Adding Instruments and Workflow Support to Existing Grid Architectures

D.J. Colling[1], L.W. Dickens[1], T. Ferrari[2], Y. Hassoun[1], C.A. Kotsokalis[3], M. Krznaric[1], J. Martyniak[1], A.S. McGough[1], and E. Ronchieri[2]

[1] Imperial College London
{d.colling, luke.dickens, y.hassoun, marko.krznaric, janusz.martyniak, andrew.mcgough}@imperial.ac.uk
[2] INFN, Italy
{Tiziana.Ferrari, Elisabetta.Ronchieri}@cnaf.infn.it
[3] GRNET, Greece
ckotso@grnet.gr

Abstract. Many Grid architectures have been developed in recent years. These range from the large community Grids such as LHG and EGEE to single site deployments such as Condor. However, these Grid architectures have tended to focus on the single or batch submission of executable jobs. Application scientists are now seeking to manage and use physical instrumentation on the Grid, integrating these with the computational tasks they already perform. This will require the functionality of current Grid systems to be extended to allow the submission of entire workflows. Thus allowing the scientists to perform increasingly larger parts of their experiments within the Grid environment. We propose here a set of high level services which may be used on-top of these existing Grid architectures such that the benefits of these architectures may be exploited along with the new functionality of workflows.

1 Introduction

As the Grid is becoming more significant to the application scientist, they now seek to perform ever more complex work with it. Initially they have sought to run computationally expensive executables over the Grid. Hence the Grid has evolved (in general) as a batch submission system (for example [1, 6, 14, 18]). However, as their confidence has grown with the Grid, they now seek to orchestrate the whole of their scientific process on the Grid. This will often require use of scientific equipment along with the storing and/or processing of instrument output. This process may be repeated many times before the scientist obtains the results that they desire. The way in which these tasks are performed and the way in which they interact with each other can be described in terms of a workflow.

By adding instruments into the Grid we not only increase the requirements on the Grid to support workflows but also increase the requirement for many other functionalities. Many of these are well known to the Grid community, although in general these are not widely supported yet.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 956–963, 2006.

Workflows are not a new concept to the application scientist. Many have their own ad-hoc ways of performing workflows. In general these often require much human intervention, thus they are eager to automate this process as much as possible. However, the automated processing of workflows through a Grid architecture needs to proceed in a timely manner and match any Quality of Service (QoS) requirements the application scientist may have. To achieve this, reservations become important, both for the instruments and the existing Grid resources. Instruments are often in high demand, thus the ability to reserve time on them becomes significant. Conversely when an experiment is run on such an instrument, it is vitally important that the Grid resources are available to cope with the results produced. In order to make reservations it is also essential to provide a framework for the establishment of *Service Level Agreements* (SLAs).

In order to provide workflow support within the Grid it is possible to develop a completely new architecture from the ground up. Although this does have the advantage that the architecture can be defined to be workflow aware from the outset, it does have the disadvantage of not reaping the benefits already achieved from the existing Grid middlewares, such as wide acceptance within the community or robustness evolved from community exposure and hardening.

In this paper we propose how to augment existing Grid middlewares with workflow support. Through this we will use as much as possible of the existing middleware, such as the ability to load-balance and schedule tasks or monitor the running of these tasks. Workflow functionality is thus added as a higher level service on-top of the existing infrastructure.

We present in section 2 an overview of related work in this area. Section 3 describes the higher level services required to provide workflow and instrument support to the existing Grid architectures while section 4 provides more specific details for our work within the GRIDCC project. We conclude in section 5 and describe how we intend to progress with this work.

2 Related Work

Many Gird workflow systems and tools exist, such as: Askalon [9], DAGMan [16], GridFlow [4], Gridbus Workflow [23], ICENI [11, 13], Pegasus [5], and Triana [17]. Yu and Buyya [24] present a detailed survey of existing systems and provide a taxonomy which can be used for classifying and characterising workflow systems. We use elements of the taxonomy related to Quality of Service (QoS) and workflow modelling for comparing our approach to other systems.

A prominent feature of adding instruments to the Grid is the need for real-time remote control and monitoring of instrumentation. Users and applications will be allowed to make *Advance Reservation* (AR) of computational resources and instruments. AR guarantees availability of resources and instruments at times specified [15]. In ICENI, the scheduling framework supports AR by using the meta-data of application components together with corresponding historical data stored for the purpose of performance analysis and enhancement [12, 21, 22]. This approach is also used in Pegasus, Gridbus Workflow and Askalon. Some

systems like Askalon, DAGMan, ICENI, GridFlow and Gridbus Workflow allow users to define their own QoS parameters and to specify optimisation metrics such as application execution time, desired resources and economical cost. Other systems such as Pegasus and Triana do not. We are using a similar mechanism for performance enhancement and estimation taking into account the real-time QoS constraints.

