looking for a suitable workflow description. Once it is found, the user can initiate this workflow with the required input sets. The portal will check the WFGS type if it can manage such a workflow. If yes, the portal starts to manage the execution of the workflow. If not, the portal checks in the portal registry if there is any portal that can manage the execution of such type of workflow. Once such a portal is found, the portal sends a request to the selected portal.

Notice that portals should be able to communicate with each other and hence a network of portals should be established. Since there will be many portals they could be efficiently connected by a P2P network. These portals can be connected to

- meta-brokers
- Grid brokers
- both

In any case these portals should be able to

- communicate with each other
- access the WFREG, WFREP and PREG services

This concept means that each Grid can set up its own specialized portals but these portals can collaborate with other portals in order to realize workflow interoperability. The communicating portals can also help in creating a balanced usage of the portals. Overall, we have to define the following protocols:

- Communication between portals

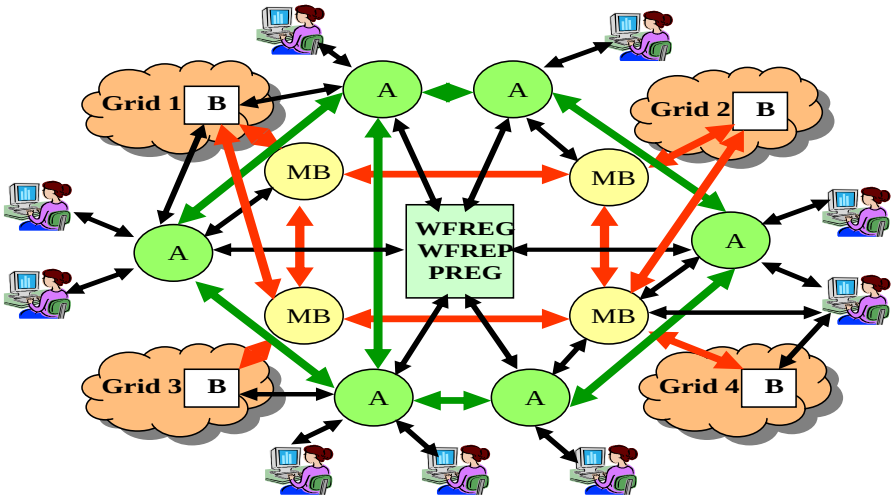

Figure 9: The final architecture of the WWG

- Communication between portals and WFREG, WFREP and PREG services

The final architecture of the scientific computational WWG could be the one shown in Figure 9 (where A=AGP, B=Broker, MB=Meta-Broker, WFREG= WFREG, WFREP= WFREP). For the sake of simplicity WFREG and WFREP services are presented as centralized services but naturally they could be distributed services, too. The main features of this WWG architecture are:

- There can be unlimited number of Grids connected in the WWG
- To connect a Grid to the WWG means to connect its broker (B) with a meta-broker (MB)
- The meta-brokers are connected by a network (e.g. P2P network)
- Portals can be connected to

  a Grid's broker (special Grid portal)

  a meta-broker

  both
- Portals are connected to each other by a network (e.g. P2P network)
- Portals are connected to the WFREG, WFREP and PREG services
- Users can access the Grid

  via portals (either special or generic portals) and can execute any Grid service (workflow) that is registered in WFREG and stored in WFREP

  directly via a Grid broker (this access is for advanced users)

  directly via a meta-broker (this access is for advanced users)

As a conclusion of chapter 3 we can say that in order to achieve a WWG where workflow interoperability and shared use of workflows can be achieved we have to define:

- The WFREG, WFREP and PREG services and their protocol with the AGPs
- The AGP intercommunication protocol

The success of WWW comes also from the use of search engines that help the users to discover the required web pages. In a similar way the WWG should be extended with search engines to help the users to discover the required services, i.e. WFGS services. Web services are described by WSDL. A web service is considered as a black box where only the external world interface should be defined. A workflow can also be considered as a black box where only the external world interface should be defined and hence WSDL can be used to describe Grid services, too. Therefore the user access mechanism of a web service and the WFGS could be the same (or something very similar). From the user's point of view a WFGS should appear in the same way as a web service.

