Workflow issues in atomistic simulations

Clovis Chapman[1], Jon Wakelin[2], Emilio Artacho[2], Martin T Dove[2], Mark Calleja, Richard Bruin[2] and Wolfgang Emmerich[1]

[1] Dept. of Computer Science, University College London,
Gower St, London WC1E 6BT, United Kingdom
[2] Dept. of Earth Sciences, Downing Street, Cambridge CB2 3EQ, UK

Abstract

The following article describes the techniques and mechanisms that we have used to tackle workflow problems encountered in the e-minerals project. We examine how established tools and technologies can be brought together to specify and deploy a *computational process*, consisting of a set of jobs and tasks, on our production level mini-grid infrastructure, with respect to a specific problem - the distribution of calculations required to determine, in a systematic way, the mechanisms by which pollutant molecules such as DDT, dioxins and biphenyls, become bound to soil minerals. We also briefly discuss the use of data standards such as CML and Web-Service based grid standards as a means to facilitate workflow specification.

Keywords

Workflow, Grid, combinatorial problems, CML

1. Introduction

One of the aims of the e-minerals project [1] is the systematic study of adsorption characteristics of pollutant molecules on soil mineral surfaces. The problem is very complex from different points of view.

(i) In order to establish significant trends and possibly correlations between adsorption energies and phenomenological results, the study of a large enough set of systems is required. The targeted family of organic pollutants of interest has over two hundred different molecules, which, together with the several relevant soil minerals, their likely surfaces, possible edges, steps etc., make a huge combinatorial problem with lots of (fairly independent) calculations. We are particularly interested in the chlorinated dibenzodioxins, dibenzofurans, biphenyl, DDT and PCB families. Some of them of industrial origin, others coming from agricultural practices, these compounds are particularly persistent in the environment, and represent a pressing problem even if some of them are no longer used.

(ii) Each separate case requires many calculations (defining by ``calculation" a geometry relaxation to a close local energy minimum), in as much as many possible locations and orientations are a priori possible for a given molecule on a given surface.

(iii) Different situations will require different levels of theory (from empirical force fields to ab initio calculations).

These characteristics demand special computational treatments beyond what is customary nowadays for condensed matter simulations. It requires efficient automation of the calculations, including inter-code operability, in addition to making simultaneous use of large amounts of computational resources.

These type of workflow problems are inherent to grid computing. Most scientific problems cannot be solved using solely one application but usually require the definition of a *computational process* or *workflow process,* consisting of a set of tasks or jobs, which must be executed as a whole for significant results to be obtained. However the grid is a relatively complex environment in which to deploy such computational processes. It consists of a number of resources independently managed, each with its own scheduling and operating system, and each offering very different execution environments: one must be able to coordinate the remote submission of jobs across these heterogeneous and autonomous resources, transfer any required input and output data, monitor the job executions, handle potential faults, and most importantly be capable of responding to the *dynamic*

nature of the grid - where the availability of specific resources might vary greatly from one moment to the next.

From a user perspective the problem is relatively simple to define: how can one specify a computational process and simply 'throw' as many resources at it as possible in order for it to be completed in a timely fashion and with minimum intervention? But from a computing perspective, no single tool or solution exists to tackle the workflow problems that we have encountered so far in the e-minerals project.

We will cover in this article the ways in which we have brought together existing tools and technologies to allow users to specify computational processes and deploy them on our e-minerals mini-grid infrastructure – as well as non-dedicated e-science resources such as HPCx [8]-, with respect to the scientific problems that we aim to solve. We will also discuss how the use of standards such as the Computational Mark-up Language (CML) will facilitate these processes, and how the adoption of Web Services by the Grid community may provide us in the near future with more flexibility in specifying these processes.

2. A Real World Problem: Pollutants in Soils

The workflow problem that we introduce here as our reference use case is the management is the management and distribution of calculations required to determine, in a systematic way, the mechanisms by which pollutant molecules such as DDT, dioxins and biphenyls, become bound to soil minerals.

Though the scientific details are outside the scope of this article (these are described more extensively in [2]), this has all the traits of a complex simulation scheme: there are large numbers of relevant molecules, minerals and mineral surfaces that need to be taken into account, and for each combination we need to systematically investigate the potential energy surface of the system, which in itself requires a large number of calculations.

