

# ***SUPPORTING ON DEMAND, POLICY BASED MONTE CARLO PRODUCTION, LEVERAGING CLARENS, AND RUNJOB***

F. v. Lingen, J. Bunn, I. Legrand, H. Newman, C. Steenberg, M. Thomas, Y. Xia, California Institute of Technology, United States<sup>1</sup>

D. Bourilkov, R. Cavanaugh, University of Florida<sup>2</sup>

D. Evans, E. Lipeless, S.C. Hsu, Fermi National Accelerator Laboratory<sup>3</sup>

T. Martin, A. Rana, F. Wuerthwein, University of California San Diego<sup>4</sup>

## ***Abstract***

We describe a set of web services, created to support scientists in performing complex, tasks (such as Monte Carlo production). The web services described in this paper provide a portal for scientists to execute complex applications which can consist of many consecutive steps (e.g. work flows). The main design goal of the web services discussed is to provide controlled access for (multiple) set(s) of users in different roles (e.g. scientists, administrators, grid operators,...) to complex applications without the added trouble of updating, configuring, and patching these ever evolving applications and keep the users focused on their core tasks (e.g. scientific analysis). The web services have been implemented using the Clarens grid toolkit. This Python (and Java) based toolkit provides, amongst others, x509 authorization, access control and VO management for its services. The (stateful) web services discussed in this paper re-use several of these Clarens components in providing access control and usage quotas. Initially the services described in this paper where developed to support users in (private) Monte Carlo production activities, however due to their generic design, can be used to expose other (potentially complex) applications to users as will be shown in this paper.

## **INTRODUCTION**

Many Scientific software applications are complex - they involve components under development, intricate inter-dependencies and rich functionality, with the choice of many operating parameters. Some of these applications have been extended to become "grid aware" and are thus amenable to being submitted to grid Schedulers or grid sites directly. These features add to the complexity of the applications.

Often, due to this complexity, scientists prefer not to install these applications on their laptop/desktop, as the maintenance burden is too distracting from their scientific research goals: scientific discovery through data analysis. Instead, they favour a working environment in which the

applications are available, but kept up to date and ready to use by a small group of experts. Previous versions of the applications are also made available, and easy access is provided to the most often used setup environments.

The web services portal described in this paper supports this type of working environment. The scientist only needs to browse through a collection of applications and work flows hosted at centers, select one, supply parameters, and then initiate the task and wait for the result of jobs being executed on the grid. This scheme, which is based on the "*me, my friends and the grid*" paradigm [10] makes it easier for scientists to be productive. Many tasks can be initiated, and results returned rapidly, by "clicking away". Consequently, the portal supports forms of access control, quotas and policy enforcement that can be used to throttle resource usage if necessary.

The model in which these web services are designed to operate, envision a collection of Tier2s, primarily for High Energy Physics (HEP), providing services to scientists in supporting them doing analysis on the grid. Each scientist is associated to a Tier2, although this does not exclude him from using other Tier2s. The Tier2 a scientist is associated to, is usually geographically closest and the scientist is familiar with the persons managing the Tier2 (we assume that this social aspect will still be important as things go wrong with software and hardware) which function as a support team for the scientist. Typically within HEP (in the US), Tier2s will serve between 50-70 scientists. Initially the web services portal had been designed to support usage of the generic (work flow) tool RunJob [3] and to expose arbitrary work flows associated to Monte Carlo production within HEP.

In the remainder of this paper we use the term workflow to identify an application that can consist of multiple applications where the output of one, serves as input for another. These applications are typically executed in a grid environment where they can be executed on different computing resources.

## **OVERVIEW**

Figure 1 shows an overview of several of the components. Multiple centers host the services and provide access to scientists associated to this center as stated in the operational model of the previous paragraphs. The services have an administrative and a scientist (=user) component. Before a scientist can get

---

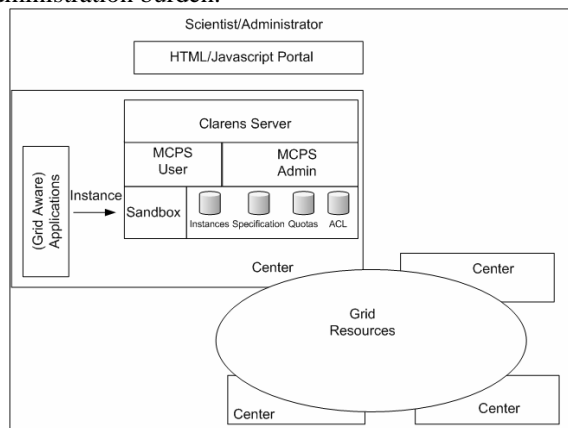
<sup>1</sup> {newman,conrad,thomas}@hep.caltech.edu,  
{fvlingen,julian.bunn,yxia}@caltech.edu, iosif.legrand@cern.ch

