YML/LAKe Web Portal: Web based research portal for linear algebra application

Olivier Delannoy and Nahid Emad

PRiSM Computer Science Laboratory, Versailles University
45, avenue des Etats-Unis, 78035 Versailles CEDEX FRANCE
{Olivier.Delannoy,Nahid.Emad}@prism.uvsq.fr

Abstract

In the passed decades the development of the numerical libraries was a response to the needs in the field of high performance computing. With the advent of the GRID computing, it becomes necessary to adapt these libraries and or their subroutines as components and to integrate them in a global computing system. To allow the end-user to be concerned by its science activity and not by the implementation details of the tools used, the infrastructures complexity of the GRID must be hidden to him. Indeed, a GRID computing environment, proposing services such as numerical libraries, must offer a convivial web portal to the user. GRID portals can interact with underlying middleware using integrated workflow engines and with the components constituting a library relying integrated implementation catalogs.

This paper presents a web based solution for scientific computation portals. The proposed portal example is based on the integration of LAKe, a component-oriented library in the YML workflow engine. YML is composed of several tools allowing workflow graph description adaptable to several middleware and LAKe is an object-oriented linear algebra class library. By using a representative numerical hybrid method we show the interest of our high-level application portal.

INTRODUCTION

In high performance computing, the demand for re-use led the design and the development of the numerical libraries towards an oriented-component approach. It is possible to adapt some applications of the scientific computation to the GRID environments. In addition, the class of numerical applications constituted by the hybrid methods is well adapted to global computing. Indeed, these applications have asynchronous communications between their coarse grain subtasks, are fault tolerant and present dynamic load balancing potentials. For all these reasons it is necessary to design or adapt existing numerical libraries to GRID environments. However, of share the complexity of the environments like that of the quoted numerical algorithms, the use of these libraries becomes very difficult for the end-user. It is thus necessary to design and develop convivial user interfaces.

The NetSolve/GridSolve[5] project is being developed at the Innovative Computing Laboratory of the University of Tennessee Computer Science Department. It provides remote access to computational resources including hardware and software and it supports different architectures and operating systems, including Unix and Windows platforms. Because of its deployable software architecture, NetSolve has been used with other GRID middleware such as Globus[8] and Condor[9]. In order to monitor the execution of applications in NetSolve, one can use VisPerf[1]. This tool provides information concerning the NetSolve environment. Nevertheless, it does not allow application level events, instead it focuses on maintaining, in real-time, a snapshot of the underlying middleware as well as the list of resources involved in the execution of the local client. The execution monitoring tool discussed here rests heavily on the application, even for information used to create the middleware snapshot. YML automatically integrates support for such information in all applications. However, there is no global overview of the whole middleware activity. The UNICORE[7] GRID technology provides a seamless, secure and intuitive access to distributed GRID resources. UNICORE graphical client proposes similar monitoring capabilities. It also allows per application specific monitoring through the use of plugins on the client. BOINC[10] (Berkeley Open Infrastructure for Network Computing) is a software system that makes it easy for scientists to create and operate public-resource computing projects. In systems similar to BOINC, users are on the one hand volunteers providing computing resources and on the other hand scientists in charge of designing and installing the application. The BOINC project provides a web interface to monitor the application and to create a new *community*.

This work was supported by the French Ministry of Research

This article discusses a web portal solution dedicated to linear algebra software study in the context of GRID computing. The solution proposed here relies on the YML framework[2], which allows workflow graph execution over large scale middleware. The web portal acts as a client for the YML framework and targets scientists audience. The goals of this portal encompass easing experimentation, helping scientific communication between distant research groups.

Following this introduction, the second section gives an overview of the YML framework. The third section presents the web portals and its integration with YML. The fourth section discusses the example of a linear algebra method for computing a few number of eigenvalues within YML and the web portal. The last section presents conclusions and future works.