The computational process is illustrated in figure 1. This process, consisting of SIESTA jobs, calculations dealing with energy minimisations, is a particularly good example of the type of workflow requirements that can be encountered in a grid environment. A vast majority of the tasks can be executed in parallel, but some are inter-dependent, where the output of one job is processed and fed into another, and after each calculation a small post-processing step is required before the next simulation. Recursion may be required, and we must also take into account the large number of jobs, and the potentially large amount of data that will be produced. This does not make it possible for the computational process to be handled manually nor can it be computed by a single machine in a reasonable amount of time.

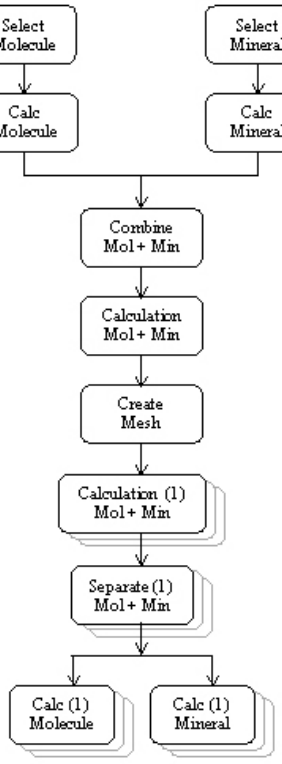

Figure 1: Sequence of simulations required to model the interactions between pollutants and soil minerals.

From a computational perspective, the structure of this process is hence a particularly complete example of the type of workflow problems that may be deployed in a grid environment. Not only is the workflow a combination of sequential and parallel executions but the process is also not *static*: the outcome of a particular subset of the process - (1) on the figure - could modify the course of action to be taken next: for example, a particular result may or may not require further processing to take place before moving on to other tasks in the process.

Coupled to job executions is the issue of data management: input data needs to be transferred alongside jobs to the machines responsible for their execution, output data needs to be stored and later retrieved from data stores, data might need to be parsed and converted to the format expected by a particular application. These data management tasks must be fully integrated into our computational process.

The difficulty in managing all of these tasks manually, particularly when jobs may take considerable amounts of time, in a scale of days or weeks, or may simply be too numerous to handle, means that we need to *automate* the workflow process.

3. The e-minerals mini-grid

The environment that we use to deploy our use case process is the e-minerals mini-grid; our production level grid infrastructure. Described in [3], it encapsulates a number of contributed and shared resources such as:

- 3 PBS-managed 16-node clusters
- The UCL Condor pool (~1000 nodes) and another Condor pool at Cambridge (26 nodes)
- An IBM pSeries parallel computer
- A Sunfire 880
- UK e-science level 2 grid resources such as HPCx and NGS (National Grid Service) systems.

It is worth briefly describing some aspects of our mini-grid architecture that must be taken into consideration when deploying a workflow.

All resources are accessed via a Globus Resource Allocation Manager (GRAM) interface – either 2.4 or 3.2 [4, 5]. The GRAM is meant to provide a secure and *standard* interface for job submission and monitoring, which will hence conceal from the users the differences that may exist between the different scheduling systems run by the underlying resources.

Regarding data management, a number of SRB (Storage Resource Broker) [12] vaults were set up on our dedicated resources to support the high data storage requirements of our users. The SRB essentially brings together a number of separate data stores (vaults) in order to make them appear to the user as a single unified virtual file system. It also provides functionality such as data replication in order to allow for increased robustness.

Though our mini-grid is in itself a powerful infrastructure, the fact that these resources are independently managed proves to be a relatively big obstacle to overcome when dealing with workflows. The GRAM may provide us with a standardised way of submitting jobs to a resource, but it is still merely a job submission interface and does not provide us with the means to manage and queue multiple submissions to a single or multiple independent resources. There is for example no simple and central way of determining resource availability; some resources may be heavily used, whilst others may be left mostly idle, and this may vary during the lifetime of the computational process. We must also find means of linking GRAM submission to SRB storage functions to ensure that required input data is retrieved from the SRB and transferred to the resource alongside the job for execution and output data stored back into the SRB upon job completion.