<sup>2</sup> {bourilkov,cavanaugh}@phys.ufl.edu

<sup>3</sup> tmartin@physics.ucsd.edu, {rana,fkw}@fnal.gov

<sup>4</sup> {evansde,lipeles}@fnal.gov, schsu@physics.ucsd.edu

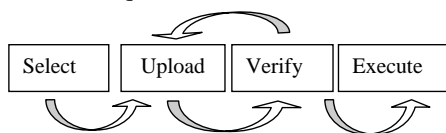
access to a workflow the distinguished name of his certificate needs to be added to a group that is granted access to this workflow, by the administrator. Administrators can be located via the Clarens based discovery service [1],[2] which associates names and email addresses to server and service instances. Each workflow has an owner who not only give users access to the workflow he owns, but has the ability to make users administrators of this workflow, hence distributing the administration burden.



**Figure 1. Overview of the service components**

The submission process is an interaction between the Clarens based web services and the HTML/Javascript client (portal) using an AJAX like technology based on JSON (JavaScript Object Notation) for communication.

A user can select only workflows that the workflow administrators allow him access to. This information is stored in the Access Control database. The quota database specifies the frequency with which he can use this service. For example it can be set to once per hour, while other groups or individuals can use this (or other) workflows at a different rate. Both ACL and quota can be viewed as a form of policy management. It does not address other policy issues such as storage quotas, CPU usage, and network quotas (nor is it intended to).



**Figure 2. State diagram of the user actions, supported by a stateful web service**

The authors argue however that *this type of policy management, provides some level of fair access\** to resources, such as storage, network and CPU, and was motivated under the assumption that resources in the grid will be scarce. The steps a user takes (once he gains access) in executing a workflow are depicted in Figure 2 and are enforced by a web service instance that keeps track of the state of the user session.

\* We leave it up to the administrators and virtual organizations they represent to decide what fair is.

The workflows that are exposed to users are described in the specification database. The specification consists of workflow name, a set of parameters, the parameter format, if parameters are mandatory or not, and sometimes a set of choices a user can select. No choices means a user needs to supply its own (e.g. a file name). Once the user selects a workflow and submits values for the parameters, a sandbox is created for this session and only the user has access to this sandbox. The sandbox serves as a “launching pad” for the workflow instance in submitting jobs to grid resources. The HTML/JavaScript client uses the sandbox ID and the distinguished name of the user certificate to manage the state of the web service session. The client enables a user to save his settings (which includes the sandbox ID); end or reset the client application; load these settings at a later time, and continue where he stopped.

Certain workflows do not only allow users to submit a list of parameters, but also have the possibility of using plug ins a user can supply. For example in HEP such an application can be ROOT[4] which can accept user code that represents an analysis. Users can upload files through the client. *They can not however (for security reasons) execute any piece of code they upload in the sandbox.* Only the workflow is capable of doing that. As many of the workflows will be executed in a grid environment several of these workflows require the user to submit his proxy that will be used by the workflow. The proxy is inserted in the sandbox and can pose a security risk when the machine hosting the sandbox is compromised. A requirement for workflows is therefore that *the workflow needs to delete the proxy if it does not need it any more.* This minimizes the risk, however certain workflows can run for several days where they submit grid jobs that need a proxy, go to a sleep state, wake up several hours later and submit again.

When a user presses execute in the client, the input of the user will be verified first. Two levels of verification take place: the first level examines the input of the user (e.g. is it an integer, is it a string, and so on). The second level passes the parameters (and possible code) to the selected workflow which runs its own customized verification script which can be different for every workflow. If there is an error at one of these verification levels the code will not execute. *This level of verification minimizes the risk that jobs crash during their execution and thus waste valuable grid resources.* Once execution starts, an entry is made into the instances database. This database logs the user (his distinguished name), the workflow name, and parameter values.

As shown in Figure 1, four databases are used. All these databases are relative small (smaller than a Megabyte) and could have been implemented as one database. The reason for this separation is that both the quota and the ACL database can be used by other services. For example the ACL database is based on the standard Clarens ACL database, which is also used by other Clarens based web services such as the system and file service [2]. The ACL database is based (for

performance reasons) on Berkeley DB (<http://www.sleepycat.com/>), while other databases are using MySQL (<http://www.mysql.com/>), as performance was not an issue for the latter. In order to keep information in the four databases consistent a two phase commit protocol is used.

