

Workflow-level Parameter Study Management in multi-Grid environments by the P- GRADE Grid portal¹

Peter Kacsuk, Zoltan Farkas, Gergely Sipos, Adrian Toth, Gabor Hermann

MTA SZTAKI, 1111 Kende utca 13

Budapest, Hungary

e-mail: {kacsuk, farkas, sipos, toth.adrian, gabor.hermann}@sztaki.hu

Abstract

Workflow applications are frequently used in many production Grids. There is a natural need to run the same workflow with many different parameter sets. Unfortunately current Grid portals either do not support this kind of applications or give only specialized support and hence users are obliged to do all the tedious work needed to manage such parameter study applications. P-GRADE portal has been providing a high-level, graphical workflow development and execution environment for various Grids (EGEE, UK NGS, GIN VO, OSG, TeraGrid, etc.) built on second and third generation Grid technologies (GT2, LCG-2, GT4, gLite). Feedback from the user communities of the portal showed that parameter study support is highly needed and hence the next release of the portal will support the workflow-level parameter study applications. The current paper describes the semantics and implementation principles of managing and executing workflows as parameter studies. Two algorithms are described in detail. The black box algorithm optimizes the usage of storage resources while the PS-labeling algorithm minimizes the load of Grid processing resources. Special emphasis is on the concurrent management of large number of files and jobs in the portal and in the Grids as well as providing a user-friendly, easy-to-use graphical environment to define the workflows and monitor their parametric study execution.

1. Introduction

One of the most promising utilizations of Grid resources comes to life with parameter study (or sometimes written as “parametric study” or “parameter sweep”) applications where the same application should be executed with a large set of input parameters. Such parameter study applications are easy to implement in the Grid since the different executions started with different parameters are completely independent. Indeed, there are several projects [1], [2], [3] that demonstrated that parameter study applications are easily manageable in the Grid. However, most of these projects tackled only single job based applications. The real challenge comes when complex applications consisting of large number of jobs/services connected into a workflow should be executed with many different parameter sets. There have been only two projects that tried to combine parameter studies with workflow-level support in the Grid. ILab [4], [5], [6] enables the user to create a special parameter study oriented workflow. With the help of a sophisticated GUI, the user can explicitly define how to distribute and replicate the parameter files in the Grid and how many independent jobs are to be launched for each segment of the data files. This approach is very static restricting the exploitation of the dynamic nature of Grids that enables the dynamic collection of resources. The SEGL [7] approach puts much more emphasis on exploring the dynamic nature of the Grid. They also provide a GUI to define the workflows and to hide the low level details of the underlying Grid. The SEGL workflow provides tools for several levels of parameterization, repeated processing, data archiving, handling conclusions and branches during the processing as well as synchronization of parallel branches and processes. The problem with this GUI is that it might be too sophisticated, requiring very large skill from the application developer. Furthermore both ILab and SEGL are connected to a particular Grid although in case of a parameter study execution there is a large need to exploit as many resources as possible even if they should be collected from different Grids. Finally, neither of them can be used as a service through a Grid portal.

¹ This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265) and under the SEE-GRID-2 project funded by the European Commission (Contract number: 031775).

Although our approach to support workflow-level parameter study applications in the Grid has many similarities with these two projects, there are significant differences, too. Our main goals are as follows:

1. Keep both the workflow GUI and the parameter study support concept as simple as possible. This enables the fast and easy learning of the tool as well as its easy usage.
2. Enable run any existing workflow with different parameter sets without modifying the structure of the workflow.
3. Manage the execution of the workflows on as many Grid resources as possible. Enable the collection of Grid resources from several Grids even if they are based on different Grid technology.
4. Enable the access of the workflow-oriented GUI and the available Grids via a single Grid entry point, i.e., via a Grid portal without installing any software on the user's machine.
5. Provide a dynamic balance between the usage of processing resources, storage resources and network resources.

Clearly, these goals differ from the main concept of the above mentioned two projects. The last goal can be found in several other parameter study projects that aim at the support of parameter study applications at the individual job level. In case of Nimrod/G [3] and Apples/APST [8] the main emphasis is on scheduling and hence in these projects the fifth goal of our project is dominant. In Nimrod even an economic model is considered during resource scheduling.

The starting point for our project was the P-GRADE Grid portal [9] that provides a workflow-oriented GUI as well as workflow-level interoperability between various Grids even if they are built on different Grid technologies. This means that the same portal can be connected to several different Grids and the portal manages the workflow execution among these Grids according to the users' requirement [10]. The portal even enables the parallel exploitation of the connected Grids, i.e., different jobs of the same workflow can simultaneously be executed on Grid resources taken from different Grids. Such a multi-Grid workflow execution mechanism is a unique feature of the P-GRADE portal that is now widely used for many different Grids (SEE-GRID, VOCE, EGrid, HunGrid, CroGrid, GILDA, etc.). Besides LCG-2 and gLite based production Grids the portal is successfully used as service for the GT2 based UK National Grid Service (NGS) and it was also successfully connected to the GT4 based Westfocus Grid (UK). Recently the GIN (Grid Interoperation/Interoperability Now) VO of OGF is supported by the portal enabling the simultaneous access to all of its resources coming from different Grids. This portal is also connected to US OSG and TeraGrid, UK NGS, UK WestFocus Grid, EGEE Grids and hence via this portal all the major Grids of the US and Europe can be accessed even from the same workflow.

Experiences of the portal revealed that many applications require not only the single execution of a workflow rather they seek for parameter study support to execute an existing workflow application with many different parameter sets. Therefore, our motivation was to extend the existing single workflow support of the portal towards a generic workflow-level parameter study execution support. Such support should enable the automatic starting, execution, monitoring and visualization of all the workflows belonging to the same parameter study. Of course the same way as in the case of the single workflow management environment the users should neither know any details of the underlying Grids nor insist on any particular programming language. Even legacy codes can be used as services in the workflows if the portal is integrated with the GEM/LCA legacy code architecture service [11].

