Collaboration in Grid Environments using Clarens

Article

10 authors, including:

Frank van Lingen
Cisco Systems, Inc

53 PUBLICATIONS   5,288 CITATIONS

SEE PROFILE

Conrad Steenberg
TriNet

83 PUBLICATIONS   6,090 CITATIONS

SEE PROFILE

Ashiq Anjum
University of Derby

97 PUBLICATIONS   2,240 CITATIONS

SEE PROFILE

Tahir Azim
Stanford University

26 PUBLICATIONS   208 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project   Establishing Grid node at SEECS (NUST) fund for MS Thesis View project

Project   CERN: Object-Relational Transformation View project

Collaboration in Grid Environments using Clarens

Frank van Lingen[1], Conrad Steenberg[1], Michael Thomas[1], Ashiq Anjum[2], Tahir Azim[2], Faisal Khan[2], Harvey Newman[1], Arshad Ali[2] , Julian Bunn[1], Iosif Legrand[1]

[1]California Institute of Technology, United States
Email:{fvlingen, julian.bunn@caltech.edu} {iosif.legrand@cern.ch} {newman, conrad, thomas@hep.caltech.edu}

[2]National University of Science and Technology, Pakistan
Email:{arshad.ali, ashiq.anjum, tahir, faisal.khan@niit.edu.pk}

Abstract

In 2007 the LHC experiments will start data taking, resulting in vast amounts of data being generated (tera bytes to peta bytes). These data are stored in globally distributed facilities. Hundreds of geographically dispersed groups of physicists will utilize these data for their analysis using a resource limited distributed system (limited in bandwidth, cpu, and storage). Other than for example astronomy, physics analysis resembles the "needle in the haystack" problem which results in unpredictable resource usage patterns as physics groups are using their analysis algorithms to sift through the data.

In order to facilitate this unpredictable use of distributed resources by geographically dispersed research groups, applications are being developed to enable collaboration, This article will give an overview of some of the components that have been developed using the Clarens Web Service framework.

1. INTRODUCTION

Scientific collaborations are becoming more and more geographically dispersed. Researchers from all over the world collaborate on new scientific discoveries and breakthroughs in many "big science" experiments such as the Virtual Observatory [50], the Large Hadron Collider (LHC) program [7], LIGO [8] and Nuclear fusion [9] . Not only do these experiments generate tera bytes to peta bytes of data. In many cases resources for analyzing and storing these large amounts of data are distributed on a national or international scale.

Some of the largest scientific collaborations today, such as CMS [10] and ATLAS [11] who are building experiments for CERN's LHC program, each encompass 2000 physicists from 150 institutions in more than 30 countries. Each of these collaborations include 300-400 physicists in the US, from more than 30 universities, as well as the major US HEP laboratories.

Realizing the scientific wealth of these science experiments, presents new problems in data access, processing, distribution, and collaboration across national and international networks, on a scale unprecedented in the history of science. One technology that holds the promise to form the basis of such an integrated, managed, distributed system are Web Services. The service oriented architecture (SOA) as proposed by [4] has been endorsed by many grid projects such as OSG [5] and EGEE [6].

The (soon to be started) DISUN[1] project and the Ultralight project [1] are two SOA based projects that are providing components to enable distributed physics analysis for a large user base. It is not only important that the system enables users to perform their data analysis, but it is equally important that they are able to share and publish their results, and are able to collaborate with other users. The authors argue that in order to enable collaboration between groups in a distributed environment based on Web Services some basic requirements need to be satisfied (this is not en exhaustive list):

- Authentication of users and secure access to data and Web Services.
- Virtual organizations and role management: users can have different roles in different groups.
- Access control on data: groups can share data while excluding others from reading/writing this data.
- Access control on Web Services: groups (or administrators) can restrict access to Web Services.
- Discovery of services, data, and software: geographically dispersed groups will install/move/remove Web Services and (analysis) software in an unpredictable manner. Other members of the groups need to be able to discover these services, applications and data.
- Automated tests of Web Services: within a global distributed service environment services need to be available 24/7. This can be achieved by offering the same services on multiple locations. However when a service does not respond this has to be identified as quickly as possible.
- User feedback when a problem occurs. Being able to diagnose what went wrong during a session.

