Scientific workflow orchestration interoperating HTC and HPC Resources

Luis Cabellos, Isabel Campos, Enol Fernández-del-Castillo

{*cabellos, iscampos, enolfc*}*@ifca.unican.es*
Department of Advanced Computation and e-Science
Institute of Physics of Cantabria, CSIC
39005, Santander, Spain

Michal Owsiak, Bartek Palak, Marcin Płóciennik

{*michalo, bartek, marcinp*}*@man.poznan.pl*
Poznan Supercomputing and Networking Center
Institute of Bioorganic Chemistry PAS 61-704 Poznań, Poland

Abstract

In this work we describe our developments towards the provision of a unified access method to different types of computing infrastructures at the interoperation level. For that, we have developed a middleware suite which bridges not interoperable middleware stacks used for building distributed computing infrastructues, UNICORE and gLite. Our solution allows to transparently access and operate on HPC and HTC resources from a single interface. Using Kepler as workflow manager, we provide users with the needed integration of codes to create scientific workflows accessing both types of infrastructures.

Keywords: Scientific Workflows, Grid, High Performance Computing, High Throughput Computing

1. Introduction

Research infrastructures dedicated to computing have grown and diversified in the scientific world substantially over the last decade. The ecosystem of computing resources ranges from local resources at departmental and research center level, either in the form of workstations or computer clusters, up to national and international facilities run by Supercomputer Centers, or High Performance Computers (HPC).

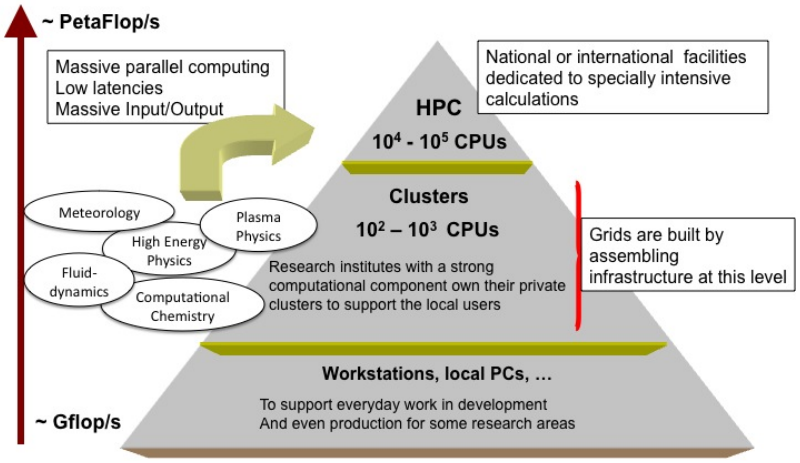

Figure 1: The structure of available computing infrastructures described as a pyramid with high-demanding computing facilities at the cusp

From the point of view of size and utilization of resources, they can be described with a pyramidal organization (see Figure 1). The bottom of the pyramid forms workstations and PCs that mainly support the everyday work of the user. Clusters (the middle layer) owned by particular institutions are usually deployed to improve performance and/or availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability [1]. The most powerful but also the most expensive - HPC machines dedicated for performing massive computation - resides on the top of the structure.

A global trend, which could be seen over the last years, is to combine existing resources to ensure on the one hand their effective usage, on the other hand their permanent availibility. Good example can be grid infrastructures - constructed by assembling together the clusters of a number of research centers interested in data and/or resource sharing for a particular research project(s)[2].

Scientific communities have nowadays the possibility of accessing a large variety of computing resources. HPC systems, Grid-infrastrutcures and clusters are also from the conceptual, architectural and policies point of view very different infrastructures, with its own administration and accessing mechanism. The situation has evolved very heterogeneously, in such a way that it

is frequently the case that scientist do not have a problem with lack of resources, in terms of the total amount of CPU and storage resources at their disposal, but rather with utilization of these resources in an effective way.

A typical use case consists of a sequence of related applications processing data and transferring results from one step to another, constituting so called workflow. The flow of input/output between the different parts of the workflow job(s) need to be automatized. This automatization is in no way trivial when different computing infrastructures are being used at the different stages. It is often the case that parts of the workflow need to be run, or can be run in smaller facilities like local clusters, or distributed over HTC resources, while other parts of the workflow need to perform an intensive simulation involving many processes in a HPC system.

In this work, we address the joint usage of different computing infrastructures. For that, we have developed a middleware suite which allows to transparently access resources based on different middlewares, in a secure and controlled way, to construct workflows that facilitate the work of researchers. The usage of this middleware allows job submission and data access in mixed infrastructures using the UNICORE gate [4] or based on gLite [9] and derivated middlewares. In our previous example a job would be submitted to the Supercomputer, and after completion, output data would be copied onto Grid storage elements or cluster facilities, for analysis, in a complete automatized way in a different set of resources. Furthermore the well-stablished data management tools on Grid infrastructures can be used by all the members of the collaboration if necessary to perform a distributed analysis.