In order to reach the five main goals mentioned above we have developed two algorithms to manage workflow-level parameter studies across multiple Grids. The first algorithm is based on the "black box" concept that minimizes the required storage resources of the portal but results in superfluous job executions in many cases. The second algorithm, called as the PS-labeling algorithm minimizes the usage of processing resources of the Grids but increases the storage needs of the portal. They represent two extreme points in the scale of possible execution management algorithms. The algorithms should be mixed according to a dynamic and adaptive scheduling algorithm that is the subject of our future research.

The paper first introduces the workflow concept of P-GRADE portal in Section 2. The next section explains the "black box" execution semantics and its portal support both at the user interface and the portal

workflow manager level. Section 4 introduces the PS-labeling algorithm and compares it with the “black box” execution mechanism. Section 5 summarizes the various problems that should be tackled in a parameter study environment (resilience, multi-Grid execution, large number of processes and workflows, dynamic changing of input parameter set, etc.). Finally, Section 6 compares our research with related ones.

2. The workflow concept of P-GRADE portal

When designing and implementing P-GRADE portal, we focused on the following basic principles.

Simplicity and expressiveness of the user interface

P-GRADE portal provides a graphical editor to develop workflow applications. The workflow graph is a simple DAG (Directed Acyclic Graph) where nodes of the graph are either jobs (sequential, MPI or PVM) or legacy code services (if GEMLCA is integrated with the portal). For each node input and output ports can be defined. Input ports represent the input files to be used by the node and output ports represent output files to be generated by the node. Input and output ports can be connected by directed arcs (from the input to the output port). The arcs represent the necessary file transfer between the two connected nodes. Indeed, the portal run-time system automatically takes care of these predefined file transfers, so the user is completely released of transferring files among jobs during the workflow execution. A node can be executed if all the input arcs have received the necessary input files. This very simple dataflow semantics enable the parallel execution of those nodes that are simultaneously supplied with the required input files. Notice that based on this semantics the user can easily achieve two-level parallelism. The first level is among nodes of parallel branches of the workflow (inter-node parallelism), the second level is within a node if the job executable defined for the node is a parallel code (MPI or PVM). This second level is called intra-node parallelism. To keep the concept as simple as possible neither if-then-else nor loop constructs are provided in the graph. (However, based on the existing graph facilities, the user can easily create if-then-else graph templates [12]).

In order to illustrate the graphical workflow concept of P-GRADE portal Figure 1 shows a simple example workflow that is used for solving the $Ax = B$ equation where A is a matrix, B and X are vectors. The application consists of 5 jobs (all of them having sequential executable code). The first job called as “Separator” accepts the A and B matrices as input parameters on its input port, separates them and then copies A to jobs “Invert_A” and “A_mul_X”, and copies B to “Multip_B” and “Subtr-B”. The job “Invert_A” creates the invert matrix of A that is multiplied by B in job “Multip_B”. The output of “Multip_B” is the searched X vector. The next two jobs are used to check the quality of the result.

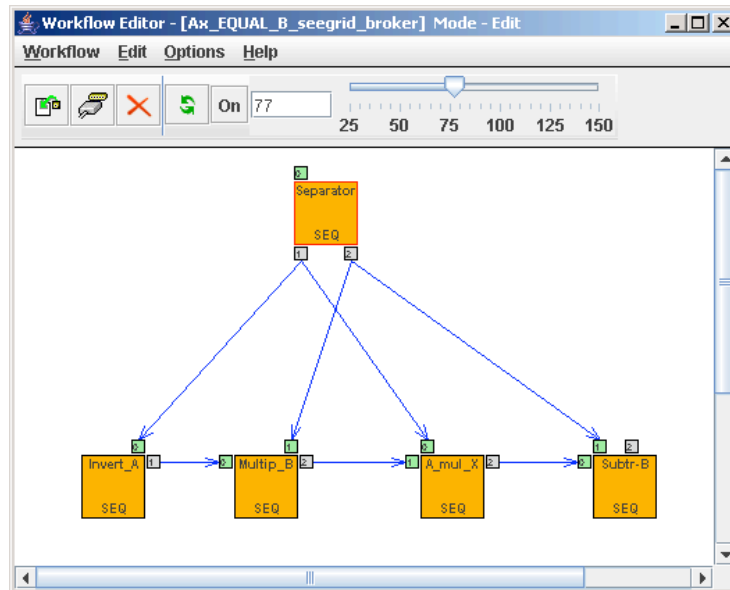


Figure 1 Workflow for Computing the $Ax=B$ equation

Exploiting the available services of existing Grids

Instead of developing a new kind of middleware as it has been done in several other parameter study related Grid projects [3], [8] our aim was to exploit the available services of existing production Grids. For this reason we have not developed our own enactment service rather we use Condor DAGMan [13]. If there are alternative services, we use them in order to ensure Grid interoperability through the portal. For example, the portal can access, process and visualize the information in both MDS and BDII type information systems. However, the portal should work even if a specific service is not available. For example, the portal can exploit the EGEE broker services to assign jobs to Grid resources but it also can manage GT2 Grids where such broker does not exist (e.g. the UK NGS). In the former case the portal automatically generates a default JDL file based on the user created workflow definition and enables the user to tune this JDL file with additional visual editing. In the latter case, the portal enables the user explicitly assigning jobs to Grid resources. In both cases, first the user should select the Grid where the job should be executed. For different jobs of the workflow the user can select different Grids either with broker or without broker.

Tailoring the Grids to user needs

The portal administrator can connect the portal to several Grids. The portal maintains a list of connected Grids common for every users. This list cannot be modified by the users only by the portal administrator. For each connected Grid a list of available resources is provided for every single user. The content of these Grid resource lists can be modified by the user. Grids usually have many unreliable sites and the user can remove from his Grid resource list those sites that he identifies as unreliable ones. On the way round, if he knows some extra sites not shown by the Grid resource list he can easily add those sites to his own Grid resource list. If later the user would like to assign a job to a Grid resource, the portal offers the available Grid resources according to these Grid resource lists.