To address the requirements mentioned, the Clarens project was started in 2001 [28] to provide a scalable Web Service framework for the development of distributed applications. Initially Clarens was developed as part of the CMS experiment, however as Web and Grid Services

---

[1] DISUN: Data Intensive Science University Network

became two of the de facto standards for development of distributed applications, Clarens became part of several projects: Ultralight [1], HotGrid [18], Monte Carlo Processing Service using RunJob [19], the physics shell project (PHYSH) [20], Lambda Station [21] project. IGUANA [22] and the PROOF [23] Enabled Analysis Center (PEAC). Development and deployment of Clarens is also part of several large Grid collaborations such as PPDG [24], IvdGL [25], Griphyn [27], OSG and Grid3 [26]. Clarens was also used in the winning SuperComputing 2003 bandwidth challenge (23 Gb/s peak), in which Clarens servers generated a peak of 3.2 Gb/s disk-to-disk streams consisting of CMS detector events.

Section two gives an overview of Clarens, a Web Service framework. Section three to eight discuss several services developed to enable and foster collaboration between users of the system as outlined in the requirements in this section. More information about Clarens can be found in [28], [29], [30], [31], [32] and, [33] Although the Clarens project focuses on the physics community much the functionality and many of the Web Services can be used by other scientific communities.

Within this paper we use the following definitions for Web Services and Web Service Framework: A Web Service is a component performing a task, most likely over a network. A Web Service can be identified by a URI and its public interfaces and bindings are described using WSDL. At the basis of a Web Service call (invocation) is a protocol (frequently, but not exclusively this is XML-RPC [12], or SOAP [13]). A Web Service Framework is an application that provides support for developing and deploying Web Services. Unless mentioned otherwise Clarens refers to both the Python and the Java implementation of Clarens. PClarens refers to the Python implementation and JClarens to the Java implementation.
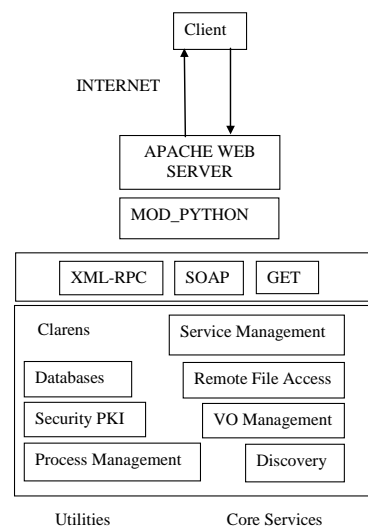
2. CLARENS

Clarens aims to provide the basis for a consistent, high-performance, fault tolerant system of distributed Web Services deployment and development. By leveraging existing, widely implemented standards and software components, including HTTP, SSL/TLS (RFC 2246) encryption and X509 (RFC 3280)[2] certificate-based authentication, and SOAP/XML RPC data serialization, Clarens also aims to be easily accessible to a wide variety of client implementations with the minimum of software dependencies. This approach lowers the barriers of entry to participate in the service network, re-use of existing developer skills, and a wide choice of development tools and languages.

In order to improve scalability, the PClarens server is implemented as an extension to the Apache Web Server [14] using the `mod_python` extension in the Python byte-code compiled language. PClarens itself is both

architecture and platform independent by virtue of using Python as an implementation language. Figure 1 shows the PClarens architecture. The Apache server receives an HTTP POST or GET request from the client, and invokes PClarens based on the form of the URL specified by the client (other URLs are handled transparently by the Apache server according to its configuration). Secure Sockets Layer (SSL) encrypted connections are handled transparently by the Apache server, with no special coding needed in PClarens itself to decrypt (encrypt) requests (responses).

After the request has been processed, a response is sent back to the client, which is usually encoded as an RPC response, but may also be in the form of binary data. GET requests return a file or an XML-encoded error message to the client, while XML-RPC or SOAP encoded POST requests return a similarly encoded response error message

In response to a preference for developing not only Python based web services but also Java based Web Services, a second Java based Web Service framework has been implemented (JClarens). The Java language and runtime environment have several desirable characteristics, including implementations on several platforms, a large developer community, and mature Web Service development tools.

Figure 1. PClarens Architecture

The JClarens implementation is based on so-called servlets implemented inside a commodity container, in this case the open source Apache Tomcat server [15] For JClarens the Tomcat server replaces the Apache web server and `mod_python` module in the architecture as depicted in Figure 1.