It is hence left to the client (i.e. the machine from which the user is initiating the computational process) to manage the deployment and correct execution of the computational process across these resources. This task would consist of:

- Selecting suitable and available resources on which to run jobs
- Handling individual job submissions to resources according to the workflow requirements
- Manage the transfer of executable binaries and input data retrieved from the SRB stores
- Monitoring the correct execution of the jobs
- Rescheduling jobs elsewhere if for example errors occur or new and better resources become available
- Transfer jobs back into the SRB for storage

4. Tools and usage techniques

4.1 Job submission management

Our approach in the e-minerals project has been to build upon established tools and technologies, such as Condor, Globus and the SRB, and we have maintained this principle when dealing with workflow problems. The tools and technologies that we have adopted to handle the computing aspects (job submission) are all part of the widely adopted Condor system [6][7]. Though primarily a high throughput computing resource management system for machine pools and clusters, it also provides us with the following client tools to manage the submission of jobs to Grid resources:

- *Condor-G*: Condor-G is a *client-side job scheduler*, used to submit jobs to resources presenting a Globus GRAM interface, such as our mini-grid resources. Essentially it provides

means for users to '*queue*' jobs to be submitted. It allows a large number of jobs and their requirements to be specified very simply, whereupon it will manage their individual submissions, handle the transfer of input and output files between client and resource and monitor the correct execution of the job. It is also much more flexible and convenient to use than Globus client tools, relying on the traditional Condor submission interface.

- *Condor-G with advertisements*: One of the limitations of using Condor-G is that users must specify explicitly the resource on which they want a job is to be run, and this does not really allow users to take advantage of an environment consisting of many different resources with different capabilities. To partially overcome this limitation, Condor-G can provide limited *matchmaking* capabilities: users can define advertisement files describing the characteristics of available resources, such as the operating system, memory available, maximum amount of jobs that can be run, etc, which will be used by Condor-G to determine which site is most suitable for a specific job to be run.
- *Condor glide-in*: Glide-in is the 'step above' Condor-G, providing us with means to respond to the dynamic nature of the grid. It essentially allows users to build a *personal Condor pool* out of grid resources. It achieves this by submitting server-side components (daemons) of Condor to selected sites as jobs via the GRAM. Once these daemons have been allocated resources and started, they will report back to the client and notify it of resource characteristics and availability. Condor-G can then submit jobs directly to these daemons, which will be responsible for monitoring their execution locally. This approach provides us with additional functionality, such as checkpointing – where the state of a job is saved on a regular basis so that it can be restarted from where it last stopped in case of failure – or migration of jobs – the execution of a job can be stopped and continued on another resource. From a workflow perspective, this addresses many of our requirements: allocations can be requested from all the various grid resources, and Condor will then dynamically assign queued jobs to suitable resources as they become available. Multiple resources can be exploited in parallel in this manner.
- *DAGMan (Directed Acyclic Graph Manager):* DAGMan is used in conjunction with the Condor-G scheduler (and hence glide-in) and allows us to specify dependencies that may exist between jobs. Following a specified sequence, DAGMan will submit jobs to the scheduler and monitor the queue for job terminations, upon which it will submit the next set of jobs in the sequence. The process is specified in the form of an acyclic graph, with jobs represented as parents and children nodes in the graph. It also provides us with means to run pre and post execution scripts on the client machine, which can be used for example to rename files or for simple clean up tasks.

An appealing aspect of these Condor tools is the fact that they require only the minimum to be set up on the resource. A user in a grid setting generally has very little influence over the management of the resources, and hence cannot make any assumption as to what would be available on the server side. The only requirement of the Condor middleware is a correctly configured and, in the case of glide-in, a complete (i.e. alongside the GRAM, GridFTP, and other Globus tools are required) Globus installation – Globus being a de-facto standard in grid computing.

4.2 Data Management

There are 3 aspects of data management that must be taken into consideration: data storage, data transfer – between client, execution and storage sites - and data transformation – which deals with the parsing and conversion of data from one application format to another.

As previously described, the SRB is used to handle our storage needs. The most obvious and perhaps simplest way of managing interactions with the SRB is for the client to retrieve input files and binaries from the SRB before job submission and store the output data into the SRB upon job completion; the actual transfer of files to and from the resource being handled by Condor-G's file transfer mechanisms. However, the client might quickly become a system bottleneck. Indeed, it must have sufficient storage capacity and transfer capabilities to handle the data produced and required by all jobs in a process, which may become highly problematic when these increase in number.