We present a solution that provides the needed integration of the codes using a workflow engine that is able to access the different available computing infrastructures. Our solution leverages and integrates various already existing middleware components: the Kepler [5] workflow engine provides the framework for the integration of codes in a single and traceable workflow that access the infrastructure using the Roaming Access Server (RAS) [6] extended with the use of the Vine Toolkit [7]. RAS provides access to the underlying infrastructures using gLite-based and UNICORE middlewares. Interactive access to the resources is achieved using the i2glogin [8] application. All these tools and their interdependencies are detailed later in this article.

The article organizations goes as follows. We first analyze the users requirements and state of the art, afterwards we describe our methodology

to create an interoperable access layer for computing resources, and finally we apply the methodology and software developed to two examples in two different research areas: Lattice QCD and Nuclear Fusion.

2. Requirements

Scientists face complex problems that require using a number and variety of analysis tools, each of them designed for specific computing platforms, that consume data from diverse sources, which may need conversions. This set of heterogeneous scientific computations and data sources can be integrated and coupled into a *scientific workflow*, providing researchers a traceable and reproducible tool for solving their scientific challenges.

Researchers need a workflow management tool that facilitates this integration and provides a framework for the execution of the actions needed to solve their problems. This tool should also permit performing additional tasks such as computational steering and interactive monitoring or control, that are especially useful in the early stages of the workflow design, while at the same time the tool should allow long-living workflows to run unattended for weeks or even months.

The workflow management tool should provide seamless access to the distributed resources and services used for the execution of the applications and storage of data. Ideally, the different types of resources should be accessed in a similar way from the tool and new resource types should be easily added without major changes in the tools or the workflow definition process. The access to the resources must be accomplished using the existing middleware implementations that are already deployed in several computing infrastructures, namely gLite and UNICORE.

3. State of the art

Over the last few years, several Grid middleware initiatives have contributed to the deployment and support distributed computing infrastructures. Midleware solutions Globus-based (like gLite [9], ARC [10] and i2g [11]) and UNICORE-based provide the basis for the construction of environments that researchers can use for their computational needs.

However, the lack of standards or their implementations and the limited support of each of the middlewares to very specific platforms and architectures have reduced the possibility of seamless interoperation between the

different existing environments. Interaction between the user and target infrastructures is carried out using the different client tools provided by each middleware. These tools are not interoperable between them and range from Command Line Interfaces (CLIs) to rich graphical applications that hide the underlying complexity of the systems (see Figure 2).

Most of the Grid resources available in Europe use the gLite middleware or one of its extensions. gLite is a middleware developed and deployed by the EGEE [1] project and it is focused on providing a HTC infrastructure for the support of distributed processing of very large data volumes with sequential batch jobs, common in High Energy Physics analysis. Initiatives like i2g, have extented the functionality of gLite with the support for parallel and interactive execution on the resources.

Although access to HPC systems is typically accomplished using direct ssh login, several HPC centers in Europe use UNICORE for exposing their resources to external users. UNICORE is a middleware that implements some of the available standards for Grid systems in order to provide secure and seamless access to the HPC resources.

Standard protocols of Grid middleware, which define the content and sequence of message exchanges used to request remote operations, have emerged as an important and essential means of achieving the interoperability upon which Grid systems depend. However, the support for such standards is still limited in some of the current middleware implementations. Moreover, the existing standards for accessing computing resources (like BES [14] and JSDL [15]) cover basic functionality and do not provide direct support for workflow execution.

Coupling different computational modules and applications requires coordination, structures data management and resource scheduling to be performed efficiently. It is precisely at this level where the problems of lack on interoperability among different infrastructures becomes more evident.

We have analyzed the state of the art concerning existing workflow management systems in order to choose one that fits the mixed environment needs best. After preliminary analysis of available workflow systems, three of them were choosen for further research: Salome [12], Cactus [13], and Kepler [5]. Each of them has advantages and dissadvantages that have impact on application's requirements.

[1] See http://www.eu-egee.eu

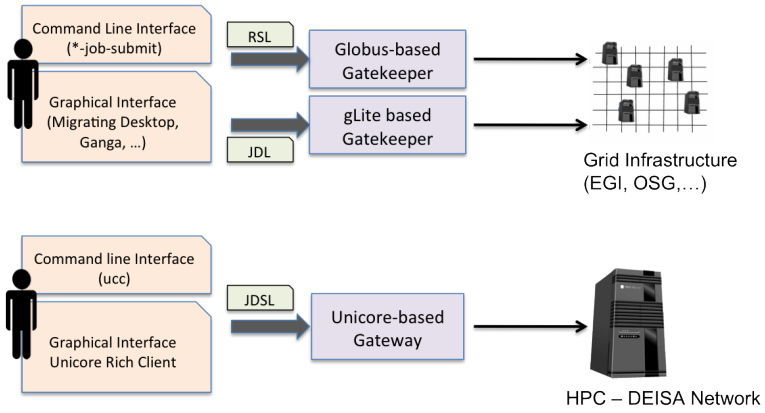

Figure 2: Job submission to different infrastructures

Cactus is an open source, modular, portable, programming environment for collaborative HPC computing. It allows to develop code in F77, F90, C, C++ and runs on a wide range of architectures and operating systems. It is quite efficient in terms of performance, however it was ruled out due to limited capabilities in terms of workflow building.