The architecture discussed in this section can give the impression that Clarens (both the Python and Java implementation) is very similar to for example a Tomcat server and an Apache AXIS module [16] (the Java implementation actually uses Tomcat and AXIS). Both systems can be called Web Servers that host Web

---

[2] For Internet Engineering Task Force Request For Comment (RFC) documents, see `http://www.ietf.org/`

Services. The difference is that the Clarens Web Service framework address issues such as:

- Certificate based authentication when establishing a connection.
- Access control on Web Services.
- Remote file access (and access control on files).
- Discovery of services and software.
- Proxy management.
- Shell access based on certificates.
- Virtual Organization management.
- Multiple protocols (XML-RPC, SOAP, Java RMI (only for JClarens), JSON-RPC [17]).

The next sections will discuss several of Clarens based services in detail. Other services and functionality discussed in the requirements in section one such as remote file access, access control lists, and service discovery, have been discussed in [33].

3. VIRTUAL ORGANIZATION AND ROLE MANAGEMENT

Virtual organization management allows (geographically dispersed) users in large collaborations to be grouped together. Using this group structure it is easier for administrators of Grid resources to create fine grained access control lists for different groups and sub groups of scientists.

Each Clarens server instance manages a tree-like Virtual Organization (VO) structure, as shown in Figure 2, rooted in a list of administrators. This group, named admins, is populated statically from values provided in the server configuration file on each server restart. The list of group members is cached in a database, as is all VO information. The admins group is authorized to create and delete groups at all levels.

Each group consists of two lists of distinguished names (DNs), for the group members and administrators respectively. Group administrators are authorized to add and delete group members, as well as groups at lower levels. The group structure is hierarchical because group members of higher level groups are automatically members of lower level groups in the same branch. The example in Figure 2 demonstrates the top-level groups A, B, and C with second level groups A.1, A.2, and A.3

A further optimization, the hierarchical information in the DNs may also be used to define membership, so that only the initial significant part of the DN need to be specified in defining members of a group. DNs are structured to include information on the country (C), state/province (ST), locality/city (L), organization (O), organizational unit (OU), common name (CN), and (Email). An example DN issued by the DOE Science Grid CA for individuals is:
/O=doesciencegrid.org/OU=People/CN=John Smith 12345

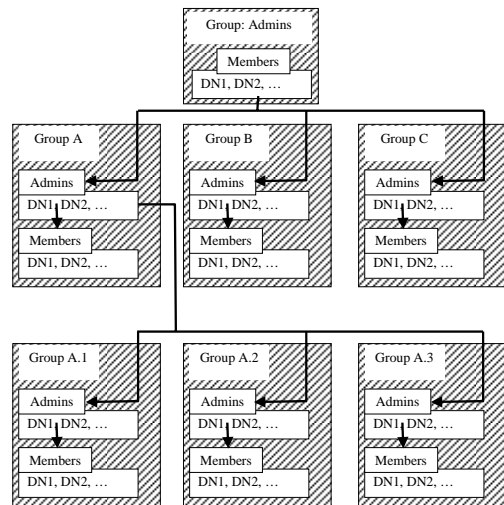**Figure 2. Clarens Group Diagram**

For servers a DN could look like:
/O=doesciencegrid.org/OU=Services/CN=host/www.mysite.edu for servers.

To add all individuals to a particular group, only /O=doesciencegrid.org/OU=People need to be specified as member DN

In many Grid projects virtual organization structures are not more than one or two levels deep. OSG, CMS, ATLAS virtual organizations are examples this. The hierarchical group structure within the Clarens group service enables these VO's to define roles within their VO's. For example within the CMS VO we can identify several roles. E.g. : detector physics (user doing analysis for detector studies), analysis physics (user doing analysis on event data) US grid-operator (persons responsible for data transfer in the US), EU grid-operator (person responsible for data transfer in Europe). Within the Clarens group service the following groups can describe these roles:

- CMS.physics.detector
- CMS.physics.analysis
- CMS.grid-operator.US
- CMS.grid-operator.EU

Persons that have multiple roles will have their DN associated to multiple groups. The Clarens group service offers a flexible and extendible structure for defining VO's and roles within a VO.

