Supporting dynamic parameter sweep applications in workflows – lessons learnt from the CancerGrid project

Péter Kacsuk, József Kovács, Gergely Sipos, Gábor Hermann, Krisztián Karóczkai

MTA SZTAKI, Parallel and Distributed Systems Laboratory, 1111, Budapest, Kende u. 13-17, Hungary
{kacsuk, smith, sipos, ghermann, krisztian.karoczkai}@sztaki.hu

Abstract. A vast majority of scientific grid applications are either parameter sweep applications or a significant subpart of these applications belong to class of parameter sweep activities. The paper describes a new graphical workflow language in which any node of the DAG-based workflow can be a parameter sweep node and the execution of these nodes are transparently executed either in service grids or in desktop grids depending on the computational complexity of the workflow node. The new concept is supported by the CancerGrid portal that has been established for a chemist community.

Keywords: grid computing, scientific workflow, parameter sweep applications, desktop grids.

1 Introduction

The original P-GRADE workflow system was based on DAG and used Condor DAGMan to manage DAG workflows [1]. However, user feedbacks showed that parameter sweep (PS) support is highly needed for many grid applications. So P-GRADE workflow concept was extended with the workflow level parameter sweep support [2]. In the new P-GRADE portal workflow system (from version 2.5 and above) [3] any existing workflow as a black-box can be turned into a parameter sweep (PS) application by defining some of the input ports of the workflow to be PS inputs. P-GRADE automatically creates the cross product of these PS input files and generates as many executable workflows (e-workflows) as many input files sets are in the cross product of PS input files [2]. Moreover, we introduced special generator type workflow nodes to conveniently generate the necessary PS input files at the PS input ports and collector type nodes to collect and process the result of e-workflows. As a result a typical PS application is created as a three-stage workflow graph.

1 The research and development published in this paper is partly supported by the European Commission under contract numbers LSHC-CT-2006-037559 (FP6 STREP, CancerGrid) and RI-211727 (FPF7 IST Capacities, EDGeS), and by the Hungarian Government under grant TECH_08-A2/2-2008-0097 (WEB2GRID).
first stage contains as many generators as the user wants to apply. They all work in parallel producing all the necessary input file sets for the PS input ports of the core workflow. When all the generators have been completed the P-GRADE workflow manager creates the cross-product of the input file sets and generates an e-workflow of the core workflow for every member of the input file sets. Of course, all these e-workflows can run in parallel leading to the exploitation of a very high degree of parallelism in executing P-GRADE workflows. Finally, when all the e-workflows have been finished the last stage is started when all the collectors connected to the output ports of the core workflow are executed in parallel. The tasks of the collector nodes are typically creating statistics and visualization of the execution results of the e-workflows.

The advantages of the three stage concept are that it is very easy to understand and use, and furthermore it can fit to a very large class of PS workflow applications. However, there are applications where this simple concept is not sufficient; we need more dynamicity in the representation of PS features [4][5]. Such an application area was found in the EU FP6 CancerGrid project [6] where large number of molecules should be checked against some criteria to be applicable as potential anti-cancer drug. Therefore intensive research started to extend the PS capabilities of P-GRADE towards a very flexible PS workflow concept.

In order to achieve such a flexible PS workflow concept we dropped the Condor DAGMan workflow engine and developed a new workflow engine. At the same time we restructured the original monolithic software architecture of P-GRADE portal and created a new flexible, SOA based architecture that can support the execution of generic PS workflows on local systems (PC, Cluster), on remote systems accessible as Web Services and on various Grid systems like GT2, GT4, LCG-2, gLite, etc. Different nodes of a PS workflow can be executed simultaneously on any of the resources mentioned above. The new system is called gUSE (grid User Support Environment) and the portal using the services of gUSE is called WS-PGRADE showing in the name that this new portal already supports WS access.

The paper is organized in the following way. Section 2 explains the main features of the new PS workflow concept of WS-PGRADE. Section 3 shows a workflow that supports the work of the chemists in their anti-cancer drug research. In this workflow the features of the new PS workflow concept are exploited. Section 4 introduces the CancerGrid portal and gives some insight of its internal operation. In section 5 our experiences are summarized briefly and section 6 discusses the most related works. Finally, conclusions are summarized in section 7.

2 The new PS workflow concept of WS-PGRADE

In the three-stage PS workflow concept of P-GRADE portal we identified the following main limitations:

1. The generators and collectors could not be placed at any node of the workflow and could not be applied at any stage of the workflow execution.
2. The PS execution could be applied only for the core workflow but there was no possibility to apply it for individual nodes of the workflow.
3. Since the core workflow was executed many times according to the number of input files even those nodes of the core workflow that were creating always the same output file were redundantly executed all the times.