Salome is open source software as well. It provides strong basis for numerical computations and provides a generic user-friendly interface. It allows to integrate new components on heterogenous systems and pass computation into external solvers using CORBA. However, CORBA is not widely used in some of the scientific communities, limiting the possible adoption of Salome as a generic tool. Additionaly, there is no support for accessing any grid middleware in the current version.

Kepler is a Java based workflow system. It allows user to build and execute scientific workflows. It provides users with efficient and user-friendly interface, is easy to learn and allows to build complicated workflows that include complex data flow structures. Thank's to Java based development it can be executed virtually anywhere - as long as a Java Virtual Machine is available for the given architecture and/or system. Another benefit of the Kepler engine is that it can execute a previously defined workflow in batch mode, allowing to perform long-term executions without user interaction.

After evaluation and analysis of the workflow management systems Kepler was choosen as the platform for workflow management to work on for developing our interoperability developments.

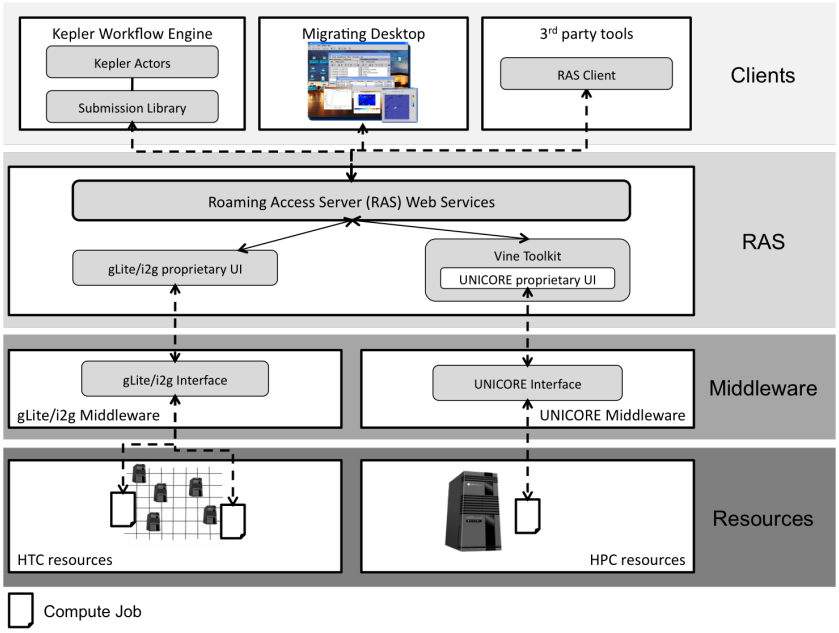

Figure 3: Overall architecture of RAS/Vine/Kepler

4. Proposed Framework

The objective of our framework is to provide researchers with a user friendly environment that allows the definition and execution of scientific workflows where the different tasks must be able to access different resources (both HTC and HPC) for application execution and data access without exposing details of the middleware that drive those resources.

Figure 3 shows the overall architecture of our solution. It consists of a client-server framework with clear separation of the different middleware layers that allows hiding the various infrastructure behind a common and single API. Users are presented with a Graphical User Interface (GUI) composed by the Migrating Desktop [16] and the Kepler workflow engine. Both use the Roaming Access Server (RAS) as a common access layer to the underlying resources and their different midleware implementations. All these components will be described in the following subsections.

4.1. A Unified Access Layer for Resources

The RAS intermediates between the client desktop applications and the computing infrastructure with a set of well defined web services for the management of jobs and files in computing resources. RAS has a modular architecture where the access to different infrastructures is provided by specific middleware plugins that implement the low level details of each type of resources.

Access to gLite based resources is provided by two different middleware plugins: *gLite* and *i2g*. The gLite plugin uses the gLite User Interface (UI) commands to submit sequential and parametric jobs to resources based on that middleware. The i2g plugin provides advanced capabilities on top of gLite based resources. This plugin makes use of the specific int.eu.grid command line tools which access the CrossBroker [17] metascheduler for execution of parallel jobs and fast startup of prioritized applications. Furthermore, the i2g plugin enables interactive access to Grid jobs during their execution by integrating i2glogin into the architecture. i2glogin is a tool that is capable of creating communication channels between the application and the user. Using i2glogin only requires outbound connectivity in the host where the application runs. i2glogin uses the globus security features to create a secure bi-directional data exchange channel during runtime without any modification to the user application.