3 The Architecture

In this section we outline a generic architecture which may be applied to many of the existing Grid architectures. Figure 1 illustrates this generic architecture for providing workflow and instrument support. The rest of this section outlines the requirements for each of the items in this generic architecture and the functionality they should provide.

Planner: The planner sits between the user environment and the Grid providing a stable interface for both application scientists and computer experts.

The planner accepts complex abstract workflows from the user describing a application which needs to be executed along with any QoS requirements. QoS may include requirements on total (or partial) runtime and/or maximum price. When writing workflows, the user should generally not be concerned with choosing specific resources, but rather with resource/service types. This not only hides unnecessary detail, but also allows writing portable workflows for submission at different sites and different times.

User-specified high-level constraints are then decomposed by the planner in terms of a simple set of basic pattern behavior [20]. In order to draw a quantitative picture of the system, the planner extracts performance parameters about resources and then independently analyzes the information to produce a concrete set of requirements. Equipped with the above parameters from the available resources, the planner builds an optimal workflow and deploys it using a workflow engine.

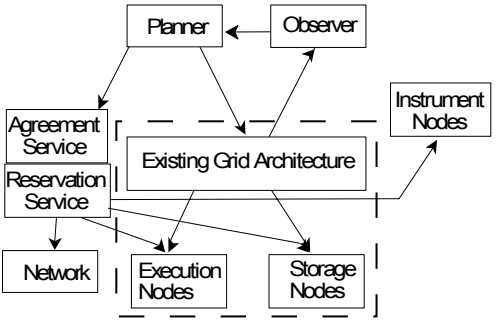

Fig. 1. The generic Architecture

Observer: In an ideal world once the planner has finished its task the workflow can be deployed successfully to the selected resources and run as desired. However, the nature of the Grid is that of an environment where resources may appear and disappear without warning, network links break and tasks not perform as expected. This may be both to the advantage of a application scientists workflow as well as a disadvantage.

The observer is designed to monitor not only the progress of the executing workflow but also the state of the Grid. If the workflow is not operating as expected (both in a good or a bad sense) then it can prompt the planner to alter the workflow plan. Alternatively if the observer discovers new more appropriate resource it can prompt the planner to make use of these.

Reservation Service: Advance reservation is one of the mechanisms used to support reliable service levels, for a specific time interval in the future, by a set of resources satisfying specific requirements. This mechanism is useful to various mission-critical applications, for example to Grid applications for the reservation of network bandwidth for bulk transfers, to Data Scheduling services requiring storage space reservation prior to the start of a data transfer session, and to workload management services to allocate computing power (e.g., to high-priority execution tasks).

Instrument Node: The Instrument denotes a coherent collection of services that provide all the functionality for controlling and monitoring of a physical instrument. The main features of IE are:

1. Accessibility through standard interfaces.
2. Allowing for remote controlling and monitoring of physical instruments.
3. Providing exception handling mechanisms with very low latency which, in case of locally thrown exceptions, are capable of protecting physical instruments from potential harm.
4. Supporting information and monitoring services to insure effective and successful operation of the IE.

Existing Grid Architecture: has the role of allocating resources in such a way that concurrent user requirements are efficiently satisfied. Assigning resources with the aim to achieve optimal performance is known as the problem of resource allocation or *scheduling*. Globus [14], Condor [10, 16, 19], gLite [7] and Sun Grid Engine [8] are examples of middleware offering these capabilities. Typically they will abstract the underlying grid into **Execution Nodes** and **Storage Nodes**.

Security: As a remote, distributed and multiuser system, an interactive GRID is subject to a wide variety of threats. In order to eliminate unauthenticated and unauthorized access to the system a mechanism of secure information exchange has do be developed. In general this is best achieved through the use of certificates, based on user credentials, thus allowing for single sign-on to the Grid. Users can then be first authenticated before checking to see if they are authorised to use the particular service. In general non-symmetric key cryptography is preferred due to its higher level of security. Though as interaction with instruments may be time critical symmetric keys may be needed.

There are potential solutions available to achieve above requirements. The widely used X.509 certificates might be utilised to allow secure resource access, and when needed, a faster symmetric Kerberos cryptography system. A single sign-on might be realised by using proxy certificates.

Agreement Service: In order to make use of reservations within the Grid along with the ability to draw up SLA's it is required to have some form of service which can perform this task. This is carried out by the agreement service. This service is capable of making multiple agreements between different Grid resources which may or may not be concurrent.

4 Example Implementation Case: GRIDCC

Grid Enabled Remote Instrumentation with Distributed Control and Computation (GRIDCC) is a European project that aims at extending the current Grid technologies which provide batch access to distributed computational and storage resources, so these technologies include access to and control of distributed instrumentation.