The YML FRAMEWORK

Research efforts on GRID and peer to peer middleware concentrate on increasing the global efficiency and on providing more and more services to the user. A few projects focus on the study and design of tools to simplify the use of this kind of platform. The available solution for such environments can be classified in three main categories. Message Passing Interface and similar API adapted to GRID and peer to peer middleware is one of the most interesting solutions for the user. She/he is able to use its existing application directly over large scale middleware. However, the behavior of a high performance computer and of a grid middleware is really different and most applications require updates in order to be efficient. The RPC approach is available with almost any GRID middleware. This approach is well adapted to the GRID middleware and numerous projects such as OmniRPC[6] and NetSolve make use it. Another approach consists in workflow engines. It is based on a program that manages the execution of the computation involved in an application. This is the solution used in YML.

The YML framework provides the end-user with a set of tools to develop and execute applications over large scale architectures. It defines an abstraction and hides the specificity of each middleware. The abstraction rests on a dedicated programming language, called YvetteML. It is organized in two distinct aspects. The first one enables the description of components using XML. A component in YvetteML is a chunk of computation without communications. A component acquires data before its computation starts and exports data afterward. The second aspect is used to link components together to create a complex application. The user describes the computation of its application using the component description aspect and uses a graph language to describe the communication of its application. An application developed using the YvetteML language should be ready-to-use on multiple middleware. The YML framework strictly separates middleware specific information from the application description. The same compiled application can be executed on multiple middleware. Graphs described by the YvetteML language are fully expanded during the compilation process: loops are unrolled, condition evaluated, unvisited branches spread out of the graph and constants are propagated.

The YML framework separates in two parts. The user view is middleware independent and contains the main services of the YML framework. These services consist in a compiler for the YvetteML language, a real time scheduler and a development directory. The middleware independent part is associated to a backend for middleware dependent services. Figure 1 describes the overall organization of the YML framework. This figure highlights the different parts of the YML framework and clearly shows the middleware specific services known as the backend. The web portal positions itself as a standard client for the YML framework. The testbed platform relies on the XtremWeb[11] middleware.

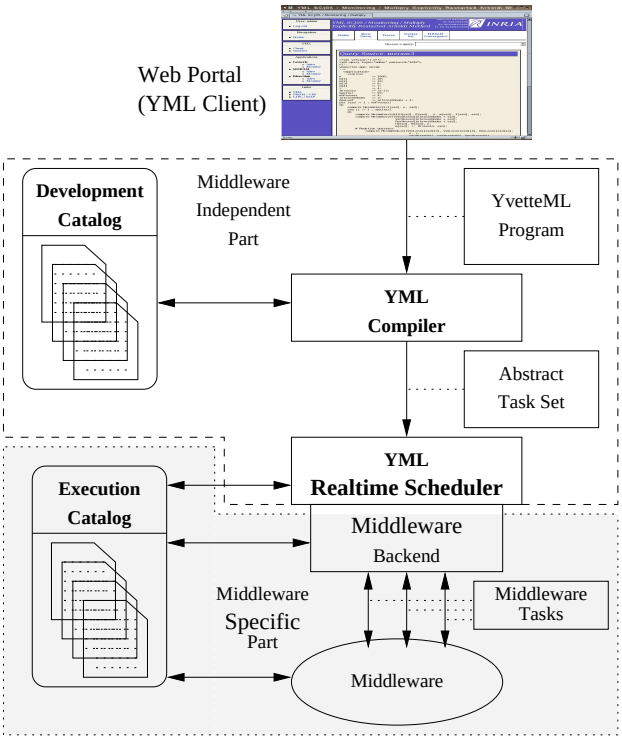

Figure 1: The YML framework organization

The list bellow describes the role of each tools composing the YML framework:

- **Compiler:** It translates applications described using the YvetteML language to a set of components calls. Its component call is decorated with two pieces

of information. The execution condition is a boolean expression which determines whether the component can be executed or not. The post condition is a list of boolean flags used to describe the state of the application. Execution conditions are evaluated based on the events.