The approach taken in construction of this workflow portal and web service differs from some other workflow projects: A user can not define his or her own workflow other than using the output of one workflow (after execution has completed) as input for another workflow. The reason for this limited functionality is two fold: first the input/output paradigm is not envisioned to occur a lot, other than that users want to analyze the output of one execution first before deciding to use it as input for another. Second, the workflows a user can select already contain several steps that execute different applications. For example a Monte Carlo workflow contains four stages (which are four different applications): generation, simulation, reconstruction and storage. These workflows are managed (due to their complexity) by experts while users can execute them by supplying different values for parameters exposed by the workflow.

As a consequence of the model presented in Figure 1, users would need to request access at different centers, although we assume that users typically will only use one center. There is however the possibility that this center is unavailable for unspecified periods (maintenance, software/hardware crashes, or natural disasters).

## PERFORMANCE AND EARLY RESULTS

Although performance is important, the performance of these web services does not have to be very high. The server itself should handle approximately 5 calls per second for these web services<sup>†</sup>, and the response time for results from web service calls can be 2 to 3 seconds, although response time has typically been measured below 1 second. To limit the latency between call and result, the client side also caches information about the workflows available. Typically the execution time for workflows exposed by these web services run between one hour and several days. As stated in the introduction our operational model envisions 50-70 users and it is very unlikely they will all access the services at the same time.

A first prototype of web services and the JavaScript portal front end has been released. The first workflow that has been deployed is based on Monte Carlo production of the Compact Muon Solenoid experiment (CMS, <http://cmsinfo.cern.ch/>): a sequential workflow that consists of 4 steps: event generation, detector simulation, reconstruction and data storage.

Two additional workflows have been created. Neither of these two workflows is related to the Monte Carlo process these web service where designed to support, showing the wider applicability of the web services. The first workflow exposes a collection of datasets (located at

different sites) from a catalog to an authorized user. This user can then decide to move one or several of these datasets to the site that hosts the web service. Although currently it connects to a static catalog (it exposes a pre defined set of datasets), this workflow exposed the need to have dynamic parameters in the workflow specification: parameters that have not a collection of predefined values a user can select, but who's value can be updated dynamically as it queries external applications (e.g. catalogs).

## RELATED WORK

Providing web service based access and workflow support has been investigated in past projects and we list a short (non exhaustive) list of them.

gridAssist [5] contains a Java based graphical front end and enables users to construct workflows, execute them and monitor their progress as different steps are executed on potentially different grid resources. Pegasus [7] is an application that accepts abstract workflows and maps it to the available grid resources. Pegasus may map the entire workflow at once or portions of it depending on the execution environment. The abstract workflows can be constructed by using Chimera [9] (which also deals with data provenance) or can be written directly by the user. Triana [6] is an open source environment in which users can combine various data analysis components. Triana includes a large library of pre-written analysis tools and the ability for users to integrate their own tools. Several of these analysis tools have been made grid enabled (e.g. submitting remote jobs).

Several of these projects have been assessed to determine if prior results can be used. The main requirement for this project was that the web services described in this paper should become the front end for RunJob [3] as this is being used within the CMS domain. Most existing application offer users the possibility to create their own workflows. Within the domain the services where designed for, workflow creation by users does not occur often. More likely is that users examine the output of one step and then decide what to do in the next step (iterative). Both the RunJob requirement and iterative behaviour limited the use of other Workflow applications. It would have taken considerable time to integrate workflows already specified and implemented using RunJob into existing Workflow applications, instead we choose to create a thin web service layer from scratch that exposes workflow applications to authorized users through a Clarens based web service portal. Clarens was chosen as it provides several components needed for these web services, such as authentication, authorization, access control lists, remote file access, and group management.

## SUMMARY

This paper describes a web service based portal that provides gradual, policy based access to complex (grid aware) applications, without having scientists to worry

<sup>†</sup> Tests showed that the Python version of the Clarens server can handle 1400 calls per second [2]

about installation details and frequent updates, enabling scientists to focus on their core activities such as data analysis.

The current portal is based on a HTML/JavaScript front end, a Clarens web service back end, and provides authorization and access control based on X509 certificates. A future front end of this prototype will be based on Java as part of the Clarens web services GUI framework allowing users to be seamlessly connected (after authentication) with multiple grid resources and providing developers the ability to develop GUI plug ins. Although current policy is based on a simple quota mechanism, future work will take into account the progress being made on computing, storage and network policies being addressed in projects such as UltraLight [8] and Lambda Station (<http://www.lambdastation.org/>). User identities for authorization are handled by Clarens but will be extended to work with grid User Management System (GUMS) servers [11] which is used within USCMS and the Open Science grid (<http://www.opensciencegrid.org>).