4.1 Use Cases

A number of Grid workflow use cases involving the interaction with instruments have been identified in the framework of the GRIDCC project [2]. In this paper we focus on on distributed intrusion detection and meteorological conditions prediction. Other applications based on workflows involving the interactions with instruments are the high energy physics and power Grids.

Intrusion detection is applied to network infrastructures with the aim to detect *Denial of Service* (DoS) attacks. In this case Instrument Elements (IEs) are services responsible for collecting and processing large amounts of data gathered by network routers. The corresponding workflow is relatively simple as it contains one decision point and one endless loop. Firstly, data collected by routers is processed by the IE, which performs data collection, sampling, aggregation and filtering. Then, results are handed to a soft computing component which performs additional processing to reduce the volume of collected data. Finally, data is fed to a central decision-support system and a decision is made. If no attack is detected, control is returned to the IE in order to let the processing continue, otherwise the following tasks are performed in the system: processing takes place to back-trace it; network devices are reconfigured accordingly; the "reaction" phase of the workflow ends, while traffic processing on the IE continues.

The meteorological use case concerns an application with the aim to use meteorological data for whether prediction and alerting when extreme whether conditions are expected. The meteorological workflow is more complex, as it includes a variety of Grid tasks and workflow patterns, such as advance reservation, multiple decision points, branched output, nested loops, etc.

The upper-bounded time requirements of these two applications have to do with the requirement to react urgently under an unusual situation: The presence of an attack, or probabilistic model results indicating extreme whether conditions.

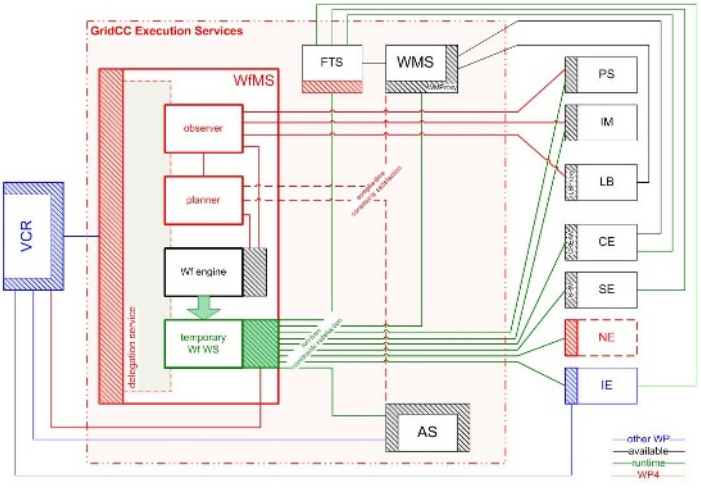

Fig. 2. The GRIDCC Architecture

4.2 The GRIDCC Architecture

In this section, we give an account of the GRIDCC architecture. To give the GRIDCC components relevance, they must be placed in context with an existing grid toolbox; here the gLite [7] model is used. Figure 2 shows the overall structure of this architecture. More details are available in [2]. The GRIDCC architecture represents a realisation of the abstract architecture depicted in Figure 1. The GRIDCC architecture constitute a set of components each of which represents a set of services. There are services for planning and submitting workflows, services for maintaining and monitoring the tasks that make up the workflow, agreement services for reserving resources, for monitoring and controlling physical instruments and services dealing with security issues. GRIDCC uses the EGEE-gLite packages [7] where the main components of the GRIDCC architecture are:

The Virtual Control Room (VCR) which represents the user interface that enables users to build and submit complex workflows.

The Instrument Element (IE) which represents a novel concept to the GRIDCC project. A Local Problem Solver (LPS) is embedded within IE for diagnosing and reacting to error conditions in such a way as to protect the physical instrument from harm. Also, a performance Information and Monitoring Service (IMS) is provided to ensure LPS reliability and successful operation of the IE.

Compute and Storage Elements (CE & SE) are similar to those which exit within other Grid projects.

The Execution Services (ES) which are responsible for the execution of the user-defined workflows and maintaining the status of the tasks that make up the workflow. The ES include: the Workflow Management System (WfMS) for workflow management – which is decomposed here into the Planner, the Observer,

a workflow engine (provided by a third party), and a temporary workflow Web Service – the Workload Management System (WMS) for logging and bookkeeping and service discovery, and the Agreement Service (AS) for reservation management. The File Transfer Service (FTS) is used for staging files to and from the Compute Elements as required.

The Agreement Service (AS) which is responsible of handling resource reservation requests and is invoked by the WMS. The AS controls the quality of service by making resource reservation agreements with the available resources, IEs, CEs or SEs. The AS attempts to satisfy a single resource reservation request by invoking one or more resource service providers.