A much preferable approach is to ensure that data is transferred *directly* between resources and the SRB, with the client only being responsible for orchestrating the transfers remotely. This can be achieved by using DAG-Man to perform the following sequence of actions when submitting a job to a specific resource:

- Before submitting a job, a script is sent to be run on the selected resource, which will set up the necessary environment, retrieve required input files and binaries from the SRB and store them in a temporary location.
- The job is then submitted. Rather than relying on file transfer mechanisms; job requirements will specify the location of the required files on the remote resource, and a location as to where the output files should be stored on the remote resource upon completion.
- A final script is sent over to the resource, which, when run, will retrieve the output files from the specified location and store them into the SRB. The remote environment is then cleaned up and the script exits.

DAG-Man will ensure that these three tasks are performed in order and that each task is completed before the next begins. Appropriate applications were written to simplify the specification of these tasks, so that data management tasks can be specified within a Condor-G submission file, without having to actually deal directly with the DAG-Man scripts. These applications enable users to specify their jobs as they would normally with Condor-G, but with an additional set of parameters allowing them to specify which files should be retrieved and stored into the SRB from the actual resource. The actual 3-step process described above would hence be abstracted from the user by the application.

It is also possible to use DAG-Man for data transformation. Before and after submission of each job a small script can be run to parse and convert the data into and from the format expected by the executable program. However the use of data standards can considerably ease this task and we discuss this in a later section.

5. Applying these mechanisms to our workflow process

Though the tools and mechanisms described above allow us to meet many of our workflow requirements, there are still many issues that need to be addressed. One of the primary difficulties that we have encountered in applying these mechanisms is that they result in a potentially large collection of scripts, submission files and DAG-Man specification files being required to specify the process. In order for such a process to be manageable, it is necessary to adopt a unified approach to process specification. We have achieved this by writing a case specific front-end system using Perl, which will generate all the required files and submit these to the Condor middleware according to the user's requirements and input parameters.

If we consider individual tools and technologies, though Condor-G and DAG-Man worked in most cases 'out of the box', the requirements of Condor glide-in, with respect to infrastructure layout, have proven difficult to meet. In order to use glide-in with a cluster resource, it needs to be able to access individual nodes in order to communicate with the daemons it has submitted as jobs. This unfortunately is often difficult: individual execution nodes are often behind a firewall or on private local area network (LAN) and hence cannot be accessed externally. There are means to circumvent this problem (open firewall ports – add client to private LAN), but we can only achieve this with resources over which we have administrative control (i.e. dedicated e-minerals resources). We hence have built our initial workflow system limiting ourselves to using mainly Condor-G, advertisements, and DAG-Man/SRB links into our application.

6. Adopting standards to facilitate workflow management

6.1 CML: The Chemical Mark-up Language

CML (Chemical Mark-up Language) is an XML (eXtensible Mark-up Language) based data formatting standard for chemical data. It defines a standardised way of representing this data, concentrating specifically on molecules, and supporting amongst other things a hierarchy for compound molecules, reaction and macromolecular structures/sequences. Being based on XML it can be easily be extended to incorporate additional information if required. CML has been described in [6].

With regards to workflow and combinatorial problems, adopting a data presentation standard can solve the problem of incompatibility between application specific-formats and remove the need for data parsing and conversion. Indeed, each application used in a workflow may have its own specific ways of representing similar information and it is often necessary to rely on parsers to transform data from one representation to the next. Of course different parsers may be required for different types of conversion.

CML-enabled versions of applications allow increased compatibility and facilitate workflow integration. The e-minerals team has been involved in a number of CML related projects. We have for example participated in the development of a CML-enabled version of SIESTA [2] for use in the case described above and are currently in the process of developing a CML-enabled version of DL-POLY, modifying individual functions of the application for these to be capable of processing or producing CML data.

6.2 Web Services for Grid computing

The recent adoption of Web Services by the Grid computing community, initiated with the Open Grid Services Architecture (OGSA) [13] and later continued with the Web Service Resource Framework (WSRF) [14], has opened up access to a wide range of Web Service technologies and tools, and of particular relevance to this article are Web Service orchestration technologies, such as the Business Process Execution Language (BPEL) [11].

These differ from traditional grid workflow specification tools by specifying a workflow in terms of service invocations rather than jobs. This approach allows all the various aspects of our workflow – execution management, data management and general service management aspects such as fault handling - to be specified as a single integrated process rather than handled separately. The BPEL language also allows us to specify conditional statements (where we can specify behaviour according to different outcomes), specify exceptions (where one can state what actions to take when an error occurs), and manipulate XML-based data (such as CML) within the process rather than have modifications of input files being performed via independent scripts.