Since a WF Grid service appears as a web service the same search engine mechanisms that are used for web services could be used for WF Grid services. There are several projects that investigate the creation of such search engines for web services:

- Woogle - Web-Service Search Engine [13]
- SAL, "The Web Services Network" provides a Web Service search engine [14]
- eSigma's web service search engine [15]

For example, Woogle instead of crawling the web for web services, obtains web services from UDDI registration nodes, and focus on extracting the semantic meaning of web services based on WSDL descriptions. It performs composition of web services to form a web service repository and presents for the users a search interface that exposes the semantic relationship of web services to the largest extend.

Since the web services community intensively pursuits research on web service search engine mechanisms, the Grid community does not have to develop a special WWG search engine rather can use the existing web service search engines. As a result one of the most challenging problems of tackling workflow interoperability, i.e., discovering WFGS services in the WFREG registries can be inherited from the WWW community. Although this significantly simplifies the problem of workflow interoperability, it still remains one of the hardest problem to establish a scientific, workflow-oriented, computational WWG.

4 Related work

The most notable work related to Grid interoperability is going on in the framework of the GIN initiative [16] of the OGF as written there: "The purpose of this group is to organize and manage a set of interoperation efforts among production Grid projects interested in interoperating in support of applications that require resources in multiple Grids." The GIN related web page of the UK NGS [17] writes: "Grid Interoperation Now (GIN) is a Community Group of the Open Grid Forum (OGF). It aims to organize and manage a set of interoperation efforts among production Grid projects to support applications that require resources in multiple Grids." Obviously the goal of the GIN is very close to the objectives of establishing a WWG although their ambitions do not go so far. The phase 1 tasks of the GIN VO is "to plan and implement interoperation in specific areas, initially data location and movement, authentication/authorization and identity management, job description and execution, and information services."

The GIN has created the GIN VO with resources from the major production Grids: TeraGrid, UK NGS, EGEE, OSG and NorduGrid. All these Grids allocated some Grid sites to do experiments on their interoperability. In the framework of the GIN VO activity a GIN Resource testing portal [18] has been set up based on the P-GRADE/GEMLCA portal technology. This portal enables the job submission (even workflow submission) to all these Grid sites and hence can be used to constantly monitor their availability and usability in the GIN VO.

GIN Phase 1 services comprises the following agreements:

- Authentication is via x509 credentials.
- Credentials used should come from Certification Authorities recognised by the IGTF
- Authorization attributes are transmitted via VOMS extensions to the x509 credentials
- VO Naming convention is described in the GIN VO naming document
- Grid FTP is the lowest common denominator for file transfer
- SRM and SRB islands for data management are being established

- GRAM and WS-GRAM islands for job submission are being established
- Grids offering Job Execution services should offer a persistent Community Software area where groups can install persistent code for their applications.
- Each Grid's internal Information system will act as a buffer/translator for accessing information about GIN participating services
- A subset of the GLUE schema is being used as the common description schema for GIN services

Although the goals of the GIN and the WWG described in this paper have many similarities this list of services in GIN show that the concept of the GIN is quite different from the implementation concept of the WWG.

A major European effort in providing Grid interoperability between gLite, UNICORE and Globus is the OMII-Europe project [19] that tries to establish interoperability at the level of five services:

1. a Basic Execution Service supporting **JSDL**
2. a Data Integration Service, specifically **OGSA DAI**
3. a Virtual Organisation Management Service, specifically **VOMS**
4. an Accounting Service, based on the forthcoming OGF **RUS** specification
5. a Portal capability, specifically **GridSphere**

Comparing the list of these services and the GIN list it can be seen that the two approaches are quite similar and both differ significantly with the WWG concept described in this paper.