In order to provide access to HPC resources, we have developed a new *Vine Toolikit* plugin. Vine Toolkit is a Java framework based on several Open Grid Forum (OGF) [2] standards, and originally designed to work within Web Applications containers. The production version provides support for UNICORE 6 and Globus GT4 middlewares using a proxy based security model. This allows the RAS to support the middleware implementations without changing the security infrastructure of the system. The developed plugin exposes the Vine Toolkit API through the already existing RAS Web Services API. RAS clients in the upper layers are able to access the new types of resources without major changes in their code, freeing the developers of this clients from encapsulating the JSP/Servlet container that Vine Toolkit requires or changing the security model that the UNICORE client requires.

All the middleware plugins can access common RAS features like central repository for job management, up-to-date job status, user management,

[2]See http://www.ogf.org/

plugin management, VOMS [19] proxy management and access to LFC [20] file catalogues. These common utilities and the uniform and well defined services provides clients with a unified layer for submitting both serial and parallel jobs into different types of resources.

RAS is developed in Java and uses Apache Tomcat [3] as Web Application Server. However, the use of the gLite and int.eu.grid plugins require the clients of those middlewares to be installed in the machine hosting the service. Currently only Scientific Linux 4 or 5 versions of the gLite middleware are available. A single RAS installation can handle several users from different VOs, although the typical deployment uses one RAS per VO.

4.2. Client Tools

Users have different client tools for interacting the reosources. The Kepler engine allows them to define and run their workflows, while Migrating Desktop provides an advanced environment that hides the Grid middleware complexity and makes the access to the resources easy and transparent.

In the Kepler workflow engine each action is represented by an *actor*. The development of new actors or composite actors (blocks that can represent whole workflows and having ability to be incorporated into another workflow as regular actor) can be accomplished using Kepler extension mechanisms and its Kepler Archive files (KAR). We have developed a set of actors and composite actors that allow Kepler to access distributed infrastructures using the RAS. These actors and a helper library with common functionality are bundled together in a package that can be easily added to an existing Kepler installation.

The developed actors provide the needed functionality for dealing with the distributed infrastructure. Using those actors and a set of predefined templates common cases, users can create complex workflows that access HTC and HPC resources in various execution modes. The actors provided by our package can be classified in the following areas:

Security Actors for generating proxies with or without VOMS extensions that will be later used by the other actors to enable secure access to resources

Job Generation These actors create the appropriate description in the JSDL or JDL language for submitting the jobs to the resources from

[3] See http://tomcat.apache.org/

a set of basic parameters from the user (e.g. executable, arguments, input and output files). Advanced users with deep knowledge of the submission languages may not use the job generation actors and create directly the descriptions.

Job Submission The job submission actors take a job description and a user proxy and contact the RAS for the submission of the job to a remote resource and return a unique identifier that can be used later in other actors. Various execution modes (serial, parallel and parametric) are supported by the job submission mechanisms.

Job Monitoring Taking a previously submitted job identifier, these actors check the status of the job or wait until a particular state has been reached (e.g. job is done or is aborted). They also provide the possibility of resubmit the jobs if any failure is detected.

Job Output Retrieval Actors for retrieving the output and error files of jobs previously submitted.

Remote File Access These are a set of actors that allows user to manage files in remote locations using file catalogues (LFC).

Besides the Kepler engine, users may access RAS functionality with the Migrating Desktop. Oriented to job based functionality, it complements the workflow oriented Kepler with a powerful and flexible user interface that gives easy access to resources and network file systems independently of the system version and hardware. It allows the user to run applications and tools, manage data files, and store personal settings independently of the location or the terminal type. The idea of the Migrating Desktop usage is to have the platform to start remotely and monitor different predefined Kepler workflow scenarios.

Both Kepler and Migrating Desktop are Java applications that can be executed in almost every system with a Java Virtual Machine. The definition and control of the workflows is performed at the user's desktop, while the execution and file management is handled by the RAS on the distributed resources on behalf of the user.

5. Workflow Scenarios

This section presents a generic scenario for mixed workflows showing the possibilities of our proposed solution by using both HTC and HPC resources.