4. Only cross product operation was allowed on the PS input ports. Dot product was not applicable.

5. Abstract workflows and concrete workflows were not distinguished. The user should specify every execution details at workflow specification time except the target grid sites since they were chosen by the underlying grid broker.

6. There was no workflow repository through which application developer and end-users could collaborate and from where ready to use workflows and workflow templates could have been downloaded.

In order to overcome these difficulties we developed a new PS concept to be used in the new WS-PGRADE workflow system. The new concept has the following main features:

1. The concept of workflow execution stages has been removed. The generators and collectors could be placed at any node of the workflow and could be applied at any point of the workflow execution time.

2. The PS execution could be defined for individual nodes of the workflow and hence for each node of the workflow individually can be decided how many times the given node should be executed. As a result superfluous execution of workflow nodes has been eliminated.

3. Both cross-product and dot-product relationship among input ports of a workflow node can be defined. Moreover, they can be applied in a mixed way among the input ports of any node.

4. Workflow abstraction has been introduced at four levels. We distinguish the following ones:
   a. Workflow graph: defining only the topology of the workflow graph
   b. Workflow template: a workflow where some of the parameters are already fixed and some of them are open to be set up by the user of the template
   c. Concrete workflow: defining every parameters of the workflow that is needed to successfully run it
   d. Workflow instance: an instance of a concrete workflow that is under execution

5. Workflows can be embedded at any node of the workflow. Even recursive embedding is allowed. As a result loop constructs can be created although there is no control arc in the workflow graph.

6. Workflow repository has been introduced in order to publish completed workflow applications by developers to end-users and also for publishing any types of workflows in order to make collaboration possible between application developers at any phase of the development.

The new PS workflow concept of WS-PGRADE is illustrated on Figure 1. Node A has two input ports with m and n input files, respectively. The relationship between the two input ports is defined as cross-product so node A will be executed m*n times with all the possible combinations of the m*n input files and will produce m*n output
files on its output port. Node B is a generator type of node that will produce K output files for every execution. Since it has h input files it will generate h*K output file on its output port. Node C receives m*n inputs from node A and h*K inputs from node B. Since the input relationship is cross-product it will run m*n*h*K times and will produce m*n*h*K output files. Notice that the input ports of node A and B are external input ports and hence it is the user who has to define the number of input files for these ports. On the other hand the input ports of node C are internal ports connected to the output ports of other nodes of the workflow and hence the number of input files on these ports is defined by the connected output ports.

Node D has also two internal input ports but their relationship is defined as dot-product. This node will be executed by S times where

\[ S = \max(m*n, h*K) \]

During the ith execution node D takes the ith input files from both input ports as long as

\[ i \leq \min(m*n, h*K) \]

When the input files from one of the input ports are consumed node D uses some special value for this input port until there is input file on the other input port. The special value can be defined by the user.

Node E has a special collector input port and hence node E works as a collector node. It means that all the files coming from node C will be considered and processed as part of a single input file. The other input port receives S files from node D. The relationship of input ports is defined as dot-product and hence node E will be executed S times. Finally, node F receives S files on both input ports and since their relationship is dot-product node F will be executed S times.

As a summary we can say that the WS-PGRADE workflow can contain generators, collectors and PS nodes at any position of the workflow without any restriction. The relationship of input ports can be cross-product, dot-product and any combination of

![Figure 1 The new PS workflow concept of WS-PGRADE](image-url)
these products. In the next section we show how the new PS features of the WS-PGRADE workflow contribute to the solution of a real problem.

3 CancerGrid application case study

The CancerGrid project started in 2007 January with the goal to develop new algorithms and application that can accelerate anti-cancer research. The new algorithms are expected to run with many different molecules and hence an institutional (local) desktop grid system will be created and used by the project partners to run the new applications and algorithms. One of the new applications is QSAR model building. In the CancerGrid project our task was to enable the calculations of molecular descriptors on the grid to support faster QSAR model building. Molecular descriptor calculation required the following algorithms to run either on a database server, on a local resource or inside a local desktop grid:

- DBRead to read molecules from the CancerGrid database
- DBWrite to write molecules into the CancerGrid database
- 2D-3D molecular structure converter algorithm (2D3D)
- 3D conformer generator algorithm (FlexMol)
- semiempirical calculation of 3D molecules (Mopac)
- codessa pro descriptor calculation algorithm (MolDesc)
- molecule structure file converter algorithms (Convert, Convert2)

![Figure 2 PS workflow application of CancerGrid](image)

The workflow of the CancerGrid application is shown in Figure 2. There are two generator nodes inside the workflow. Node Convert generates N output files out of its
single input file coming from DBRead. DBRead has the task of reading the user selected list of molecules from a relational database. The typical value for N is 30,000, which is the number of selected molecules. It means that nodes 2D3D, Flexmol and Convert2 will run 30,000 times. The second generator node is Convert2 that generates M outputs for every input. The typical value of M is 100 and hence Convert2 will generate 30,000*100 = 3,000,000 output files. As a result nodes MolDesc, Mopac and DBWrite (storing the results in a relational database) will run 3,000,000 times. This example shows how the WS-PGRADE workflow concept enables the description of very dynamic and large applications containing nodes to be run even million times.