The World Wide Grid testbed [20] initiated by Monash University has the same name as we used in this paper but their WWG has a quite different meaning than our WWG. Their WWG is not about to connect the existing production Grids in order to establish an interoperable WWG, rather it is a volunteer Grid test-bed specifically intended to test the Grid economy concept developed in the Grid Economy project [21]. On the other hand their Grid economy concept is probably the closest one to the Grid market model described in this paper. Another similarity between the Grid Economy project and the WWG concept described here is that they also intend to exploit maximum parallelism on their WWG including the support of parameter sweep applications. In order to achieve this goal they also defined and implemented a Grid resource broker called as Nimrod-G [22]. The latest version of Nimrod-G supports both deadline (soft real-time) and budget (computational economy) constraints in scheduling and at the same time it can optimise execution time or budget expenses. Overall, this is very similar to the Grid market concept we need for the WWG although their broker does not support the interoperable usage of multiply Grids. Their recent paper on InterGrid [37] shows many similarities with our concept of connecting existing Grids into a WWG. They introduce InterGrid Gateways to connect the different Grids while we porpose the usage of meta-brokers. They do not emphasize the importance of advance Grid portals and workflow interoperability but they put more emphasis on Grid economy.

The meta-broker concept for providing Grid interoperability is quite new and was first proposed by SZTAKI [23] at the CoreGrid workshop organized in conjunction with EuroPar'2006. Since that time another CoreGrid partner, the Barcelona Supercomputing Centre has started to work on this concept. The definition of the Broker Property Description Language (BPDL) is an on-going joint work between SZTAKI and the Barcelona Supercomputing Centre. The detailed architecture plan of the meta-broker is described in [4].

Providing Grid interoperability by P-GRADE portal at the workflow level has been published at the GELA workshop [24]. This paper shows that using the P-GRADE/GEMLCA concept workflow level Grid interoperability has been realized between Grids even if they are realized on different Grid middleware technologies: GT2 (UK NGS), GT4 (WestFocus Grid), LCG-2 and gLite (EGEE). There are other portals that aim to support Grid interoperability. As shown above the OMII-Europe project tries to tackle Grid interoperability by GridSphere [25]. However, GridSphere is a low level Grid portal framework where still much to do to provide higher level workflow services. This is exactly the concept of the P-GRADE portal. Similar high-level workflow services were investigated in the European K-WF Grid project [26]. The Japanese GridSpeed project [27] aimed at automatically generating Grid portal that are able to support workflow and parameter sweep application over several Grids. As mentioned earlier there are many workflow-oriented projects [5-10] whose workflow concepts could be easily integrated into the GridSphere portal framework.

Important work is going on concerning workflow interoperability in the Workflow Management Coalition (WfMC). They developed workflow interoperability standards for the Internet [28] and a reference model for workflow interoperability [29]. The reference model of the WfMC identifies five functional interfaces, that connect a workflow management system with external application systems, of which Interface 4 deals with workflow interoperability [30]. The AFRICA project [31] at the Univ. of Muenster is based upon the Wf-XML standard of the WfMC, but it also contains a number of significant enhancements that provide a secure, reliable management of global workflow processes. The Interworkflow Project at the Kanagawa Institute of Technology, Japan, focuses on the definition of a global workflow model for an interorganizational business process [32]. The CrossFlow project [33] is dealing with contract-based workflow interoperability between business partners. In the context of Grid the WOSE workflow portal framework was proposed to solve the interoperability of Taverna and BPEL4WS workflows [34].

5 Conclusions

The paper described three major steps to establish a scientific, workflow-oriented, computational World Wide Grid:

1. Create meta-brokers and connect existing production Grids by them in order to form the WWG infrastructure where existing production Grids become interoperable at the workflow level.
2. Create workflow-oriented advance Grid portals and connect them together and to the meta-brokers.
3. Create workflow Grid services and their registry and repository in order to make them interoperable.