Flexibility, expandability, standardization

In order to provide a flexible, easy to expand and standard portal we selected the Gridsphere portal framework technology [14] and used it to build our portal. Gridsphere is used by a large community enabling the exchange of available portlets that are written according to the Gridsphere standard. It also enables the easy tailoring of the portal to specific user needs. Our aim with the P-GRADE portal was to create a workflow-level core portal that can easily be extended and adapted for different user communities' needs.

3. The “black box” execution semantics of workflow-level parameter study execution

The simplest approach of supporting parameter studies at the workflow level is based on the “black box” execution semantics. It means that we consider a workflow as a black box that should be executed with many different parameter sets. These parameter sets are placed on the so-called PS input ports of the workflow. An input port is called PS input port if a set of parameter files can be received on that port. If a workflow has one such PS input port, it should be executed as many times as many elements are in the parameter file set of that port. If there are several PS input ports, the workflow should be executed according to the cross-product of these input sets. From now on we say that a workflow (WF) which has got at least one PS input port is called a PS workflow (PS-WF).

The concept of the workflow-level PS support based on the “black box” execution semantics is illustrated in Figure 2. The original WF consisting of 4 jobs is considered as a black box that have two input ports capable of accepting inputs from the outside world. On both inputs the user can provide several input sets and the workflow should be executed with the cross-product of these input sets. Figure 2 shows a case when Input port 0 accepts an input set with two elements (value 1 and 2) and Input port 1 accepts an input set with three elements (value 3, 4 and 5). Therefore, the workflow should be executed six times as shown in Figure 2.

In order to manage the execution of workflows according to the “black box” execution semantics the workflow manager of the portal was extended in the following way. Let

$$M = N_1 \times N_2 \times \dots \times N_m$$

where m is the number of PS input ports and N_i denotes the number of input files on the i -th PS input port. At run-time the portal PS-WF manager generates M executable workflows (e-WFs) from the original PS-WF. Every e-WF is labeled by m labels:

$$1 - n_1, 2 - n_2, \dots, m - n_m$$

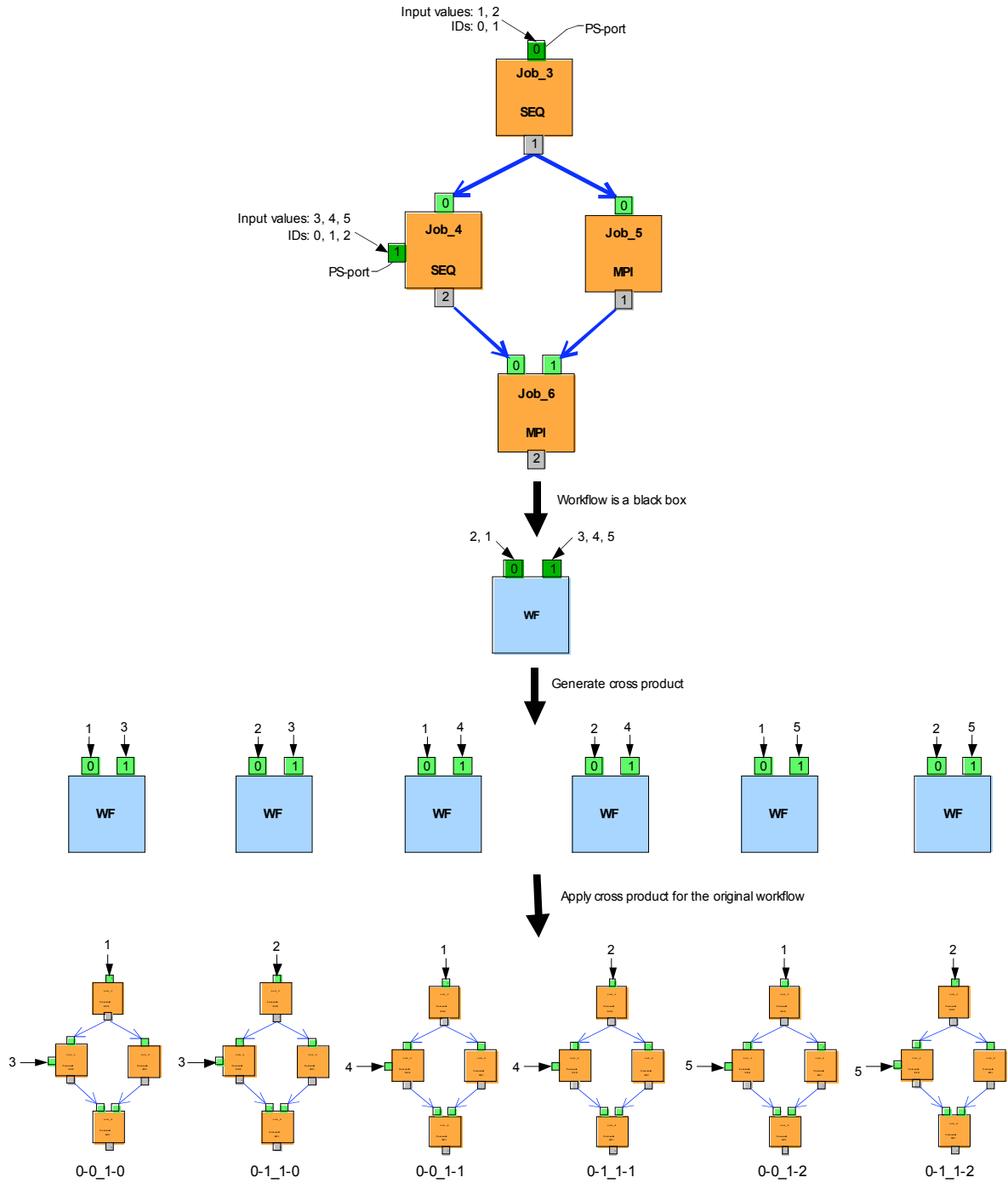


Figure 2. Concept of "black box" execution semantics

Where the internal structure of label¹ is:

$$i - n_i$$

where i identifies the i -th input PS port and n_i represents the ordering number of the input file taken from the i -th PS port in the identified execution instance:

$$0 \leq n_i < N_i$$

Figure 2 also shows these labels for each of the 6 execution instances. This labeling scheme identifies for the PS-WF manager which input file to take from the different PS input ports in the case of different execution instances (e-WFs). It also helps in identifying the output files generated at the output ports. Every output file is labeled with the label of the e-WF that generated it. Notice that output files can be local and remote. Remote output files are always permanent and once they are produced by an e-WF they can be immediately read by the user. This enables the user to study the partial results even if an e-WF is not completed. Local output files can be permanent or volatile. Permanent means that the user would like to get access to this output file only when the whole e-WF execution has been completed. These partial results are collected and stored by the portal meanwhile the e-WF runs. When the e-WF is completed these files are zipped (together with the standard output and error logs) and placed by the portal to a Grid storage resource that was defined by the user. These local permanent files should be typically small files and collecting and storing them by the portal the number of access to Grid storage resources can be significantly reduced resulting in the reduction of the overall execution time. Finally, local volatile files represent temporary partial results. As they are consumed by the connected job(s) they can be removed from the portal. This is important from the point of view of reducing the load of the portal storage resources.

Developing and running PS-WF applications according to the “black box” execution semantics require three main steps. The first step is the development of the WF application that the user would like to run as a PS-WF application. The development process of a WF application has been described in previous publications [9] as well as in the User’s Manual [15] of the current service portals and hence we do not describe it in this paper. The basics concepts are summarized in Section 2. The second step is the transformation of the WF into a PS-WF. Finally, the third step includes the submission of the PS-WF application to the Grid and monitoring the execution of the PS-WF application. Consequently, the PS-WF user interface has two major parts:

1. Definition of PS-WF graphs
2. Monitoring the PS-WF graph execution

3.1 PS-WF graph definition

In order to turn a WF application into a PS-WF application the graphical Workflow Editor (WE) of the portal was slightly extended. The user can open the existing WF by the WE and can turn any of the existing input port into a PS input port. In order to illustrate the process we use the same Ax_EQUAL_B workflow application that was shown in Section 2. The task is to modify this WF in order to solve the equation for a set of A and B parameters. Figure 3 shows how to turn the input port of the Separator job into a PS port. Notice the difference between the input port and PS input port definition. In case of a normal input port a file is associated with the port. This file can be either local (originating from the user’s machine and part of the input sandbox) or remote (placed in a storage resource of the Grid). In case of a PS port (Figure 3) a directory is associated with the port. This directory always should be placed in a storage resource of the Grid. The user should place the series of input files into this directory that must not contain any other file. Currently, the portal does not provide any portlet to support the placing of the input files in to the selected Grid storage resource. The user should use the command line interface of the actual Grid. E.g., in EGEE Grids, there are file catalog related commands by which the user can place the files in such a directory.

After defining the PS input ports the user should identify the Grid and its storage resource where the local permanent files should be stored at the end of each e-WF execution. As a summary we can say that turning an existing WF into a PS-WF is an extremely easy task. Simply turn some of the input ports to PS input ports and define the target Grid storage resource for the local permanent files. This was exactly our aim: to simplify for the user the process of utilizing existing workflows and run them as parameter studies.

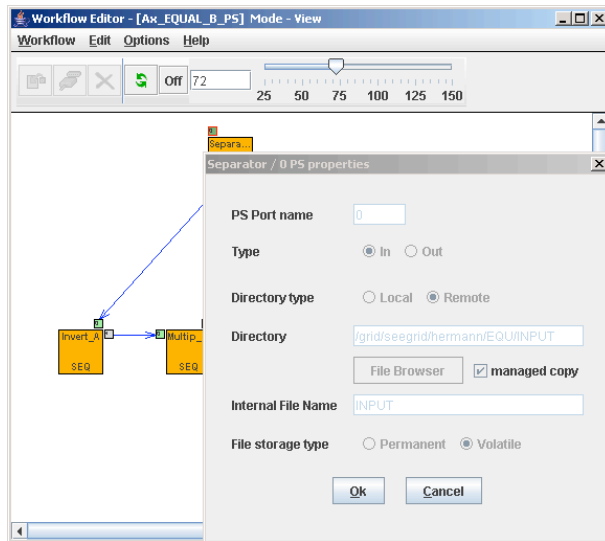


Figure 3 Definition of a PS input port

3.2 Monitoring the PS-WF graph execution

Even monitoring a single job is important for the user not mentioning when he runs thousands of jobs as part of a PS-WF execution. The challenge here is how to visualize the execution status of thousands of e-WFs and jobs in an easily understandable and manageable way. Again the monitoring of a single WF as it was managed by the portal was a good starting point. It was only slightly extended and yet it can help to monitor all the e-WFs and all their jobs in an efficient way. The ordinary WFs of a user are listed by the portal under the Workflow Manager window. Here the user can submit the WF, attach the WF to the Workflow Editor to see the graphical view of the WF, and delete the WF. Moreover, the “Details” button enables the user to see the details of the WF, i.e., to see the component jobs and their assignment to Grid resources. The PS-WFs are listed in the Workflow Manager window in the same way as the WFs. The only difference is that the PS-WFs have a “PS Details” button to show their internal details. Figure 4 shows a snapshot of the Workflow Manager window where the original “Ax_EQUAL_B_seegrid_broker” WF and its PS-WF version (“Ax_EQUAL_B_PS”) are listed before submitting the PS-WF.