Other work will focus on providing a distributed environment for storing policy and workflow specifications and sharing this information between sites, limiting the number of times a user will need to request access at different centers hosting these web services and preventing a single point of failure on the database level.

In the current prototype every site hosts this portal on one single machine. As certain workflows that submit jobs to the grid can generate a high load, there will be a need for load balancing where one site uses for example 5 machines to distribute the load of the different workflows that are being executed.

## ACKNOWLEDGEMENTS

This work is partly supported by the Department of Energy grants: DE-FC02-01ER25459, DE-FG03-92-ER40701, DE-AC02-76CH03000 as part of the Particle Physics Datagrid project, DE-FG02-04ER-25613 as part of Lambda Station, and by the National Science Foundation grants: ANI-0230967, PHY-0218937, PHY-0122557, PHY-0427110, ANI-0113425, ANI-0230967, EIA-0303620. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Department of Energy or the National Science Foundation.

## REFERENCES

- [1] M. Thomas, C. Steenberg, F. van Lingen, H. Newman, J. Bunn, A. Ali, R. McClatchey, A. Anjum, T. Azim, W. ur Rehman, F. Khan, J. Uk In, "JClarens: A Java Framework for Developing and Deploying web services for grid Computing", In Proceedings of 2005 International Conference on web services (ICWS 2005), Orlando, Florida, July 11-15, 2005, IEEE Computer Society Order Number P2409 ISBN 0-7695-2409-5 pp141-148
- [2] F. van Lingen, J. Bunn, I. Legrand, H. Newman, C. Steenberg, M. Thomas, A. Anjum, T. Azim, "The Clarens web service Framework for Distributed Scientific Analysis in grid Projects", In Proceedings of the International Conference on Parallel Processing Workshops, Oslo, Norway, June 2005, IEEE Computer Society Order Number P2381, ISBN 0-7695-2381-1, pp45-52
- [3] P. Love, I. Bertram, D. Evans, G. Graham, "Cross Experiment Workflow Management: The Runjob Project", In proceedings of Computing for High Energy Physics (CHEP), October 2004, paper 385, (<http://indico.cern.ch/contributionDisplay.py?contribId=385&sessionId=23&confId=0>)
- [4] F. Rademakers, R. Brun, "ROOT: an object-oriented data analysis framework", in Linux Journal archive Volume 1998, Issue 51, (July 1998) Article No. 6 1998, ISSN:1075-3583
- [5] M. ter Linden, H de Wolf, R. Grim, "gridAssist, a User Friendly grid-Based Workflow Management Tool", in Proceedings of the International Conference on Parallel Processing, Oslo, June 2005, IEE Computer Society Order Number P2381, ISBN 0-7695-2381-1, pp5-10
- [6] S. Majithia, M. Shields, I. Taylor, I. Wang, "Triana: A Graphical web service Composition and Execution Toolkit", in Proceedings of the IEEE International Conference on web services, pp514-524
- [7] G. Singh, E. Deelman, G. Mehta, K. Vahi, Mei. Su, B. Berriman, J. Good, J. Jacob, D. Katz, A. Lazzarini, K. Blackburn, S. Koranda, "The Pegasus Portal: Web Based grid Computing", In proceedings of the 20th Annual ACM Symposium on Applied Computing, Santa Fe, New Mexico, March 13 -17, 2005
- [8] H. Newman, J. Bunn, I. Legrand, S. Low, D. Nae, S. Ravot, C. Steenberg, X. Su, M. Thomas, F. van Lingen, Y. Xia, R. Cavanaugh, S. McKee, "The UltraLight project: The Network as an Integrated and Managed Resource for Data Intensive Science", in Computing In Science and Engineering, Issue on grid computing, 2005.
- [9] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation" in proceedings of Scientific and Statistical Database Management conference, 2002.
- [10] F. Wuerthwein, "Me - My Friends - The grid", OSG Internal document #128, May 2005 (<http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=128>)
- [11] G. Carcassi, T. Carter, Z. Liu, G. Smith, J. Spiletic, T. Wlodek, D. Yu, X. Zhao, "A Scalable grid User Management System for Large Virtual Organization", In proceedings of Computing for High Energy Physics (CHEP), October 2004, paper 122, (<http://indico.cern.ch/contributionDisplay.py?contribId=122&sessionId=12&confId=0>)