4 The CancerGrid portal

The CancerGrid portal is based on the WS-PGRADE/gUSE portal software. It is deployed on a central server, where a desktopgrid server (based on BOINC [9]) is also installed to coordinate the execution of the high number of jobs generated by WS-PGRADE. In the current configuration WS-PGRADE handles three different job types:

- short-running jobs: these jobs execute for only a short time (i.e. converters in the workflow), they are assigned to local dedicated resources
- web-service calls: reading and writing data from/to the molecule database by the workflow are implemented by web-service calls.
- long-running jobs: these jobs are scheduled to be executed under BOINC

The resource type for the jobs can be defined by the workflow developer, but the grid to which the job should be submitted is decided by the broker component of gUSE. In this way different nodes of the same WS-PGRADE workflow can simultaneously be executed in different grids based on the decision of the user or of the gUSE broker [7]. WS-PGRADE and BOINC is connected through a so-called 3GBridge that is able to convert WS-PGRADE jobs into BOINC workunits. BOINC clients are attached to the server in order to process the workunits. These clients are collected by the consortium partners, since a non-public type i.e. local desktopgrid called SZDG [8] is formed in this project.

The clients are processing the workunits containing the GenWrapper tool and the computational application itself. The GenWrapper tool provides an interface between the BOINC client and the application. The client requires some functionality from the application to execute it properly. Since the GenWrapper “talks to” the BOINC core client on behalf of the application, modification (i.e. “boincification”) of the application is not necessary. This way the execution of legacy applications under BOINC is also supported and more than one job can be grouped together into a workunit in order to increase granularity if required. Job grouping is implemented by the 3GBridge component with support of the GenWrapper tool.

A typical usage scenario happens the following way: A user first logs into the portal and goes to the molecule database browser. In that browser, lists of molecules are shown which can be manipulated by the user. Molecules in the list can also be
inspected, modified, uploaded, etc. When a list of molecules is prepared the user
downloads a predefined workflow from the repository, assigns the list to the
workflow and configures the parameters of the various tools. A configured workflow
is then submitted and its status can be monitored. Results are inserted into the
molecule database that can be inspected with the same molecule database browser.

As a summary the WS-PGRADE/gUSE portal in the CancerGrid project
- provides web access for the users
- stores predefined workflows in its repository
- provides access to the molecule database for the chemists
- provides workflow configuration and submission mechanism
- coordinates workflow execution
- manages underlying computational resource types
- provides monitoring of the progress of the workflows

The CancerGrid portal is operated and actively used by the CancerGrid consortium
to help the chemists finding new anti-cancer related drugs. The portal is continuously
improved based on the users’ feedback.

5 Lessons learnt

Comparing the P-GRADE 2.x series to the WS-PGRADE/gUSE portal a significant
progress happened on the following two fields: workflow concept and performance.
The main changes regarding the workflow concept are detailed in Section 2. With the
introduction of PS nodes in workflow, users can easily generate hundreds, thousand
(or even millions) of jobs. In the P-GRADE 2.x series the number of jobs was limited
to a few hundreds and the number of workflows executed at the same time can only
be a few. In WS-PGRADE (3.x series) this limitation is meaningless, since it would
eliminate the power of the PS concept. Therefore, in order to increase the capacity of
the portal a more robust and scalable middleware (i.e. gUSE) was developed. While
P-GRADE was implemented as a monolithic component, WS-PGRADE/gUSE is
realized by a set of web-service components that can be hosted on several machines.
As a result, the WS-PGRADE/gUSE system can handle millions of jobs and the
number of workflows handled by the portal is not limited in the current version.

After stepping into a new dimension regarding the number of jobs and workflows,
the new portal required a grid that is able to handle large amount of jobs. The new
portal therefore is extended with the ability to submit jobs into DesktopGrids through
the 3GBridge component. Based on our experiences, desktop type grids are more
appropriate for a large number of parameter study applications and it is much easier to
be built by a community, like the chemists, than building a service grid.