- **Scheduler:** It manages application executions. It acts as a client for underlying middleware accurately requiring computing resources. During the application execution the scheduler detects task ready for execution solving dependencies at runtime. Each scheduling iteration may or not generate a set of parallel tasks which are translated in computing requests to middleware through dedicated backends.

- **Development Catalogs:** The YML framework stores components in Catalogs. The *Development Catalog* store information used only during the development stage. The middleware independent part relies only on this catalog. The *Development Catalog* store components information and data type information used to validate the YvetteML input program. This catalog is not required for executing an application.

- **Data Repository:** The YML framework implies a lot of data exchange through the network. The Data Repository server act as a resource provider and delivers data to each components on demands. Note that the repository does not appear in figure 1.

- **Backend:** All middleware specific services are encapsulated in a backend. A backend generally consists in a YML worker which constitute the component execution supporting layer to be executed on remote peer and a client for the middleware itself. It is easy to add support for new middleware in YML. Backends can rely on several services provided in YML such as the Data Repository client library and the component generator. YML comes with a default backend for the XtremWeb[11] GRID middleware.

WEB PORTAL

YML is used as foundation for the web interface. YML enhances the abstraction level so that this portal does not depend on the underlying middleware used. We are interested in a user interface that achieves several goals. The user interface should help in the experimentation process. Targeted applications have numerous parameters. Some of them are purely practical and require a lot of expertise specific to each method on one hand and on each data set on the other hand. In the mean time users interested only in computation results should not have to select all parameters for several reasons and especially because all of them are not experts of the used numerical method. For such users a guided approach must help them in selecting all parameters of the methods. Finally, the web portal is used as a scientific communication tool. Using such web portals one can evaluate and comment results described in publications. It has also been used as a live presentation tool for YML in several events and in particular at Super Computing in November 2005 [13].

The web portal was designed with the goal of being an experimentation tool that helps us increase our knowledge and our understanding of the influence of the various parameters involved in hybrid linear algebra methods. An overview of the methods and their adaptation to YML is discussed in the next section. The web portal integrates a full featured YML client. It consists in queries submission facilities, a queries manager, a global status to monitor YML activity. These generic services are well adapted for application conception and application testing. Once an application becomes stable enough for experimentation, then it is interesting to provide specific features to help the user with query submission and monitoring. The web portal is organized in applications. Each application comes with several operations. Each operation is associated to a dynamic web page. The available operations are listed below :

- Introduction: This page is static and describes the application. It also lists available monitor views for this application.

- Submission: This is a wizard used to construct the YvetteML program of the application. All submission wizards start with one or several parameters screens, followed by a code generation page and finally a query submission to the YML Manager. Submission wizard acts as a dedicated client for the YML framework.

- Manager: This page lists all submitted queries. A query can be in one of the following states: ready for execution, running and finished. The user can start a query execution, stop a running query and retrieve finished query results. This page is an application specific query manager.

- Monitor: This page provides several views of a query. All views rely on the analysis of the query logs collected during the execution of all tasks as well as of the data repository associated to the query. The web portal defines a set of common views available to all applications. These views are used to display the state of the query, the source of the query and the processors or peers involved in the computation of the query. Applications can also add their own view and provide application specific displays of the query execution.

All these views are dynamically computed depending on the information provided by the query execution.

The submission page provides the user with a simple and guided way to launch an execution. A user interested only in the computation results can benefit from an automated parameters selection and configure the methods quickly. The current submission facilities are the first step toward an advanced decision making tool with automatic learning features.

The most interesting aspect of the application specific portal facilities consists in the monitoring features. The monitor presents the application execution in real-time in several views. Each view is dynamically created and based on the analysis of the log of the query. YML component generator integrates information in the log of the execution of each component in order to construct views. This generic information is automatically available thanks to the component generator of YML. Application specific information can appear in the implementation component in order to feed application specific views. Figure 2 presents one standard view. It displays the YML program associated to the query currently displayed. The query corresponds to a MERAM process which will be described in the next section. The figure shows that a MERAM application defines five views. It makes use of four standard views: Status which displays statistics on the application execution (number of tasks waiting, executing, etc), Query showing the YvetteML program of the application, Log displaying the log of the application execution and finally Workers listing the workers and the number of times they contribute to the application. The last view is discussed in the next section.