4. DISTRIBUTED TESTING AND MONITORING

The two implementations (Java and Python) of Clarens are used within several projects. In these projects several Clarens servers are running on several locations. Within

the Ultralight [1] testbed there are at least 15 servers running a variety of services. It is vital that servers and services are fully functional all of the time (if possible) and that downtime is minimized. Within (large) collaboration services and servers are administrated by different (geographically dispersed) groups/persons. It is therefore important that these services can be periodically validated without human intervention and when a service fails a test a person responsible for that service gets notified.
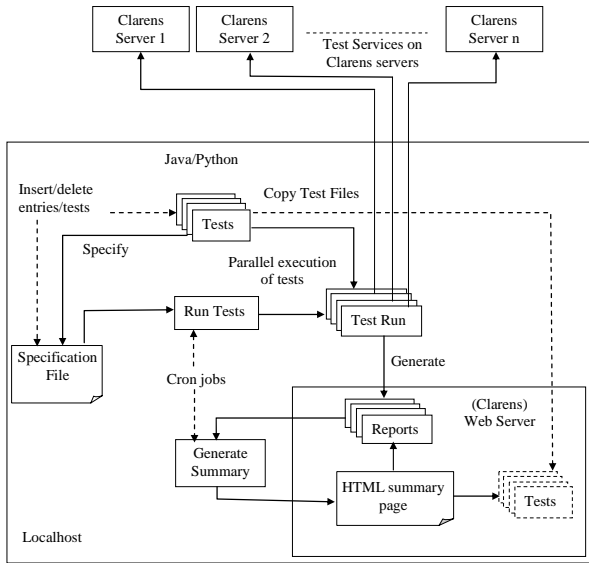
Figure 3. Clarens Distributed Testing Framework.

Figure 3 shows an overview of the components of the Clarens test environment. An administrator can define groups of tests, servers and email addresses in specification file. A `run tests` application parses this file periodically and spawns off separate test runs (one for each server). Each of these runs generates a status report. Periodically a summary is generated. The reports, summary and the tests itself are published on a web server. If a test failed emails are send to the administrators of that service. Users can browse the summary page and analyze the test results and tests. Within Clarens the testing framework also serves to identify compatibility issues between Java/Python clients and Java/Python servers. Multiple sites can use the test framework to prevent a single point of failure for testing services.

5. LOGGING SERVICE

Large scientific collaborations will deploy numerous Web Services in a globally distributed system. In most cases users will access these Web Services through so-called portals. These portals hide much of the Web Service complexity behind an easy to use (web based) interface. However when errors occur (generated by either a user or the system) it is important that users (and administrators) get the appropriate feedback to diagnose the problem to identify which group within the collaboration is responsible and has the authority to resolve the problem.

The base PClarens server contains support for a request and response logging facility. The logging service allows users to access this facility remotely. The Clarens server logs Web Service requests and responses in various levels of detail, namely: 1) no logging 2) request method name, call time, calling user DN, and return status, and lastly 3) the complete request/response text in addition to the information of level 2. A default minimum logging level can be specified by the server administrator for debugging or security purposes. The logging service API contains methods to set and get the log level for a particular user session: `logging.get_log_level`, and `logging.set_log_level`. Additionally, the logging service allows users to create named logging sessions to organize the storage and retrieval of logged information for later use. These sessions can be listed, accessed and deleted by their owner. Currently access control is implemented such that logging data can only be access by the server administrators and the original calling user. In future access control lists may be added to allow users to share sessions with other users on a selective basis. Logging information can be accessed using various criteria, including access times, session names, method names, or return status. A bowser-based interface is available to allow zero-install access to the service by remote users.
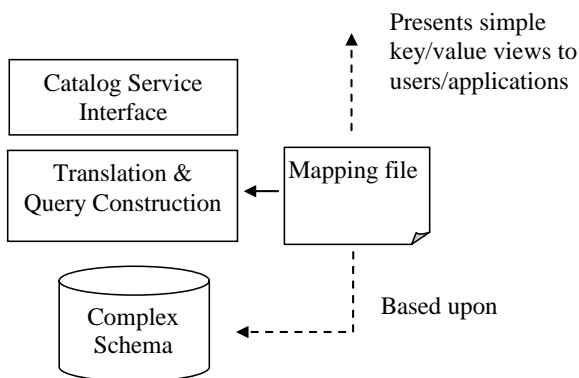
6. CATALOG SERVICE

As discussed in [33], the Clarens file service enables secure and access controlled access to (groups of) files and directories. In many cases however, physicists want to access and discover data on a higher conceptual level. When collaborating with colleagues, scientists can refer to data as for example "dataset Muon342_Run2" without referring to a specific location or to the numerous files representing this dataset. For example within the CMS experiment there is the notion of datasets. A dataset consists of collections, which in turn consists of logical file names. Each logical filename (lfn) is associated to several physical file names (pfn), representing replicas. Within CMS several catalogs (refdb [41], pubdb, phedex [42], pool file catalog [43]) contain information that describe this data hierarchy. Furthermore, these catalogs change overtime (schema evolution or designed from scratch). Whatever the database, there were several characteristics that are similar in all catalogs within CMS:

- Many users want to query information associated to data that can be represented as key values
- All catalogs are implemented as a SQL database.