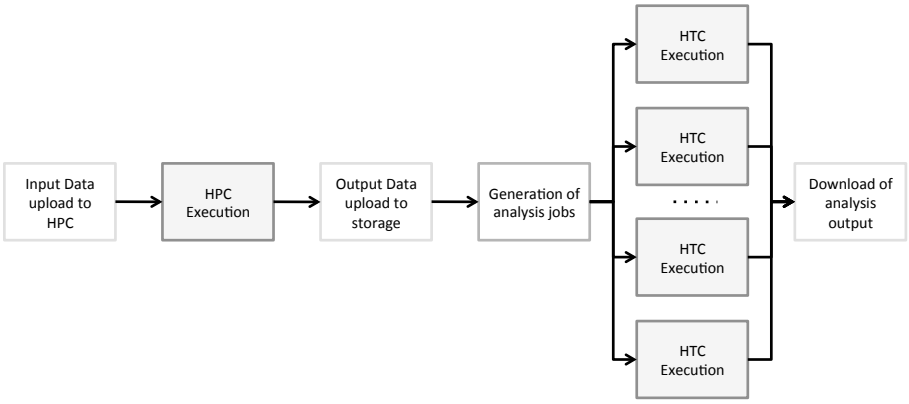

Figure 4: Generic Mixed Workflow

The application use cases described in the next section take this scenario as basis for building the workflows used by the researchers.

A use case common to several scientific areas involves the analysis of big amounts of data that is generated by a preceding simulation. Problems related to complex fluid dynamics, meteorology, plasma physics, quantum chromodynamics fall clearly into this category. While the generation phase usually requires large computing power only found in HPC systems, the analysis phase can be performed by independent tasks that handle different chunks of data in HTC resources.

Figure 4 shows the main actions in this mixed workflow scenario. The first step consist in the upload of any required input data to the HPC system that will be used for the simulation. Once this data is available, the simulation application can be run according to the user defined requirements (e.g. number of cores to use, estimated memory usage or estimated duration of the execution). The execution can last for several days or weeks and produces the data to be analyzed at later stages of the workflow.

Once data is produced, it is transfered to a storage system accessible to all resources involved in the execution. This data transfer may be performed concurrently to the execution of the simulation thus allowing to start following steps of the workflow as soon as partial simulation results are available. The amount of data and the characteristics of the analysis application determines the number of analysis jobs that will be executed. An intermediate step between the data transfer and the actual execution of analysis contains

the logic that creates the appropriate description for the jobs: either as a set of individial jobs or as single parametric job that will spawn into different subjobs. The details of the execution of these jobs are described in the next section. As the HTC jobs generate the final output data, it is fetched and stored in a accessible place by the last action of the workflow.

5.1. Execution in HTC resources

The analysis phase of the workflow is normally executed as a set of independent HTC jobs that run simultaneously using the available resources. Each of these HTC jobs may be a stand-alone application or a complex task composed of several interdependent steps described as a subworkflow. Stand-alone applications range from sequential codes which use a single CPU to parallel applications using MPI than can spawn over multiple sites.

Subworkflows are useful for complex analysis that require the execution of different codes that depend one on each other. Researchers define the subworkflow in their desktop by using the Kepler engine GUI and state the number of CPUs required for its execution. This subworkflow is submitted to the resources as a single job requesting the number of CPUs requested by the user.

In order to manage the group of computing nodes available for the job, the subworkflow uses the MPI-Start [18] tool. MPI-Start is a middleware layer for starting and parallel jobs in a uniform way in heterogenous resources. It detects the allocated machines for the job and allows the Kepler engine to use those machines for the execution of the different tasks that comprise the subworklow. Figure 5 shows the execution flow for the subworkflows. The job is submitted from the RAS and it is executed on a set of worker nodes (from WN0 to WNn). At the master node (WN0), MPI-Start constructs a list of machines, depicted by arrow (1), and copies any input files from this WN0 to the rest of the nodes using the most appropriate method available. An special version of Kepler without GUI (Kepler GL) is started by MPI-Start as shown by arrow (2). This Kepler engine runs the subworkflow defined by the researcher making use of the available machines for the execution of different analysis applications as depicted by arrows (3).

Both types (stand-alone and subworkflows) of jobs running in gLite infrastructure can use the interactive features of RAS and i2glogin, allowing the researcher to steer the behavior of the application while its running. This might be useful for adjusting analysis parameters to reach a desired solution. The i2glogin tool creates an interactive channel that forwards I/O from the

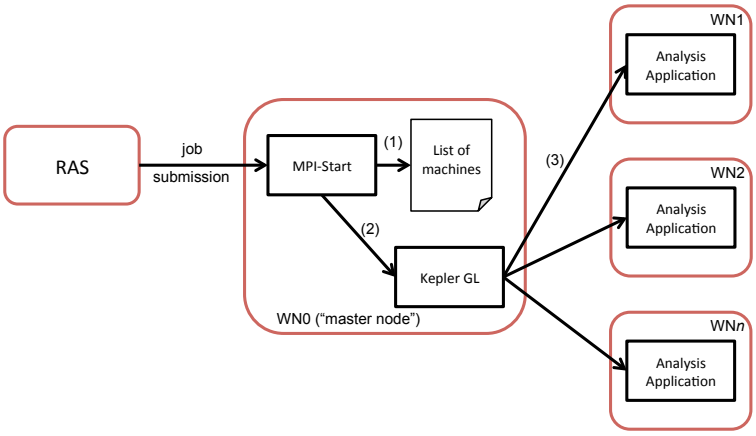

Figure 5: Execution of subworkflows in remote resources.

user desktop to the hosts where the application is executed and viceversa. i2glogin can be used with legacy appications that read and write their tuning options by the standard input and output streams or can be integrated in the application logic in order to gain much better control of the execution.