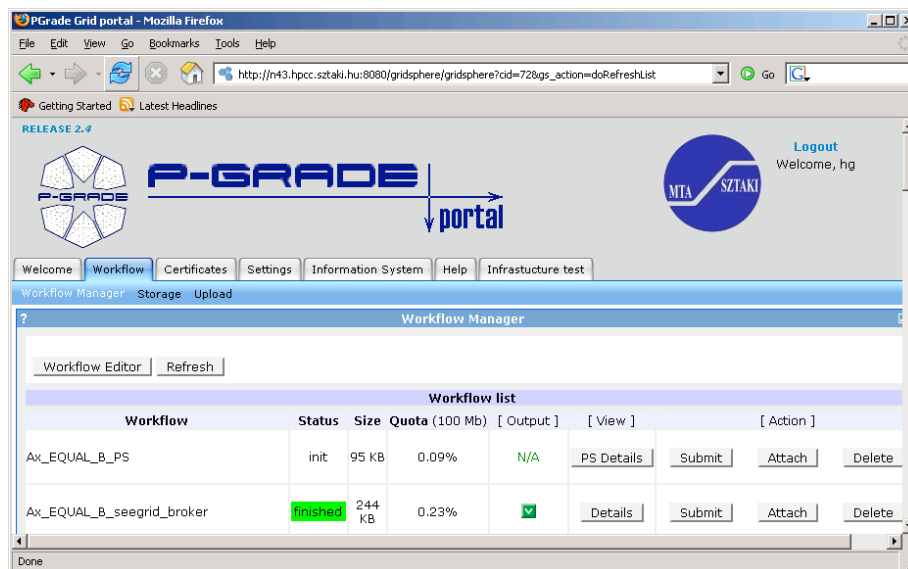


Figure 4 Workflow Manager before submit the PS-WF

Once the user submits the PS-WF (simply clicking on the corresponding “Submit” button), the portal workflow manager (WM) creates all the e-WFs that are defined by the cross-product of the PS input ports’ file sets. Then WM submits simultaneously as many e-WFs as many are permitted by the portal administrator. In principle all the e-WFs could be submitted into the Grid. However, every e-WF submission requires a significant resource set both from the portal server and from the underlying Grid. In order to prevent the flooding of the portal and the Grid by the e-WFs and jobs of a single user, the portal administrator can restrict the number of e-WFs that can simultaneously be submitted to the Grid. After the PS-WF submission using the PS Details button the user can see the statistics of the e-WFs: how many were initiated, submitted, finished and how many went on Rescue or Error. Figure 5 shows the situation where the “Ax_EQUAL_B_PS” PS-WF was submitted with 6 input parameter sets. As a result 6 e-WFs were generated by the portal. 2 of them already finished, 3 submitted and one still in init state, i.e. waiting for submission. The figure also shows that any submitted e-WF can be viewed in detail by using the “Details” button. Clicking there the detailed view of the e-WF shows the component jobs of the e-WF, their Grid resource assignment and their status. Notice that any e-WF can be aborted. It kills the selected e-WF but the other e-WFs can continue their activity. Figure 5 also reveals that those e-WFs that are finished cannot be viewed in the PS Workflow Details window. Their results (including every stdout and stderr files) are already stored in the defined Grid storage resource so the user can check those files there.

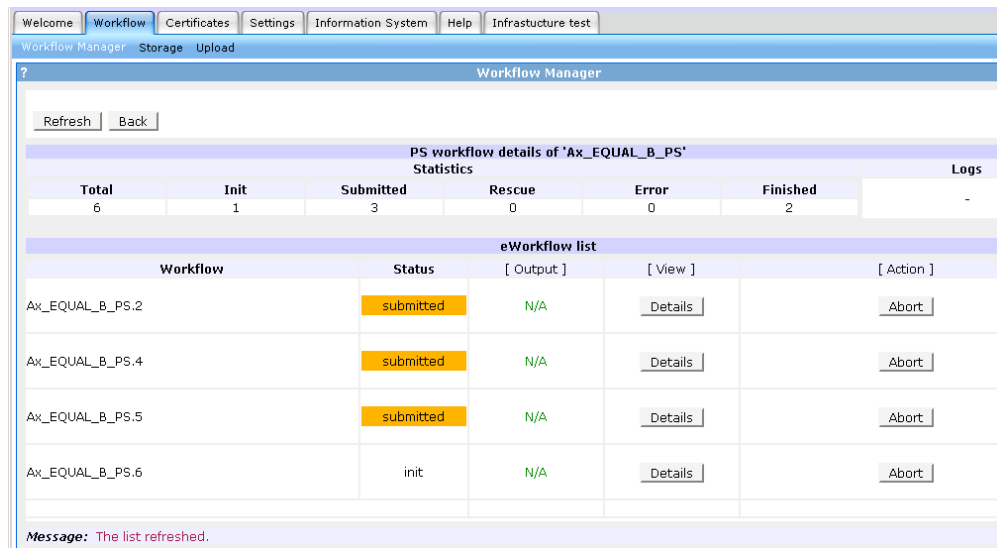


Figure 5 Detailed monitoring view of a PS-WF’s execution

4. PS-labeling algorithm

Although the “black box” execution semantics is easy to understand and apply unfortunately, it is not optimal execution semantics. In the case of the example shown in Figure 2 the whole PS-WF as a black box should be executed $2 \times 3 = 6$ times, i.e., all the jobs of the PS-WF will be executed 6 times. However, analyzing the PS-WF it quickly turns out that it is enough to execute job3 and job5 with the two input values of PS input port 0. Job4 and Job6 indeed should be executed with the cross-product of the two PS input ports, i.e. 6 times. This example shows that the “black box” execution semantics results in a redundant execution that is not tolerable if the number of files on the PS input ports are in the range of hundreds and thousands as it frequently happens during large-scale scientific simulations.