When having a huge number of jobs to execute and a powerful Desktopgrid, the
throughput of the portal becomes a significant factor. During the development the
efficiency of the workflow interpreter, portal storage and the submitter (that delivers
the jobs to the grid) was a crucial point. Technical details about the internals of the
gUSE components are out of scope of this paper, but the lessons we learnt from the
CancerGrid portal is that the throughput is very sensitive on the strategy of handling
the enormous number of jobs.
The performance of the current CancerGrid portal has been inspected during the operation and the analyses shows that it is able to dispatch a few million jobs per one day including all the coordination activities (dependence checking, scheduling, etc.), all preparation (e.g. staging i/o files) and submission for each job. The job dispatch performance of course is very sensitive to the workflow structures which are basically predefined by the CancerGrid project.

The CancerGrid portal is already used to process thousands of molecules, the aim of the project is to process (called screening) 200,000 compounds stored in the molecule database integrated to the portal. The number of client machines is continuously increasing (consortium members are attaching their machines) and the aim is to reach a few hundred machines (of which are mostly dedicated) to get more performance.

Another important point is that while the development first targeted the chemist users, CancerGrid specific features (e.g. molecule database browser) were added to the WS-PGRADE/gUSE portal as pluggable components. In this way, the core of the portal is still general enough that it can be extended for any other communities.

6 Related work

It has been recognized by several user communities that grids are most suited for parameter study applications, and consequently there are quite a few tools and environments that ease the management of such applications. Condor [10] was one of the first middleware on production grids with supports for parameter studies, for data driven workflows, but not for the composition of them. If a Condor user would like to execute a workflow over a large parameter space then this must be done manually by creating several copies of the workflow. GANGA [11] and GridWay [12] are two user-level service that extend the Globus and gLite grid middleware. They introduce new command line programs for users that are capable to manage parameter studies of jobs. Only few tools exist that combine parameter studies with workflows on grids. ILab [13] enables the creation of special parameter study oriented workflow: On a sophisticated graphical interface the user can explicitly define how to distribute and replicate the parameter files in the Grid and the number of independent jobs to launch for each segment of the data. The approach results very static processing pattern and fails to meet the demand of exploiting the resources of Grids in a dynamic fashion. SEGL [14] puts much more emphasis on dynamism. It provides a GUI where users can define workflows and a workflow supports several levels of parameterization, repeated processing, data archiving, handling conclusions and branches. The Workload Management System (WMS) service of the gLite middleware supports DAG workflows, parameter studies and their combination. Because a WMS installation runs as a dedicated service of one Virtual Organization, a workload can use services of one Virtual Organization, meanwhile WS-PGRADE can use services from several Virtual Organizations and Grids, moreover, WS-PGRADE can mix Grid jobs, Web service invocations and database management operations within a single application. The Abstract Grid Workflow Language (AWGL) [15] enables the combination of parameter study and workflows and contains several patterns to
distribute data among jobs. However, the developer of the application must split workflows into parallel jobs manually in order to execute them on multi-core architecture simultaneously.

In [16] a data oriented system for desktop PC environments is introduced. The solution is capable of setting up a peer-to-peer network of .NET services on PCs to process the users’ data according to a pre-defined data-flow graph. With the Java based API users can define the activities of workflow nodes and the topology of graphs. However the system neither supports the usage of parameter study nodes inside workflows nor the parameter study execution of workflows.

Taverna is a popular workbench especially among biomedical research to define and execute dataflow applications [5]. Taverna interprets SCUFL workflow language and supports workflows of Web services. SCUFL supports dot and cross product and one can achieve parameter study execution of workflows with these data-pairing concepts. However, Taverna executes the whole workflow for each element of the input data set there is no possibility to define different parameter spaces for different workflow nodes. Taverna workflows cannot exploit grids to execute computation intensive nodes.

7 Conclusions

The new generation of the P-GRADE workflow is much more flexible than the first generation was. Although it does not contain any control arc and the graph structure is still a DAG, users can easily organize repeated actions in the graph based on its node level parametric sweep execution feature and/or by recursively embedding graphs at any node. The new system is also very scalable. Large applications requiring millions of executions of nodes of a workflow can easily be defined by the new graphical user interface and can be executed by the new workflow engine capable of submitting jobs to various grid systems in an adaptive way.

There are many workflow systems [17][18][19], but none of them considered so far transparently submitting sequential PS workflow nodes into desktop grid systems. A major contribution of our research is the workflow engine capability of transparently submitting sequential PS nodes into not only service grids but also to desktop grids. Since desktop grids are volunteer grids they can collect much more resources than service grids on a fraction of the cost of service grids. Hence directing sequential PS nodes of a workflow to desktop grids instead of service grids has a major impact on the future grid sustainability, too.

References