The catalog service exploits these characteristics to provide a generic front end for SQL based databases. The service contains four generic methods which can be augmented with more specific methods, depending on the catalog.

The `catalog.getViews` method returns the different views a user can use in this catalog. In many

cases there will be only one view which the user does not have to specify. Examples of views can be 'dataset', 'collection', 'pfn', and 'lfn'. Once a user selects a view he/she can request the attributes that are associated to this view. The `catalog.getMetadataSpec` returns a list of meta data attributes and their formats which are the meta data keys attached to a view in a dictionary format. Formats supported are: string, integer, float, double, and xml (xml is represented as a string). The `catalog.getMetadataValues` method returns a sample of values for a particular attribute. This methods enables users to get an impression of what values are in a particular catalog and to tune their queries accordingly. Finally the `catalog.queryCatalog` method accepts simple queries ('and' and 'or' key value queries) and returns a set of key/values associated to a view.

Figure 4. Catalog Service Architecture

Figure 4 shows an overview of the catalog service. A mapping file describes a mapping between a (complex) SQL schema and key/value views that will be exposed to users/applications. A translation component converts user queries to database specific queries, and formats the result of these queries as key/value views. The only thing that needs to be changed for catalogs with different schemas is the mapping file. The advantage of such a service are:

- Shield users from database schema and changes in schema.
- Providing enough flexibility for users to query the data, and return data as key values.
- Allow users to discover the simplified schema (keys) and browse through associated values.
- Provide a uniform interface for different catalogs (which are represented by a relational database).

Using the Clarens discovery service, users can discover certain types of catalogs and query these catalogs in a dynamic service environment. Although the catalog service provides a generic frontend to SQL database, it is also restricted, in querying these datatabases, especially if these databases contain a complex schema. However it was recognized that most of the time users want to query

for key/values associated to a data entity, which is what the catalog service addresses.

7. CONFMON SERVICE

In many instances within a distributed system, information is ordered in a hierarchy. Examples of such hierarchies are: the Glue schema [44], the Clarens service hierarchy (server/service/method), Clarens VO management (see section 3), Clarens access control lists (see [33]), software discovery service (server/application/version) (see section 8), File system, etc.... Also in many instances users (or groups of users) want to organize and share their (meta) data in a hierarchical structure with groups they collaborate with (and perhaps hide it for other groups). The confmon service provides a tree structure to which users can attach key/values pairs, containing time stamps, an expiration date and access control on the tree nodes. The access control methods are based on the access control methods of the file service (see [32]). The argument for most of the (non ACL) confmon methods is a struct that consists of a list of endpoints (server and protocol), a provider (specified as a DN), a category (representing the node in the tree), a set of key/values and a duration (0 means it does not expire). Besides the access control methods the confmon service exposes five methods. `confmon.register` registers information in the hierarchy. `confmon.append` appends additional data to existing key/values (a key can have multiple values). `confmon.deregister` will delete entries that match the deregister pattern given as input. In order to renew the expiration time on data the `confmon.expire` method can be used. If the duration period 0 is used the data will never expire. `confmon.find` is used to query data in the hierarchy. The find method takes as input endpoint patterns, key/values, providers, and returns a list of entries that match this pattern. Results of all these methods are dependent on the read and write access of the user on the elements in the hierarchy.

The current implementation uses a MySQL database as backend, but future implementations will investigate hierarchical storage such as XML databases or LDAP [49]. The main design criteria for the confmon service have been to provide users with access controlled key/values hierarchies with limited functionality for inserting, deleting and browsing this hierarchy but flexible enough to enable users to share key/value hierarchies.