In order to be able to solve the problem illustrated by Figure 2 we have to modify the “black box” execution mechanism. The new method is based on the “black box” approach in the sense that the user should not have to modify anything related to an existing WF, except for simply turning the required input ports into PS input ports. Once this is done the portal will handle the WF as a PS-WF and provides a unique natural number as PS port identifier for the PS input ports. Of course, the user should place the necessary number of input files for every PS input port. Based on the PS port identifier and the number of input files placed on the PS ports, the portal can identify for each node of the PS-WF graph the optimal execution

number. This is done by the PS-labeling algorithm run by the workflow manager of the portal. The PS-labeling algorithm has two phases:

- Preparation phase
- Execution phase

Preparation phase

The preparation phase is executed by the workflow manager before submitting a PS-WF. Starting from each node having a PS input port represented by the label $i-N_i$ (where i denotes a unique natural number index of the PS port and N_i denotes the number of input files belonging to this port) the algorithm extends the current label of all dependent nodes by $i-N_i$. (A node j is dependent from node i , if there is a directed path of arcs starting from node i and ending at node j .)

At the end of the algorithm every node is either without any label or labeled with a series of labels:

$$\text{label}^1, \text{label}^2, \dots, \text{label}^n$$

The former case means that the node does not depend on any PS input ports and hence it should be executed only once. The latter case means that the node was on at least n paths that are rooted from n different input PS ports' node. Overall such a node should be executed

$$N_1 \times N_2 \times \dots \times N_n$$

times as the cross-product of the initial input file sets. This label set is extended to be a full label set for every node. A full label set is an ordered set of the N_i values, i.e., on position i there is the value of N_i . If there was no label for position j (PS port j) in the generated label set, then an empty position is placed here. The full label set for a given node shows those PS input ports that have impact on this node and for such PS ports it gives the number of input files demonstrating the strength of this impact. Example: let us suppose that $M=4$, $N_0=4$, $N_1=3$, $N_2=5$ and $N_3=2$ and the generated label set for node A dependent on PS port 1 and PS port 3 is: 1-3, 3-2. Then the full label set for A will be: $\{_, 3, _, 2\}$.

After completing the labeling, for each node a power set of the full label set is generated. For node A in the example above the power set will look like:

$$\{_, 1, _, 1; _, 1, _, 2; _, 2, _, 1; _, 2, _, 2; _, 3, _, 1; _, 3, _, 2\}$$

Execution phase

After completing the preparation phase, the workflow manager begins to generate and submit the e-WFs labeled as

$$n_1, n_2, \dots, n_m$$

where i identifies the i -th input PS port and n_i represent the ordering number of the input file taken from the i -th PS port in the identified execution instance:

$$0 \leq n_i < N_i$$

During the e-WF execution the workflow manager tries to substitute the execution of the node by the file(s) resulting from a previous run of the node. First the e-WF label is matched with the power set of the nodes of the PS-WF. If there is no power set of a node it means that it has to be executed only once and marked as executed for later e-WFs. If a node has a power set and the matching element is marked for the node, then this node was already executed and therefore the workflow manager does not submit the job (node) again, rather immediately provides those output files that were generated by this node during the marked run. If the matching element is not marked for a node, then this node should be executed. After executing the node

its matching element is marked and the result output files are stored either in a Grid storage resource if they were defined as remote files or inside the portal if they were defined as local files.

To illustrate the algorithm let us take the example shown at the preparation phase ($M=4$, $N_0=4$, $N_1=3$, $N_2=5$ and $N_3=2$) and consider the e-WF with label “3,2,4,1”. If node A has the power set $\{ _1, _1; _1, _2; _2, _1; _2, _2; _3, _1; _3, _2 \}$ and the matching element $_2, _1$ is not marked yet, the job or service belonging to node A should be executed. Once it is executed the matching element $_2, _1$ should be marked. If later another e-WF for example with label “1,2,3,1” is to be executed, then the workflow manager will recognize that the matching element $_2, _1$ is already marked and hence node A must not be executed again.

Notice that the price for this optimized usage of processing resources is to increase the storage capacity requirements of the portal server. There are three types of files handled by the portal in the PS-WF:

1. Local volatile files: During normal WF execution these files are immediately removed from the portal when they are consumed by their connected input ports. The same is true in case of the black box PS-WF algorithm. However, in the PS-labeling PS-WF execution algorithm they should be stored in the portal in order to substitute by them the re-execution of their producer node. They should be stored as long as there is an e-WF that have not completed yet and must use these files.
2. Local permanent files: During normal WF execution these files are stored in the portal and delivered to the user after completing the WF execution. In case of the black box PS-WF algorithm they must be stored until the corresponding e-WF is not completed yet. After that they can be removed without waiting the completion of other e-WFs. In case of the PS-labeling PS-WF algorithm they must be stored as long as there is an e-WF that have not been completed yet and must use these files. So their average storing time is much longer than in case of the black box PS-WF algorithm.
3. Remote output files: They should be stored in Grid storage resources and hence these files do not increase the storage requirements of the portal either in the case of the black box or the PS-labeling algorithm.

As a conclusion we can say that the more internal output files are defined as remote files the less storage price we have to pay for the optimized usage of processing resources. Since remote files also help in on-line checking and monitoring the execution of PS-WFs, it is recommended to use remote files instead of local files in case of PS-WFs. On the other hand remote files restrict the distribution of the execution of PS-WF nodes in different Grids. Usually, a node should be executed in the same Grid where the associated input/output remote file is stored. However, since local files are stored on the portal their processing PS-WF nodes can be assigned to any Grid connected to the portal (provided that the user has certificate and VO membership for the given Grid and VO). Therefore if a user would like to exploit many Grids for the execution of a PS-WF it is recommended to use local files instead of remote files. In this case the choice of Grids and Grid resources can be done on-the-fly by the portal provided that it is supported by a meta-broker. To develop the optimal Grid selection algorithm and developing a meta-broker is the subject of future research [16].

Notice that from the user interface point of view the PS-labeling PS-WF execution does not differ in any way from the black box PS-WF execution. Therefore it is the task of the portal to decide which algorithm is to be used for a given PS-WF. Such decision will be the subject of future research. The decision algorithm can be even more fine-grain, i.e., the portal could dynamically change the applied PS-WF execution algorithm node by node according to the size of the generated output files and the execution time of the different nodes. For example, if a node runs very quickly but generates a large local file, it would be better to execute it according to the black box algorithm.