Adopting BPEL would allow our workflows to be specified in a unified and manageable manner. The only drawback that we have currently identified would be that specifying such processes require an intricate knowledge of the inner workings of Web Services and an interesting area of research would be the abstraction of such a language to a scientific level (e.g. using CML as the data standard). We are hoping that current work with WSRF [9] will enable us in the future to further investigate the use of BPEL as a workflow specification language.

It should be noted that many grid technologies have been or currently are being brought in line with Web Service standards, including the Globus toolkit, the main building block of our mini-grid and Condor, development of which is partially handled by e-minerals project members. These standardisations of tools will considerably simplify interactions with a wider range of technologies at the infrastructure level.

7. Conclusion

We have described here some of the techniques that we are currently using in the e-minerals project to deal with workflow specification and management, and some of the future directions that are being taken. Our approach so far has been to focus on “out of the box” technologies, in order to reach a production level infrastructure within a reasonable amount of time, and also to demonstrate that much can actually be achieved with existing tools and technologies. Many of the problems that we have faced have much to do with the actual integration of technologies, and most development that has been done has gone into actually simplifying the means by which we specify workflows, and reduce the need for users to actually deal with the complexity of specifying every interaction.

This workflow process was an opportunity to identify repeating patterns, such as the need to transfer data from the SRB to the actual resource at job execution time as described in section 4.2, and develop applications that would operate at a higher level of abstraction.

There is nevertheless still much to do in terms of integration, and much of this has to do with reaching a higher level of flexibility - specifying conditional statements and exceptions, dynamically modifying input data, etc. - whilst still maintaining a certain degree of abstraction in order to make complex workflow specification relatively accessible for users without important computational skills. Hopefully the incorporation of Web Services, CML and BPEL will go some way towards assisting us at the infrastructure level. At the very least this will provide with a sound basis to develop further higher-level services such as the DAG-Man/SRB application (section 4.2) to assist the users in achieving their goals.

References

[1] Wells, S., Alfredsson, M., Bowe, J., Brodholt, J., Bruin, R., Calleja, M., Catlow, R., Cooke, D., Dove, M., Du, Z., Kerisit, S., de Leeuw, N., Marmier, A., Parker, S., Price, D., Smith, B., Spohr, H., Todorov, I., Trachenko, K., Wakelin, J., Wright, K. Science outcomes from the use of Grid tools in the e-minerals project, *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[2] Garcia, A., Murray-Rust, P. and Wakelin, J., The use of XML and CML in Computational Chemistry and Physics Programs, *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[3] Blanshard, L., Brodholt, J., Bruin, R., Calleja, M., Chapman, C., Dove, M., Emmerich, W., Kleese van Dam, K., Tyer, R., and Wilson, P., Grid tool integration within the e-Minerals project, *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[4] The Globus Project, 2003. http://www.globus.org/toolkit

[5] Foster, I., Kesselman, C., Globus: A Meta-computing Infrastructure Toolkit, *Intl J. Supercomputer Applications*, 1997

[6] Livny, M., Miller, K., Tannenbaum, T. and Wright, D. Condor - A Distributed Job Scheduler, in Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*, The MIT Press, 2002.

[7] Condor Team, Condor Version 6.7.0 Manual**.** University of Wisconsin-Madison, 2003. http://www.cs.wisc.edu/condor/manual/v6.7/

[8] Murray-Rust, P. and Rzepa, H. Chemical markup, XML and the World-Wide Web. 2. Information objects and the CMLDOM. *J. Chem. Inf. Comput. Sci,* 2001.

[9] Chapman, C., Wilson, P., Tannenbaum, T., Farrellee, M., Livny, M., Brodholt, J., and Emmerich, W., Condor Services for the Global Grid: Interoperability between OGSA and Condor, *Proc. Of the All Hands Meeting 2004*, Nottingham, 2004.

[10] HPCxm Capability computing, documentation. http://www.hpcx.ac.uk/support/documentation/index.html

[11] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S., Business Process Execution Language for Web Services, Specification v1.1, 2003, ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[12] San-Diego Super Computing, Storage Resource Broker Documentation, http://www.npaci.edu/dice/srb/docs.html

[13] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002.

[14] Foster, I., Frey, J., Graham, S., Tuecke, S., Czajkowski, K., Ferguson, D., Leymann, F., Nally, M., Sedukhin, I., Snelling, D., Storey, T., Vambenepe, W., Weerawarana, S. Modelling stateful resources with Web Services, Version 1.1, IBM, 2004