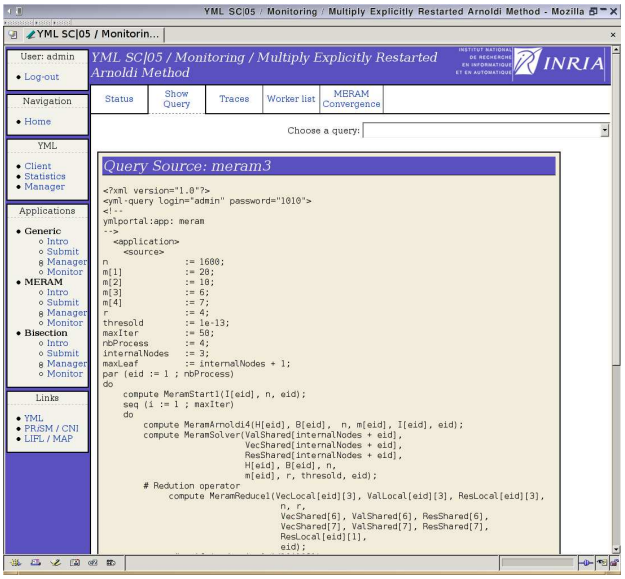

Figure 2: YML Query Display: A standard monitoring view

YML/LAKe

Linear Algebra Kernel[4] (LAKe) is an object oriented library dedicated to linear algebra methods development and especially in the computation of eigenelements. This library eases the development of iterative methods. The library integrates all supporting classes required for the calculating of some eigenpairs of large sparse non-Hermitian matrices using some iterative methods named Explicitly/Implicitly Restarted Arnoldi Method (ERAM/IRAM). These methods are well adapted to hybridization. An hybrid method is composed of several methods collaborating to enhance the convergence speed of one of them. Traditionally, we distinguish a special case of hybridization called multi-methods. It consists in several instances of the same method executed in parallel (or not). Each method starts with a different set of initial guesses and exchanges results during the restarting step. Multiply Explicitly Restarted Arnoldi Method (MERAM)[3] is a multi-method based on several instances of the ERAM Process executed in parallel. This hybrid method is well adapted to the GRID computing environments because of its coarse grain sub-tasks, built-in fault tolerance and dynamic load balancing potentials.

The coarse grain sub-tasks aspect of MERAM and similar hybrid methods are really interesting in the context of GRID computing. YML defines a component model requiring the application to be split in chunks of computation requiring no inter-components communication. Those chunks match the coarse grain sub-tasks specificity of hybrid methods. The computation of an ERAM process involved in the computation of a MERAM method can be cut in several steps as follows. The first step is called the Arnoldi Reduction(AR) and constructs a similar problem of a smaller size which will be used for the second step. The second step(S) resolves the eigenpairs problem on the smaller problem and creates an approximation of the solution. The accuracy of this solution is checked and if user criteria were not matched a restarting step is initiated. The restarting step consists in a reduction(Re) operation and the appliance of a restarting(R) strategy. The output of the second step consists in the expected eigenvalues and the eigenvectors associated named Ritz vectors. The reduction operator selects the best vector computed in all of the ERAM process for each eigenvalue independently. Restarting strategies are numerous.They consist in combining the eigenvectors selected during the reduction operation in order to head the method toward the solution. Restarting strategies details are out of the scope of this article, one can consult[12] for a detailed discussion on this topic.