5. Further aspects of PS-WF execution

Fault-tolerance in case of PS-WF execution has an outstanding importance since a PS-WF typically uses many Grid resources for a long period of time in order to execute all the e-WFs derived from the PS-WF. The fault tolerant execution of PS-WF can be supported in many ways by the portal. First of all, any job of

the PS-WF that is assigned to a Grid broker will be resubmitted if the broker rejects its execution on a selected Grid site. If a WF node is assigned by the user to a dedicated Grid resource and that resource is failed or after three attempts the Grid broker still fails to run a node, the particular node and e-WF will have a RESCUE status and the portal sends an e-mail message to the user (if the user requested it). The e-WF will go into rescue mode according to the Condor DAGMan control. The user can re-assign the failed node and resume the execution of the e-WF from the rescue state.

Sending e-mail to the user has a great importance since a PS-WF application can run for days or even for weeks and the user does not want to pay continuous attention to the execution status. However, there are situations where the user's intervention is needed for the portal or the user is interested in accessing the partial results of the execution. The user can define such situations for the portal and whenever such a pre-defined situation occurs the portal sends an e-mail to the user. One such situation is when a job failed and its e-WF turned into rescue state. Another important situation is when the user's certificate is getting close to the expiration. The user can ask for e-mails after completing every e-WFs or completing the whole PS-WF.

A portal can be used by many users and many of them can submit PS-WFs. This can result in an overload of either the portal or the connected Grid or both. In order to avoid such harmful situation, the portal administrator can set up certain limits for the number of workflows that can be submitted by a single user. If this number N is less than the number of generated e-WFs of a submitted PS-WF (M), then the portal simultaneously submits N e-WFs out of the potential M generated e-WFs. Whenever an e-WF is finished the portal submit another e-WF until all the M e-WFs are submitted. However, this restriction does not protect other portal users. In order to evenly distribute the portal and Grid resources among the different users, the portal administrator can also set up a certain limit for the number of jobs that can simultaneously be submitted by the portal into the Grid. This limit is applied for all the users of the portal and hence the portal tries to equally distribute the available job submissions among the actual users of the portal.

P-GRADE portal is a multi-Grid portal as it was explained in the Introduction. As such it can distribute e-WFs for all the connected Grids. The question is how to select the right Grid and the best performing Grid resource for a given e-WF or for a given node of an e-WF. The current solution is largely controlled by the user. He can directly assign a node to a certain Grid and Grid resource in the original WF. In this case all the e-WFs generated from the PS-WF created from this WF will assign the job to this Grid site. This solution must be used in Grids where broker is not available. Allocating different Grids and different resources in the selected Grids the user can achieve a static distribution of e-WF nodes among the connected Grids. The situation can be improved by assigning nodes to brokers in Grids where brokers are available. In this case the user statically defines in advance which Grid to be used for the execution of a node but gives the freedom to the Grid broker to assign dynamically the node within that particular Grid. Assigning nodes to different Grids and particularly to different Grid brokers will result in a static or semi-dynamic distributed processing of the e-WFs among the connected Grids and among the Grid sites of those Grids. A fully dynamic solution could be reached if a Grid meta-broker was available that can dynamically select Grids. Currently we are engaged in research to define and create such a Grid meta-broker [16].

Users of long running PS-WFs would like to dynamically access the partial results generated by the e-WFs and modify the input data sets according to the partial results. P-GRADE portal enables this dynamic change of input data sets. First of all, the user can check any remote output file as soon as the status of the job generating it became finished. If the user decides to modify the input data sets he can suspend the execution of the PS-WF. In this case all the already submitted e-WFs go into rescue state and the user can modify any input data. After modifying the input data set the user can make the suspended e-WFs resume their work utilizing the rescue mechanism of Condor DAGMan. Using the same mechanism even the graph structure could be modified by the user. This facility is not implemented yet and it requires further research to solve the consistency of all the e-WFs even if the PS-WF has been changed.

6. Related research

There are two main streams of research directions that have strong relationship with our work. Research on scientific workflows in Grid environments is an exciting and richly investigated subject. A good survey in

this field has been written by Yu and Buyya [17] and a whole special issue was recently devoted to this subject in Journal of Grid Computing [18]. These papers clearly show that meanwhile significant efforts are made to increase the usability of scientific workflows in Grid environment from many aspects (workflow composition, scheduling, performance estimation, fault-tolerance, etc.) there was little work on how to define and manage parameter studies at the workflow level.

The second main direction relevant to our investigation is the research on parameter studies. Some of the existing tools like Condor [1], UNICORE [19] or AppLeS (Application-Level scheduler) [2] can be used to launch pre-existing parameter studies within distributed resources. There are several important projects specifically aiming at the realization of parameter studies in Grid environments but most of them consider only individual jobs and not workflows [3], [8], [20], [21]. Usually, their main concern is the scheduling aspects of jobs in Grids although, [21] concentrates on data management. In many cases they define specialized middleware in order to optimize the scheduling [8] or introduce new scheduling concepts [3]. A good comparison of several of these tools and projects can be found in [22].

There are very few projects that deal with the integration of parameter study and workflow research. VisPortal [23] supports workflow-level parameter study applications but it is not a general purpose portal rather a specialized one to support only rendering applications. It was based on the Grid Portal Development Toolkit that is the predecessor of GridSphere used for building P-GRADE portal. ILab [4] and SEGL [7] are two main projects that provide a graphical workflow concept particularly tailored to support large parameter study applications. Notice that their goal is different from our goal. They want to support the parameter study applications by a workflow whose components are used to specify the next stage of parameter study processing activity. As a result their workflow is much more sophisticated than the P-GRADE workflow and its creation requires much more skill. Our goal was that existing DAG workflows should be enabled to be executed as parameter studies simultaneously exploiting as many Grids and Grid resources as possible.