The order of these steps is important and represents the priority among the steps. Step 1 is an indispensable action in order to establish the WWG infrastructure. Notice that it does not require the interoperability of different Grids at every level of the Grid middleware. This is exactly the advantage of this concept. It provides interoperability only from the user's perspective at the application level. The only concern of the user is that his application should be distributed all over the WWG resources in order to exploit as many resources as necessary. From the user point of view it does not matter how this Grid interoperability is solved, therefore the simplest solution is the best. The meta-broker concept requires only one protocol definition and standardization, namely the communication protocol between the meta-broker and the brokers of the production Grids. Once this protocol agreed the only thing the production Grids should do in order to join the WWG is to realize this protocol as an extension to their broker.

In the meantime until this protocol is not agreed the developer of the meta-broker can extend the meta-broker architecture with the necessary Translator and Invoker units that fit to the requirements of the different production Grids and enable the connection of the different production Grids to the WWG without any modification of the production Grid middleware (including its broker). The development of the meta-broker is an on-going work in SZTAKI in the framework of the EU CoreGrid project. The Translator and Invoker modules for the EGEE broker and the GTBroker are already under development.

If step 1 is completed but the other two steps are not finished the WWG becomes usable (users can submit their jobs even parameter sweep job applications) but of course it will be restricted in many ways.

Step 2 extends the usability of the WWG established by Step 1 with the usage of workflow and parameter sweep workflow applications that can be managed by Advance Grid Portals (AGP). These AGPs contain workflow managers and by collaborating with the meta-broker the nodes (jobs and service calls) of the workflow application can be distributed among the different Grids of the WWG. Notice that like in the case of Step 1 where there is no restriction on the middleware of existing and future production Grids, there is no restriction of using any kind of advance Grid portals. Any community can develop its own advance Grid portal based on its own workflow concept. The only thing to do is to connect their AGP to the meta-broker. Once it is done the AGP's user community can access the WWG and exploit all the resources of the WWG (provided that they have certificate for those resources).

Such an AGP is already exists and it is called as the P-GRADE Grid Portal. It realizes a DAG-based workflow concept, a workflow manager and can support even parameter sweep applications both at the job and workflow level. P-GRADE has been connected to several brokers and hence its connection to a new broker like the meta-broker is a routine work. As soon as the meta-broker and the WWG works, P-GRADE can immediately serve the whole community to access the WWG. Of course there are many other existing Grid portals that can be easily and quickly extended towards an AGP that can be connected to the meta-broker and the WWG. In order to do this connection the communication protocol between the meta-broker and the AGPs should be defined, standardized and implemented.

Step 3 further extends the usability of the WWG by enabling the shared and interoperable usage of workflows developed by different communities. It means that any community using its own workflow concept can easily use a complete workflow developed by another workflow community in the form of a Workflow Grid Service (WFGS) and place the call of such a WFGS as a node in its own workflow. In this way workflows developed by different users can be linked together in the same way as HTML pages can be linked together in the WWW. This extension of the WWG will significantly extend the usability and popularity of the WWG.

Unlike Steps 1 and 2 that are under development and close to be finished, work on Step 3 has not started yet. Within the framework of the EU CoreGrid project we plan to integrate the Triana and P-GRADE workflow systems as written in this paper. The hope is here that the results developed for the Internet and web services can be easily adapted for the WWG.

We are not too far from establishing a scientific, workflow-oriented, computational WWG. Steps 1 and 2 can quickly be finished. Step 3 can also be completed in reasonable time if the international Grid community accepts these ideas and is ready to work on these issues. Notice that Steps 1-3 can be realized as volunteer Grid services as written in [36] since they do not unconditionally need any modifications in the middleware of production Grids. It means that this level of the WWG can be achieved even if the different production Grids can not agree on the necessary protocols of Steps 1-3.