MERAM has been adapted to YML using the LAKe object oriented library. Adapting LAKe to work with YML consists in decomposing the ERAM process to fit the step highlighted previously. The YML graph language (Yvet-

teML) splits the application in two aspects a control aspect and a computing aspect. In YML the control of the application is externalized outside the application itself. All information needed to compute the solution is transmitted through component communication channels or parameters. It is a typical behavior when using stateless components models. Figure 3 displays a rolled up version of the graph corresponding to a MERAM made of three ERAM processes. MERAM is not limited to three ERAM processes. In figure 3, solid arrows correspond to strict dependencies while dashed arrows correspond to optional dependencies. ERAM processes involved in a MERAM are independent and progress at different speed depending on their initial guess. They also constitute the application level fault tolerance mechanism available in MERAM ; any ERAM process can fail until only one remains. The first iteration of MERAM differs from the other and require an additional step (I) which consists in creating an initial vector (it corresponds to the restarting vectors of all other iterations).

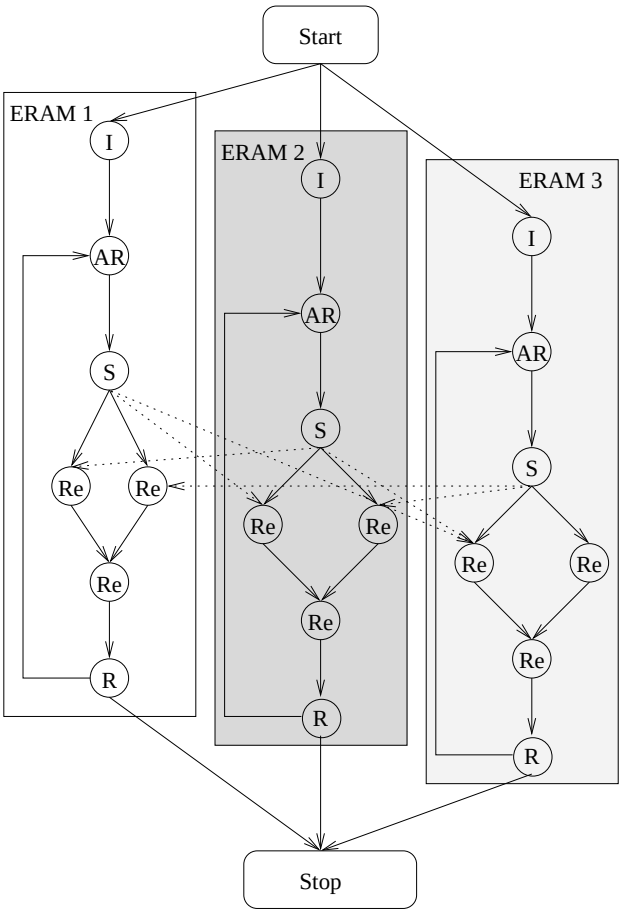

Figure 3: YML control graph for MERAM

An important parameter required in each ERAM process involved in MERAM is the size of the sub-space. This size corresponds to the size of the smaller problem. In order to understand the impact of this parameter on the convergence speed of the method, we define one special view consisting of one curve per ERAM process. Each curve denotes the accuracy of the solution computed at the current iteration. Figure 4 shows the execution of a MERAM process consisting of three ERAMs. The horizontal line at the bottom of the graph corresponds to the accuracy requirement specified in the query. The MERAM stops its execution once one process converges (finds solutions with an acceptable precision).