6. Application Use Cases

In this section we will describe two use cases applied to two different research areas, Lattice QCD and Nuclear Fusion.

6.1. Numerical Simulations in Lattice QCD

QCD is the theory describing the interactions between elementary particles, quarks and gluon fields. The implementation of QCD on a computer goes by defining a space-time lattice, with side L, and spacing a, and performing Monte Carlo sampling over the space of configurations defined by the model path integral. Physical quantities are obtained by extrapolating the results measured in the simulation to the limit $a \rightarrow 0$

In Lattice QCD quantitative results are obtained by performing numerical simulations. The implementation of QCD on the Lattice is the only well defined implementation of QCD in a mathematical sense, beyond perturbation theory. In this sense it would produce exact results, just like if we were able to do the calculations by hand, provided infinite computer time could be obtained.

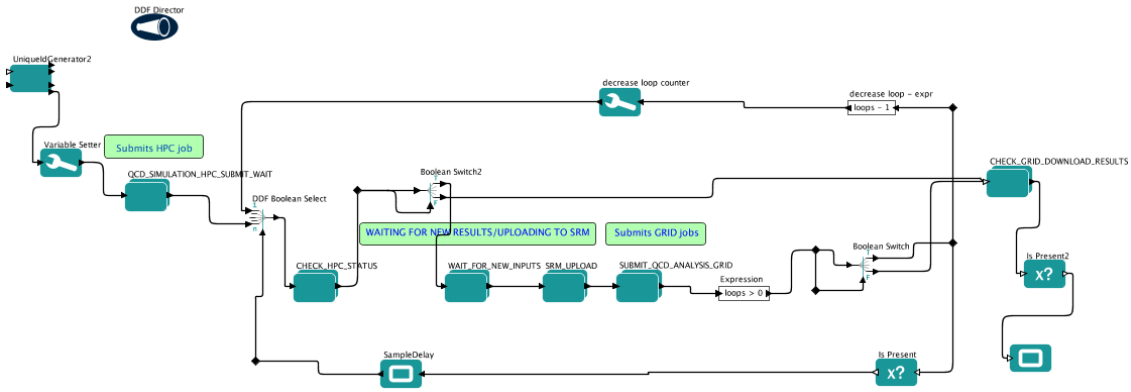

Figure 6: Kepler workflow construction for the Lattice QCD use case

Numerical investigations in Lattice QCD have traditionally been, and still are, very expensive computationally speaking, even for the scale of modern supercomputers. The situation is specially complicated if we want to include the effects of the sea quarks, or light quarks, which on the other hand is the most interesting case to simulate, since it means simulating the real world.

Experience suggests that for the example system of QCD with two flavours the lattice spacing should not larger than 0.1 fermi and the mass of the pion should be between 200MeV and 500Mev at most. In order to keep systematic errors under control this constraint implies to simulate physical lattices larger than 3fm [21]

As an example, simulating a lattice with a = 0.05, L=3,2 fm and a Pion mass of Mpion = 200 MeV, implies to simulate a lattice of size L=64 in lattice spacing units, which requires a sustained computing power of about 10 sustained Teraflop/s per year. This means that the simulation of Lattice QCD with light quarks is reserved to Supercomputers in a classical definition.

However, when it comes to storing the results, the necessities of such simulations need to be carefully analyzed as well. Each single gauge field configuration would require 20 GB, and the quark propagator would need about 70 GB. One typically needs hundreds of these configurations, at different values of L, and also different values of the mass of the Pion, lattice spacing a, etc... therefore ending-up with several Terabytes to handle is a common situation in Lattice simulations.

The obtention of results with phenomenological implications is one of

main research lines in Lattice Gauge Theory. In this context the community has gathered in collaborations in order to asemble enough human and computational resources to tackle lattice phenomenology. Indeed, these investigations require being able to simulate with enough statistics for those values of the parameters with phenomenological interest, which also happen to be to more expensive numerically. Data analysis is therefore a collaborative work, in which the members of the collaboration need to have access to the output to analyze different properties of the gauge field configurations.

In the use case we present in figure 6, the main simulation producing the configurations is performed in a Supercomputer. The system periodically writes the output to a storage element, Grid-enabled, where all the members of the collaboration have been granted access-rights. Data analysis codes, usually serial codes, are executed using the HTC infrastructure.

We would like to remark that for those Lattice QCD problems which can be solved using serial runs, a gLite based infrastrcuture offer by itself a good solution, as has been shown recently [22].

6.2. Plasma Physics applied to Nuclear Fusion

The design of ITER[4] requires from the Fusion simulation community substantial effort in physics modelling and simulation, covering both a wide range of timescales and spatial ordering which are generally very demanding from a computational point of view.