As it was pointed out earlier, this stage of the WWG (steps 1-3 completed) is already a usable version where we have a WWG that is easy to use, provides the highest available parallelism for workflow and parameter sweep applications by accessing Grid resources from any Grid/VO that is connected to the WWG and the user has an accepted certificate for it. However, as it was mentioned in the Introduction, there are a couple other steps to be realised. In step 4 we have to motivate people not only to use but also to provide resources to the WWG. This requires the development of an appropriate Grid market model for the WWG concept. There are several projects already working on Grid market models [21] [35]. It would be important to concentrate the efforts of these projects to establish the necessary scientific WWG market model. If we want to extend the scientific WWG towards a generic WWG usable by everyone not just by scientists, we have to replace the current concept of rigid and static VOs with a new, dynamic VO concept where VOs can be created on-the-fly according to the actual demands of the application (step 5). This would also need changes in the security concept of the current Grids by enabling the usage of the WWG to anyone without any restrictions in the same way as the WWW can be accessed by anybody. In order to achieve this freedom of usage of the Grid the trust model should be changed. Instead of trusting the user we should trust the infrastructure and create a WWG that is secure enough that no malicious user can make any harm to it. The emerging virtualization technology could be used for this purpose. Although these two questions are out of the scope of this paper, their significance cannot be neglected when creating the WWG and description of the details of such a new Grid market model, security and trust concept will be the subject of a forthcoming paper.

Acknowledgements

The research described in this paper was carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265) and particularly in the institute of Grid Systems, Tools and Environments.

References

[1] GT 2.4: The Globus Resource Specification Language RSL v1.0:
http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
[2] JDL Job description language.
http://www.Grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf
[3] A. Anjomshoaa, et al, Job Submission Description Language (JSDL) Specification, Version 1.0:
http://www.Gridforum.org/documents/GFD.56.pdf
[4] A. Kertesz and P. Kacsuk, Meta-Broker for Future Generation Grids: A new approach for a
high-level interoperable resource management, submitted to EuroPar2007
[5] BPEL. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
[6] Taverna. http://taverna.sourceforge.net/