8. DYNAMIC SOFTWARE DISCOVERY

Many criteria can be used to select a site on which to run user analysis jobs. One such a criteria is the type and version of a particular piece of software needed for this job. User analysis jobs (within the physics community) require (not infrequent) a specific version and type of certain analysis packages and libraries. Certain sites have restrictive software installation policies, or it can take a long time to install a required package. Often sites will

not install all possible versions of all possible packages. Within a global distributed environment software packages will be installed, (re)moved and updated. It is virtually impossible for scientists in large collaborations, and applications to keep track of these changes. Users and services need to be able to discover in real time what packages and what versions are installed on the sites such that can be decided where to run a particular analysis.
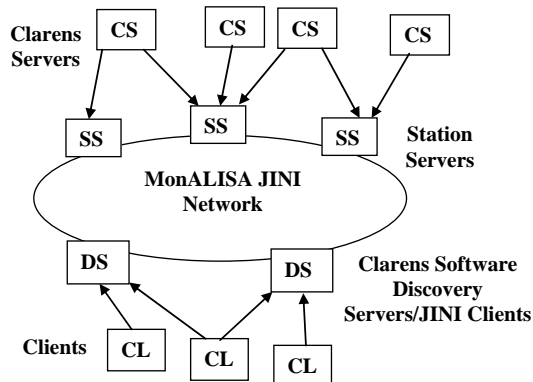
Figure 5. The MonALISA-based Software Discovery architecture.

The software discovery service is based on the Web Service discovery [33]. Registration with the Software Discovery service must happen at regular intervals in order to prove that a particular package is still available. If a site fails to notify the software discovery service within a certain time period, it is automatically removed from the registry. The `register` method is used to add a new software package to the registry while a `find` method is used to locate software instances that match certain search criteria such as name, location, version. The `deregister` method is used to remove software packages from the registry (however is seldom used since the registry will automatically remove the software package once it fails to re-register).

The Software Discovery service follows a peer-to-peer model based on JINI [46] and the MonALISA [45] Grid monitoring system, as shown in Figure 5. MonALISA is a JINI-based monitoring system that uses station servers to collect local monitoring data, and uses a JINI peer-to-peer network to share selected monitoring data with other station servers or other interested clients. Arbitrary monitoring data can be published to a MonALISA station server using a UDP based application. Each software discovery service contains a client that listens for software publications messages on the MonALISA JINI network, and stores them in an in-memory cache. The software discovery service periodically purges expired entries from this in-memory cache. Since the software registry is stored in memory, it is not persistent across server restarts, which does not pose a problem since the registry will be rapidly populated with new information once it restarts. The current refresh rate for software discovery information is between 10 to 30 seconds. The

initial version of the Software Discovery services uses SCRAM [47][3] for local site software discovery and publishes the results into the MonALISA network. Future versions will also include plugins to discover VDT [48] based software applications.

9. FUTURE WORK

Future work will focus (amongst others) on: message based protocols, mass storage integration, improved service discovery functionality.

The current Clarens Web Service implementation was designed for a request response mode of operation, making it ill-suited for the type of asynchronous bi-directional communication required for interactions between users and the jobs they are running on private networks protected by network address translation (NAT) and firewalls. An instant messaging (IM) architecture provides the possibility to overcome this limitation. Since messages can be sent and received by jobs asynchronously, jobs can be instrumented to act as Clarens servers, or clients sending information to monitoring systems or remote debugging tools.

Although Clarens provides remote file access through a web service, it does not support interfaces to mass storage facilities yet. Work is under way to provide an SRM service interface [38] to dCache[39] such that Clarens can support robust file transfer between different mass storage facilities.

Work is underway to provide interoperability between the Clarens discovery service and Globus MDS [40] such that both systems can publish/retrieve information in the other system. Other activities include collaboration with the EGEE project on a common discovery interface.

When scientists collaborate they not only share data but also share/develop sequences of commands and analysis applications with each other. The CODESH project [51] addresses distributed collaboration. At the time of writing CODESH functionality is being integrated into Clarens, such CODESH can be used within a secure access controlled distributed environment in conjunction with other Clarens functionality such as remote file access.

10. RELATED WORK

Several other Web Service frameworks have been developed in the last couple of years. The last versions of Globus [34] have been service oriented frameworks based on the open grid service architecture (OGSA). Although Globus offers secure and authenticated access using the concept of grid map files, it has a much coarser authentication and access control granularity than the Clarens ACL and VO management.

Ibm Websphere [35] is a commercial product that enables you to develop and deploy web services and is therefore not a desirable candidate to be deployed in large science collaborations that rely on open source.

---

[3] SCRAM is a software configuration environment used in the LHC experiments including CMS and ATLAS.

Glite [36] is a Perl based service framework based on Alien [37] that is used by the EGEE project [6] to develop web services. Several of the interfaces developed for services in this project are similar (but not the same) to Clarens service interfaces. The first versions of the Glite framework were based on Alien which initially was not designed to be a generic service oriented framework, while Clarens was. Substantial work has been carried out however to make Glite, less dependent of Alien.