Conclusions

The workflow concept of P-GRADE portal was very successful and popular among Grid users because its simplicity and expressiveness. Developing and monitoring Grid applications based on the workflow concept of the portal is extremely easy. Due to these advantages it was asked to set up for many different Grids (OGF GIN VO, EGRID, SwissGrid, Turkish Grid, BalticGrid, BioInfoGrid, CroGrid, Bulgarian Grid, Macedon Grid, etc.) meanwhile it runs as official portal of several Grids (SEE-GRID, HunGrid, VOCE) and serves other Grids as volunteer service (UK NGS, GILDA, etc.). The feedback from the users made it clear that they want a parameter study support at the workflow level but in a way that keeps the simplicity and expressiveness of the original workflow concept. Based on their request we have extended the portal with the workflow-level parameter study support. The new version of the portal has been prototyped and was publicly demonstrated at the EGEE conference in September 2006. The new version of the portal (version 2.5) that gives service quality full support for the workflow-level parameter study will be released in November 2006.

We have described two algorithms for executing PS-WFs in a multi-Grid environment. The black box algorithm gives an optimal solution concerning the utilization of storage resources while the PS-labeling algorithm optimizes the utilization of processing resources. Further research is required to create a dynamic and adaptive integration of the two methods where the portal can dynamically decide which method to be used for a particular node of the PS-WF graph. The current execution method of PS-WFs enables the static distribution of nodes between different Grids and different Grid resources if brokers are not available in the connected Grids. If brokers are available Grid resources can be assigned dynamically but the Grid assignment is still static. To provide a fully dynamic allocation of Grids and Grid resources a meta-broker should be developed and connected to the portal and to the Grids. The development of such a broker is subject of further research in the framework of the EU CoreGrid project.

References

- [1] Thain, D., Tannenbaum, T., and Livny, M., Condor and the Grid, in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0., pp. 299-336, 2003.
- [2] Casanova, H., Obertelli, G., Berman, F. and Wolski, R., The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid, Proceedings of the Super Computing (SC 2002) Conference, Dallas / USA, 2002.
- [3] Abramson, D., Giddy, J., and Kotler, L., High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, IPDPS'2000, Mexico, IEEE CS Press, USA, 2000.
- [4] Yarrow, M., McCann, K., Biswas, R. and van der Wijngaart, R., An Advanced User Interface Approach for Complex Parameter Study Process Specification on the Information Power Grid, Proceedings of the 1st Workshop on Grid Computing (GRID 2002), Bangalore / India, December 2000.
- [5] Yarrow, M., McCann, K. M., Tejnil, E., and deVivo, A., Production-Level Distributed Parametric Study Capabilities for the Grid, Grid Computing - GRID 2001 Workshop Proceedings, Denver, CO, November 2001.
- [6] McCann, K. M., Yarrow, M., deVivo, A. and Mehrotra P., ScyFlow: An Environment for the Visual Specification and Execution of Scientific Workflows, GGF10 Workshop on Workflow in Grid Systems, Berlin, 2004.
- [7] N. Curre-Linde, F. Boes, P. Lindner, J. Pleiss and M.M. Resch, A Management System for Complex Parameter Studies and Experiments in Grid Computing, in: Proc. of the 16th IASTED Intl. Conf. on PDCS (ed.: T. Gonzales), Acta Press, 2004.
- [8] Casanova, H. and Berman, F., Parameter Sweeps on the Grid with APST, in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0., pp. 773-788, 2003.
- [9] P. Kacsuk and G. Sipos, Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal, Journal of Grid Computing, Vol. 3, No. 3-4, pp. 221-238, 2005.
- [10] P. Kacsuk, T. Kiss and G. Sipos, Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level, Proc. of the Grid-Enabling Legacy Applications and Supporting End User Workshop, in conjunction with HPDC'06, Paris, pp. 3-7, 2005.
- [11] T. Delaitre, et al., GEMLCA: Running Legacy Code Applications as Grid Services, Journal of Grid Computing, Vol. 3, No. 1-2, pp. 75-90, 2005
- [12] R. Lovas et al, Dynamic workflows in the service-oriented P-GRADE portal using Grid superscalar, Austrian Grid Symposium, Innsbruck, 2006.
- [13] J. Frey, Condor DAGMan: Handling Inter-Job Dependencies, <http://www.cs.wisc.edu/condor/dagman/>, 2002.
- [14] <http://www.gridisphere.org/>
- [15] http://www.lpd.sztaki.hu/pgportal/v23/manual/users_manual/UsersManualReleaseV2.html
- [16] A. Kertesz and P. Kacsuk, Grid Meta-Broker Architecture: Towards an Interoperable Grid Resource Brokering Service, CoreGRID Workshop on Grid Middleware in Conjunction with EuroPar'06, Dresden, 2006.
- [17] J. Yu and R. Buyya, Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing, Vol. 3, No. 3-4, pp. 171-200, 2005.
- [18] E. Deelman and I. Taylor (guest editors), Special Issue on Scientific Workflows in Grid Environments, Journal of Grid Computing, Vol. 3, No. 3-4, pp. 151-304, 2005.
- [19] Erwin, D. (Ed.), Joint Project Report for the BMBF Project UNICORE Plus Grant Number: 01 IR 001 A-D, Duration: January 2000 to December 2002; ISBN 3-00-011592-7.
- [20] Abramson, D, Lewis, A. and Peachy, T.: Nimrod/O: A Tool for Automatic Design Optimization, The 4th International Conference on Algorithms & Architectures for Parallel Processing (ICA3PP 2000), Hong Kong, 2000.
- [21] H.A. James and K.A. Hawick, Scientific Data Management in a Grid Environment, Journal of Grid Computing, Vol. 3, No. 1-2, pp. 39-51, 2005.
- [22] DeVivo, M. Yarrow, and K. McCann, A comparison of parameter study creation and job submission tools, Technical Report NAS-01-002, NASA Ames Research Center, Mo#ett Field, CA, 2001.
- [23] <http://www-vis.lbl.gov/Publications/2004/LBNL-PUB-893-Visportal.pdf#search=%22%22parameter%20study%22%20AND%20%22Grid%20portal%22%22>