[7] I. Taylor, et al, Visual Grid Workflow in Triana, Journal of Grid Computing,
Vol. 3, Nos. 3-4, 2005, pp. 153-169
[8] P. Kacsuk and G. Sipos, Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal,
Journal of Grid Computing, Vol. 3, Nos. 3-4, 2005, pp. 221-238
[9] E. Deelman et al. Pegasus, Mapping scientific workflows onto the Grid, Proc. of Across Grids Conf.,
Nicosia, 2004
[10] G. von Laszewski, et al, GridAnt: A Client-Controllable Grid Workflow System,
http://www-unix.mcs.anl.gov/ laszewsk/papers/vonLaszewski–Gridant-hics.pdf
[11] P. Kacsuk et al. Workflow-level Parameter Study Support for Production Grids, Proc. of ICCSA07,
LNCS 4707, 2007, pp. 872-885
[12] S. R. Egglestone et al, A portal interface to myGrid workflow technology,
http://www.mrl.nott.ac.uk/ sre/papers/NETTAB2005/nettabextabst.doc
[13] Woogle web service search engine. http://data.cs.washington.edu/webService/
[14] SAL. http://www.salcentral.com/salnet/srchsal.htm
[15] eSigma's web service search engine.
http://www.esigma.com/telos/discover/browseservices.html?cid=0,telos:categories:general:utilities
[16] GIN Project. https://forge.Gridforum.org/projects/gin
[17] UK NGS web page on GIN. http://wiki.ngs.ac.uk/index.php?title=GIN
[18] GIN Resource testing portal.https://gin-portal.cpc.wmin.ac.uk:8080/Gridsphere/Gridsphere
[19] OMII-Europe Project. http://www.omii-europe.com/
[20] World Wide Grid testbed. http://www.Gridbus.org/ecoGrid/wwg/
[21] Grid Economy project. http://www.Gridbus.org/ecoGrid/
[22] Nimrod-G project. http://www.csse.monash.edu.au/ davida/nimrod.html/
[23] A. Kertesz and P. Kacsuk, Grid Meta-Broker Architecture: Towards an Interoperable Grid Resource
Brokering Service, CoreGrid workshop on Grid Middleware in conjunction with EuroPar2006, Dresden, 2006
[24] P. Kacsuk, G. Sipos and T. Kiss, Solving the Grid Interoperability Problem by P-GRADE Portal
at Workflow Level,Grid-Enabling Legacy Applications and Supporting End Users Workshop (GELA), Paris, 2006
[25] GridSphere. http://www.Gridsphere.org/Gridsphere/Gridsphere
[26] K-WF Grid project. http://www.kwfGrid.net/main.asp
[27] T. Suzumura et al, GridSpeed: A Web-based Grid Portal Generation Server, HPCAsia2004, 2004
[28] J. G. Hayes, et al, Workflow Interoperability Standards for the Internet, IEEE Internet Computing,
June, 2000, pp 37-45
[29] Workflow Management Coalition Reference Model Interface 4 for workflow interoperability.
http://www.wfmc.org/standards/docs/article_key_to_ecommerce.PDF
[30] Workflow Management Coalition: Reference Model. Document Number WFMC-TC-1003, Brussels, 1995.
[31] Michael zur Muehlen, Florian Klein, AFRICA: Workflow Interoperability based on XML-messages,
Proc. of CAiSE2000, http://www.workflow-research.de/Publications/PDF/MIZU.FLKL-AFRICA(CAiSE2000).PDF
[32] Hayami, Haruo: Development and Experimental Proof of an Interworkflow Management System.
Presentation at the Workflow Management Coalition Meeting in Tokyo, December 1999. http://www.wfmc.org
[33] The CrossFlow Project. http://www.crossflow.org
[34] L. Huang et al, A Workow Portal Supporting Multi-Language Interoperation and Optimisation,
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE, 2002,
http://epubs.cclrc.ac.uk/bitstream/917/cpe_wose.pdf
[35] Jiadao Li, Ramin Yahyapour, Negotiation Strategies for Grid scheduling, 2006
(available http://www-ds.e-technik.unidortmund.de/ yahya/papers/cei_GPC2006.pdf)
[36] P. Kacsuk, Extending the Services and Sites of Production Grids by the Support of Advanced
Portals, Proc. of VECPAR2006, Rio de Janeiro, 2006
[37] M. D. Assuncao, R. Buyya and S. Venugopal, InterGrid: A Case for Internetworking Islands of Grids,
in print of J. of Software Practice and Experience
[38] D, Gannon et al. Dynamic, Adaptive Workflows for Mesoscale Meteorology, in Workflows for e-Sceince
(eds: I. J. Taylor et al), Springer 2007, pp 126-142

Glossary

AGP	Advance Grid Portal
BR	Broker selected Resource
BGUR	Broker selected Grid User selected Resource
BGBR	Broker selected Grid Broker selected Resource
IaG	Intra-Grid
IaR	Intra-Resource
IrG	Inter-Grid
IrR	Inter-Resource
PREG	Portal Registry
PS	Parameter Sweep
PSJ	Parameter Sweep at Job level
PSWF	Parameter Sweep at Workflow level
SJ	Single Job
UR	User selected Resource
UGBR	User selected Grid Broker selected Resource
UGUR	User selected Grid User selected Resource
UTGBR	User-Transparent Grid Broker selected Resource
WF	Workflow
WFGS	Workflow Grid Service
WFREG	WFGS Registry
WFREP	WFGS Repository
WWG	World Wide Grid
WWW	World Wide Web