The Globus information and discovery system (MDS) [40] is distributed in nature and provides much more functionality than the confmon service discussed in this paper.

The catalog service functionality described in section six is not novel by itself. In many data integration projects wrappers have been developed to shield applications/users from underlying database complexity. However this catalog service is embedded in a distributed Web Service environment that facilitates access control, authorized access and service discovery.

11. SUMMARY

The Clarens Web Service framework is gaining acceptance in the science community to support the development of a scalable distributed service environment. Several of the projects have chosen Clarens as it offers a good service response performance and yet provides powerful features such as ACL and VO management, service discovery, and remote file access. These features enable the creation of services that enable collaboration on a user and administrator level for large scientific communities.

Clarens provides a growing functionality for distributed analysis in a Grid-based environment, coupled with a set of useful client implementations for physics analysis. The projects that utilize Clarens also provide valuable feedback, which enable the Clarens team to enhance and improve the core functionality of Clarens and reuse components across projects that focus on scientific analysis.

12. ACKNOWLEDGEMENTS

REFERENCES

[1] UltraLight Collaboration "UltraLight: An Ultrascale Information System for Data Intensive Research" proposal Submitted to NSF MPS/Physics: "ITR" February 2004. See also: http://ultralight.caltech.edu/portal

[2] J. Bunn and H. Newman "*Data Intensive Grids for High Energy Physics*", in "Grid Computing: Making the Global Infrastructure a Reality", edited by Fran Berman, Geoffrey Fox and Tony Hey, March 2003 by Wiley.

[3] F. van Lingen, J. Bunn, I. Legrand, H. Newman, C. Steenberg, M. Thomas, P. Avery, D. Bourilkov, R. Cavanaugh, L. Chitnis, M. Kulkarni, J. Uk In, A. Anjum, T. Azim "Grid Enabled Analysis: Architecture, Prototype and Status" CHEP 2004 Interlaken

[4] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Global Grid Forum, June 22, 2002

[5] Open Science Grid, http://www.opensciencegrid.org/

[6] E. Laure, F. Hemmer, F. Prelz, S. Beco, S. Fisher, M. Livny, L. Guy, M. Barroso, P. Buncic, P. Kunszt, A. Di Meglio, A. Aimar, A. Edlund, D. Groep, F. Pacini, M. SGaravatto, O. Mulmo, " Middleware for the next generation Grid infrastructure", In proceedings of CHEP, Interlaken, Switzerland 2004

[7] LHC Project, http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/

[8] LIGO, http://www.ligo.caltech.edu/

[9] Join European Torus, http://www.jet.efda.org/

[10] The Compact Muon Solenoid Technical Proposal, CERN/LHCC 94-38 (1994) and CERN LHCC-P1; see also: http://cmsdoc.cern.ch/

[11] The ATLAS Technical Proposal, CERN/LHCC 94-43 (1994) and CERN LHCC-P2; see also: http://atlasinfo.cern.ch/ATLAS/TP/NEW/HTML/tp9new/tp9.html

[12] XML Remote Procedure Call Website, http://www.xmlrpc.com

[13] Simple Object Access Protocol, W3 Consortium, http://www.w3.org/2002/ws/

[14] Apache Web Server, Apache Software Foundation, http://www.apache.org

[15] The Tomcat Servlet Engine, http://tomcat.apache.org

[16] AXIS, http://ws.apache.org/axis/

[17] JSON RPC, http://oss.metaparadigm.com/jsonrpc/index.html

[18] R. Williams, C. Steenberg, J. Bunn, " HotGrid: Graduated Access to Grid-based Science Gateways", In Proceedings of IEEE Supercomputing Conference, Pittsburgh USA, 2004

[19] P. Love, I. Bertram, D. Evans, G. Graham, "Cross Experiment Workflow Management: The Runjob Project", In proceedings of CHEP, Interlaken Switzerland 2004

[20] Physh, http://cmsdoc.cern.ch/cms/aprom/physh/

[21] Lambda station, http://www.lambdastation.org/

[22] I. Osborne, S. Muzaffar, L. Taylor, L. Tuura, G. Alverson, G. Eulisse, "IGUANA Interactive Graphics Project: Recent Developments", In proceedings of CHEP 2004, Interlaken

[23] M. Ballintijn, "Global Distributed Parallel Analysis using PROOF and AliEn", In Proceedings of CHEP 2004 Interlaken