The complete simulation of a fusion device requires the coupling of several codes in a complex workflow. In general one considers a selection of codes covering both edge and core physics in the fusion reactor. The requirements of the components of the workflow are in general different, some codes, special those related to the computation of the plasma evolution need to run on HPC systems, others, more related with the evaluation of the properties of the plasma are well suited for serial simulations in distributed HTC resources.

The provision of an integrated environment for fusion modelling on distributed and HPC infrastructures using a common and user friendly interface is a very challenging problem which is receiving increasing attention (see related work in [23]).

In our example we consider the codes ASTRA[24] and TRUBA[25] to show how is possible to couple, in a dynamical way, different codes calculating

[4]See http://www.iter.org

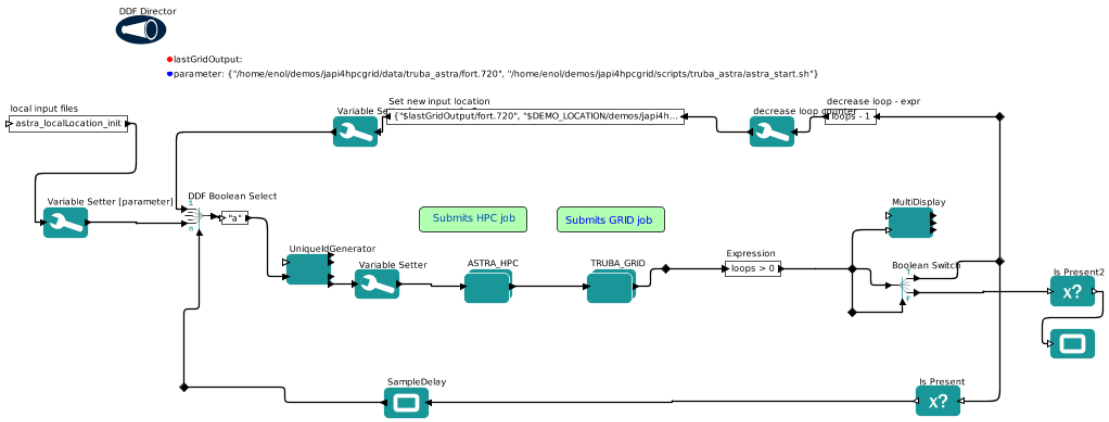

Figure 7: Kepler workflow construction for the Plasma Physics use case

different properties of fusion plasmas..

The code ASTRA is a parallel code which computes the evolution of plasma configurations, while TRUBA computes the heating properties of those plasma configurations launching of many serial jobs to analyze a particular plasma configuration. In our example the input/output data for both codes are stored in a Grid storage element, accessible to both codes

The use case has been depicted in Figure 7. The Kepler workflow manager is installed in the user desktop together with the user certificate. The user submits an ASTRA job to the HPC system, where the evolution of the plasma is being calculated, and the output configurations copied on to a Grid storage element.

In the mean time the code TRUBA has entered into the workflow engine as well. TRUBA jobs are submitted via RAS to the gLite infrastructure. Those jobs read the plasma configuration produced in the HPC system by ASTRA, and calculate some the heating properties of the plasma configuration by performing a Ray Tracing which translates into the submission of many serial jobs on the gLite infrastructure.

7. Conclusions

The proposed framework provides researchers a workflow based environment where multiple tasks can be run in different resources (both HTC and HPC) from a user friendly interface without changes in the user habits. The

workflows that the researchers may execute range from the simplest one that requires the use of a single CPU at a resource to the most demanding ones which will need all the available platforms in order to complete the calculations in a reasonable amount of time.

These developments can be considered as a the implementation of a bridge between different middleware stacks, namely Globus and UNICORE, to provide users with a certain unfied view of the infrastructures at the interoperation level. One of the main difficulties in doing it has been coping with the different development state of Globus and UNICORE, specially concerning implementation of standards. For this a number of middleware plugins, based on standards, have been developed for the RAS, providing support to scientific workflows via the Kepler workflow manager.

These plugins use common features to provide a unified interface to the upper layers of the middleware stacks: a central repository for job management, up-to-date job status, user management complying with VOMS and access to Grid storages using different protocols (ftp/sftp/gridFTP/BFT/...).

A particularly challenging aspect has been the handling of the sometimes complex errors coming from the infrastructure where the applications are being run. The error messages of the middleware stacks are often too generic, not giving enough information about the source of the problem. This has required a very complex debug of the error analysis.

We have applied this solution to two challenging areas: Lattice QCD and Plasma Physics, where the methodology has been applied to realistic use cases with success.

Our developments are implemented in the context of the European Grid Infrastructure [5] and the HPC network initiative DEISA [6]. In this context we are working hand-to-hand with the middleware developers in order to keep this solution up-to-date with the development of the underlying middleware stacks.