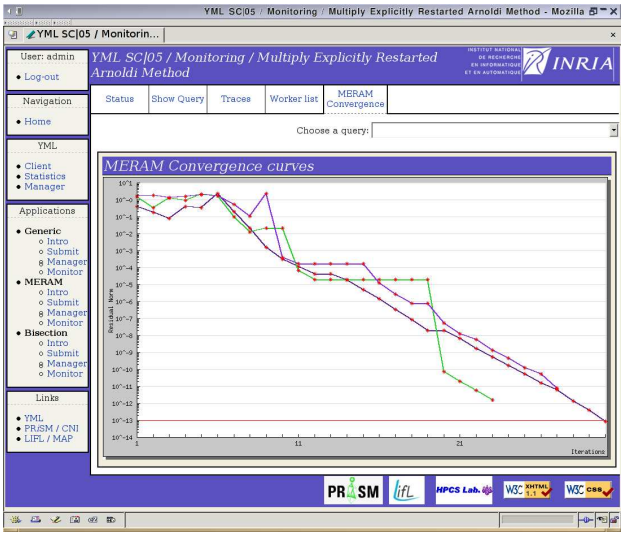

Figure 4: MERAM Convergence: an application specific view example

CONCLUSION

We have described a web portal oriented towards linear algebra applications. It rests on the YML framework, a workflow engine for GRID computing. The web portal provides facilities for the study of numerical methods hiding the complexity of GRID middleware on the one hand and of the applications on the other hand. Motivations for creating web portals are to ease experimentation, inter research group collaboration as well as scientific communication and diffusion.

The web portal behaves as a standard client for the YML framework with query submission, management and monitoring facilities as well as application specific interfaces. Each registered application enriched the web interface with new guided submission wizard, query generators, queries management and monitoring. The monitoring aspect is one

of the most important ones, each application provides several views of all running or executed queries. An application benefits from standard views and specific ones can be added to the web portal.

An ongoing work increases the number of generic and application specific views and develops integrated on-line tools for live collaboration between distant research groups. We also put effort in designing intelligent mechanism to guide the user in the parametrization of its numerical methods. Expert knowledge should be dynamically integrated in application submission wizard using a decision making solution. For the time being an important effort is needed in order to determine a coherent and generic enough expert knowledge storage format.

1 References

[1] Lee D., Dongarra J., Ramakrishna R., VisPerf: Monitoring Tool for Grid Computing *Lecture Notes in Computer Science*, Springer Verlag, Heidelberg, Volume 2659, pp. 233-243, 2003.

[2] Delannoy O. and Petiton S. A Peer to Peer Computing framework: Design and Performance Evaluation of YML. *Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks* July 2004; ISPDC/HeteroPar post conference proceedings by IEEE Computer Society Press, Ireland.

[3] Emad N., Petiton S. and Edjlali G. Multiple explicitly restarted Arnoldi method for solving large eigenproblems. *SIAM Journal on scientific computing SJSC* 2005; **27** (1): 253-277.

[4] Noulard E., Emad N. A key for reusable parallel linear algebra software. *Parallel Computing Journal, Elsevier Science B.V.* 2001; **27(10)**(10-4): 1299-1319.

[5] Agrawal S., Arnold D., Blackford S., Dongarra J., Miller M., Sagi K., Shi Z., Seymour K., Vahdiyar S., Users' Guide to NetSolve v1.4.1, *ICL Technical Report*, ICL-UT-02-05, June 25, 2002.

[6] Sato M., Hirano M., Tanaka Y., Sekiguchi S., OmniRPC: A Grid RPC Facility for Cluster and Global Computing in OpenMP.*Proc. of WOMPAT 2001*, pp. 130-136, 2001.

[7] D. Erwin, ed., UNICORE plus final report – uniform interface to computing resources, *Forschungszentrum Jlich*, 2003, ISBN 3-00-011592-7

[8] Globus Project *http://www.globus.org*

[9] Condor Project, *http://www.cs.wisc.edu/condor/*

[10] David P. Anderson BOINC: A System for Public-Resource Computing and Storage, *5th IEEE/ACM International Workshop on Grid Computing*, November 2004, Pittsburgh, USA

[11] Capello F., Djilali S., Fedack G., Herault T., Magninette F., Neri V. Lodygensky O., Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid, To appear in FGCS Future Generation Computer Science, 2004.

[12] SAAD Y., Numerical Methods for Large Eigenvalue Problems, *Manchester University Press*, Manchester, UK, 1992

[13] *http://lavezzi.rech-info.yser.net/sc05/*