The Problem Solver (PS) which may be manifested as either a Local Problem Solver (LPS) or a Global Problem Solver (GPS) is a new element included into the Grid in an attempt to protect the instruments. A Local Problem Solver will monitor a single (or small collection of) instruments in an attempt to prevent the instrument being damaged due to incorrect use from the Grid. The GPS performs the same task but deals with problems caused by collections of instruments.

Information and Monitoring (IM) service gathers information about all services currently active within the Grid.

Network Element (NE) is a service to represent the network within the Grid. This allows networking resources to be reserved.

5 Conclusions and Future Work

We have implemented the GRIDCC workflow architecture which is capable of taking a workflow deploying this to a workflow engine. The engine is then able to submit the stages of the workflow to the existing (in this case gLite) Grid middleware. We are now investigating techniques to manipulate the workflows that are submitted in order to better meet the QoS requirements of the system. We are looking into the techniques proposed in [3]. We are also developing the agreement and reservation capabilities.

References

1. Project SGE. `http://wwws.sun.com/software/gridware/`.
2. The GridCC Architecture Version 1.1.4. http://www.gridCC.org, 2005.
3. A. S. McGough and J. Cohen and J. Darlington and E. Katsiri and W. Lee and S. Panagiotidi and Y. Patel. An End-to-end Workflow Pipeline for Large-scale Grid Computing. Accepted for inclusion in the Journal of Grid Computing.
4. J. Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. GridFlow: Workflow Management for Grid Computing. In *Proceedings of 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan.* IEEE CS Press, Los Alamitos, 12–15 May 2003.
5. E. Deelman, J. Blythe, Y. Gil C. Kesselman, G. Mehta, K. Vahi, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1(1):9–23, 2003.

6. EGEE. Enabling Grids for E-science. `http://public.eu-egee.org/`.
7. eGee gLite Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/, 2005.
8. Sun Grid Engine. http://www.sun.com/software/grid/, 2005.
9. T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto Jr, and H. L. Truong. ASKALON: a tool set for cluster and Grid computing. *Concurrency and Computation: Practice and Experience*, 17(2-4):143–169, 2005.
10. M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of 8th International Conference of Distributed Computing Systems (ICDCS), Los Alamitos, CA, USA*, pages 104–111. IEEE CS Press, 1988.
11. A. Mayer, S. McGough, N. Furmento, W. Lee, S. Newhouse, and J. Darlington. ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time. In *UK e-Science All Hands Meeting, Nottingham, UK*, pages 894–900. IOP Publishing Ltd, Bristol, UK, Sep. 2003.
12. S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Performance Architecture within ICENI. In *UK e-Science All Hands Meeting, Nottingham, UK*, pages 906–911. IOP Publishing Ltd, Bristol, UK, Sep. 2004.
13. S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Workflow Enactment in ICENI. In *UK e-Science All Hands Meeting, Nottingham, UK*, pages 894–900. IOP Publishing Ltd, Bristol, UK, Sep. 2004.
14. Globus Project. http://www.globus.org, 2005.
15. S. Andreozzi and T. Ferrari and S. Monforte and E. Ronchieri. Agreement-based Workload and Resource Management. In *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing*, Melbourne, Australia, December 2005. IEEE Computer Society.
16. T. Tannenbaum, D. Wright, K. Miller, and M. Livny. *Condor - A Distributed Job Scheduler. Beowulf Cluster Computing with Linux.* The MIT Press, MA, USA, 2002.
17. I. Taylor, M. Shields, and I. Wang. Resource Management for the Triana Peer-to-Peer Services. In J. Nabrzyski, J. M. Schopf, and J. Węglarz, editors, *Grid Resource Management - State of the Art and Future Trends*, pages 451–462. Kluwer Academic Publishers, 2004.
18. Condor Team. Condor Project Homepage. `http://www.cs.wisc.edu/condor`.
19. D. Thain, T. Tannenbaum, and M. Livny. *Condor and the Grid. Grid Computing: Making the Global Infrastructure a Reality.* John Wiley & Sons, NJ, USA, 2003.
20. W.M.P. van der Aalst. Workflow Patterns. `http://www.workflowpatterns.com/`.
21. L. Young and J. Darlington. Scheduling Componentized Applications on a Computational Grid. MPhil/PhD Transfer Report, Imperial College London, University of London, 2004.
22. L. Young, S. McGough, S. Newhouse, and J. Darlington. Scheduling Architecture and Algorithms within the ICENI Grid Middleware. In *UK e-Science All Hands Meeting, Nottingham, UK*, pages 5–12. IOP Publishing Ltd, Bristol, UK, Sep. 2003.
23. J. Yu and R. Buyya. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In *Proceedings of 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, USA*. IEEE CS Press, Los Alamitos, 8 Nov. 2004.
24. J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing. GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005.