[24] Particle Physics Data Grid, http://www.ppdg.net/

[25] International Virtual Data grid laboratory, http://www.ivdgl.org/

[26] Grid3, http://www.ivdgl.org/grid2003/

[27] Grid Physics Network, http://www.griphyn.org/
[28] C. Steenberg, J. Bunn, T.Hickey, K. Holtman, I. Legrand, V. Litvin, H. Newman, A. Samar, S. Singh, R. Wilkinson (for the CMS Collaboration), "Prototype for a Generic Thin-Client Remote Analysis Environment for CMS" Proceedings of CHEP, paper 3-044, p. 186, H.S. Chen (ed.), Beijing China, 2001
[29] C. Steenberg, J. Bunn, I. Legrand, H. Newman, M. Thomas, F. van Lingen, A. Anjum, T. Azim "The Clarens Grid-enabled Web Services Framework: Services and Implementation" In Proceedings of CHEP, Interlaken Switzerland 2004
[30] A. Ali, A. Anjum, R. Haider, T. Azim, W. ur Rehman, J. Bunn, H. Newman, M. Thomas, C. Steenberg. "JClarens: A Java Based Interactive Physics Analysis Environment for Data Intensive Applications" in the Proceedings of ICWS, the International Conference of Web Services, San Diego, USA 2004
[31] Clarens homepage, http://clarens.sourceforge.net
[32] C. Steenberg, E. Aslakson, J. Bunn, H. Newman, M. Thomas, F. van Lingen "*Clarens Client and Server Applications*" CHEP 2003 La Jolla California
[33] 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*" , Workshop on Web and Grid Services for Scientific Data Analysis (WAGSSDA), Oslo, June 14-17, 2005
[34] Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit" Intl. J. Supercomputer Applications, 11(2):115-128, 1997
[35] Websphere, http://www.websphere.org/
[36] M. Lamanna, B. Koblitz, T. Chen, W. Ueng, J. Herrala, D. Liko, A. Maier, J. Moscicki, A. Peters, F. Orellana, V. Pose, A. Demichev, D. Feichtinger, "Experiences with the gLite Grid Middleware" In proceedings of CHEP, Interlaken Switzerland, 2004. see also: http://glite.web.cern.ch/glite/
[37] P.Buncic, A.J. Peters, P.Saiz "The AliEn System, status and perspectives",In proceedings of CHEP, La Jolla, California 2003
[38] A. Shoshani, A. Sim, J. Gu, "Storage Resource Managers: Middleware Components for Grid Storage", In proceedings of Mass Storage Systems conference, Maryland USA 2002
[39] P. Fuhrmann, "dCache the commodity cache", In proceedings of the Twelfth NASA Goddard and Twenty First IEEE Conference on Mass Storage Systems and Technologies, Washington DC 2004
[40] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid Information Services for Distributed Resource Sharing", Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001
[41] V. Lefebure, J. Andreeva, "RefDB: The Reference Database for CMS Monte Carlo Production" In Proceedings of CHEP La Jolla, California, 2003
[42] T. Barras, A. Afaq, W. Jank, O. Maroney, S. Metson, D. Newbold, K. Rabbertz, J. Rehn, L. Tuura, T. Wildish, Y. Wu, C. Grandi, D. Bonacorsi, C. Charlot, M. Ernst, A. Fanfani, I. Fisk, "*Software agents in data and workflow management*", In Proceedings of CHEP, Interlaken Switzerland 2004
[43] I. Papadopoulos, "POOL, the LCG Persistency Framework", In Proceedings of IEEE Nuclear Science Symposium, Portland, Oregon, 2003
[44] GLUE, http://www.cnaf.infn.it/~sergio/datatag/glue/index.htm
[45] I. Legrand, "MonALISA - MONitoring Agents using a Large Integrated Service Architecure" International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Tsukuba, Japan, December 2003
[46] JINI, http://www.sun.com/software/jini/
[47] J.P. Wellisch, C. Williams, S. Ashby ," SCRAM: Software configuration and management for the LHC Computing Grid project", CHEP , La Jolla, California, 2003
[48] VDT, http://www.cs.wisc.edu/vdt/
[49] M. Smith, T. Howes', "LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol" New Riders Publishing, Hardcover, March 1997 ISBN 1578700000
[50] National Virtual Observatory http://www.us-vo.org
[51] D. Bourilkov, "The CAVES Project - Collaborative Analysis Versioning Environment System: The CODESH Project - Collaborative Development Shell," arXiv:physics/0410226