Acknowledgments

The authors thank the European Commission funding via Grant Contract RI-211804 (*EU For ITER Applications*) where the use case presented for Plasma Physics has been analyzed and Ministry of Science and Innovation of

[5] See http://www.egi.eu

[6] See http://www.deisa.eu

Spain funding via FPA2008-01732 (*Numerical simulations on Lattice QCD Phenomenology*) where the use cases for Lattice QCD have been tested and worked out.

[1] David A. Bader and Robert Pennington, Cluster Computing: Applications. The International Journal of High Performance Computing, 15(2), 2001, pp 181-185.

[2] I. Foster and C. Kesselman, The Grid: Blueprint for a new computing infrastructure, Ed. Springer, 2004

[3] I. Foster, Globus Toolkit Version 4: Software for Service-Oriented Systems. International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[4] Romberg, M.: The UNICORE Architecture: Seamless Access to Distributed Resources. In Proceedings of the 8th IEEE International Symposium on High Performace Distributed Computing (HPDC), 1999, pp. 287–293.

[5] McPhillips, T. —Bowers, S.—Zinn, D.—Ludaescher, B.: Scientific workflow design for mere mortals. Future Generation Computer Systems, Vol 25, 2008, pp. 541–551.

[6] M. Kupczyk, R. Lichwaa, N. Meyer, B. Palak, M. Pciennik, M. Stroiski, P. Wolniewicz, The Migrating Desktop as a GUI Framework for the "Applications on Demand" Concept M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 91-98, 2004. Springer-Verlag Berlin Heidelberg 2004

[7] Russel, M. —Dziubecki, P. —Wolniewicz, G. et al.: The Vine Toolkit: A Java Framework for Developing Grid Applications. In Proceedings of 7th Int. Conf. on Parallel Processing and Applied Mathematics (PPAM), Gdansk, September 2007, pp. 331–340.

[8] Rosmanith, H. —Kranzlmüller, D.: glogin - A Multifunctional, Interactive Tunnel into the Grid. In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburg, November 2004, pp. 266–272.

[9] Laure, E et al.: Programming the Grid with gLite. Computational Methods in Science and Technology, Vol. 12, 2006, No. 1, pp. 33–45.

[10] Ellert, M et al.: Advanced Resource Connector middleware for lightweight computational Grids. Future Generation Computer Systems, Vol 23, 2007, No. 2, pp. 219–240.

[11] J. Marco et al. The Interactive European Grid: Project objectives and achievements, Computing and Informatics, Vol. 27, p161-173 (2008)

[12] Ribes, A. and Caremoli, C.: Salome platform component model for numerical simulation. In Proceedings of the 31st Computer Software and Applications Conference (COMPSAC), July 2007, vol.2, pp. 553–564.

[13] Cactus computational toolkit. Available on: `http://www.cactuscode.org/`.

[14] Basic Execution Service (BES) Specification, Version 1.0. November 2008. For a definition of the standard see http://www.ogf.org/documents/GFD.108.pdf

[15] Global Grid Forum: Job Submission Description Language (JDSL) Specification, Version 1.0. November 2005.

[16] M.Owsiak, B.Palak, M.P?ciennik , Graphical Framework for Grid Interactive and Parallel Applications, Computing and Informatics, Volume 27, pp 223-232 (2008)

[17] E. Fernandez et al. Crossbroker: A Grid Metascheduler for Interactive and Parallel Jobs Computing and Informatics, Vol. 27, pp 187-199 (2008)

[18] K. Dichev et al. MPI Support on the Grid, Computing and Informatics, Vol. 27, pp 213-223 (2008)

[19] Alfieri, R; Cecchini, R; Ciaschini, V, et al., From gridmap-file to VOMS: managing authorization in a Grid environment , Future Generation Computer Systems Volume: 21, Issue: 4, pp 549-558 (2005)

[20] Baud, JP et al. Performance analysis of a file catalog for the LHC computing Grid 14th IEEE International Symposium on High Performance Distributed Computing Proceedings Pages: 91-99 Published: 2005

[21] L. Del Debbio, L. Giusti, M. Luescher, R. Petronzio and N. Tantalo "QCD with light Wilson quarks on fine lattices (I): first experiences and physics results" JHEP 0702 (2007) 082

[22] J.T. Moscicki et al., Lattice QCD thermodynamics on the Grid, Computer Physics Communications (2010), doi:10.1016/j.cpc.2010.06.027

[23] J. Cummings et al. "EFFIS: An End-to-end Framework for Fusion Integrated Simulation," pdp, pp.428-434, 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010

[24] G.V. Pereverzev, P.N. Yushmanov. "ASTRA-Automated System for TRansport Analysis". Max-Plack Institut für Plasmaphysik, IPP-Report, IPP 5/98 (2002)

[25] M. Tereshchenko et al., 30th EPS Conf., ECA 27 A (2003)