GRID ENABLED ANALYSIS: ARCHITECTURE, PROTOYPE AND STATUS

Frank van Lingen, Julian Bunn, Iosif Legrand, Harvey Newman, Conrad Steenberg, Michael Thomas,
California Institute of Technology, United States
Paul Avery, Dimitri Bourilkov, Rick Cavanaugh, Luakik Chitnis, Mandar Kulkarni, Jang Uk In,
University of Florida, United States
Ashiq Anjum, Tahir Azim,
National University of Science and Technology, Pakistan

Abstract

The Grid Analysis Environment (GAE), which is a continuation of the CAIGEE project [5], is an effort to develop, integrate and deploy a system for distributed analysis. The current focus within the GAE is on the CMS experiment [1] however the GAE design abstracts from any specific scientific experiment and focuses on scientific analysis in general. The GAE project does not intend to reinvent services, but rather to integrate existing services into a collaborative system of web services.

INTRODUCTION

As grid middleware matures and grid (web) services become more prolific, the development of "higher level" services that can take intelligent decisions based on these "lower level" services is required. In 2003, Caltech and the University of Florida initiated the Grid Analysis Environment (GAE) project with the goal of developing, integrating, and deploying a web service framework and services for physics analysis (especially CMS [1]) at the LHC. The project's kernel is a Grid-based portal called Clarens [4]. GAE is a continuation of the CAIGEE project [5].

The GAE will be used by a large, diverse community. It will need to support hundreds, even thousands, of analysis tasks with widely varying requirements. It will need to employ priority schemes, and robust authentication and security mechanisms. Most challenging, the GAE will need to operate well in what we expect to be a severely resource limited global computing system. The GAE is where the critical physics analysis gets done, where the Grid end-to-end services are exposed to a very demanding physicist clientele, who will have to learn how to collaborate at large distances on challenging analysis topics.

COMPUTING MODEL REVISTED

The LHC experiments' computing models [2] are based on a hierarchical organization of Tier0, multiple Tier1 and multiple Tier2, centres. Most of the data will "flow" from the Tier 0, to the Tier 1s and then on to the Tier2s and downstream. The Tier0 and Tier1 centres are powerful in terms of CPU power, data storage, and bandwidth. This hierarchical model is such that the sum of resources below a certain Tier-N are of approximately the same amount as that located at the Tier-N. Data is organized such that the institutes (represented by tiers) will be (in most cases) physically close to the data they want to analyze. Trends indicate that the aggregate power data storage capacity of Tier2 relative to Tier1 is likely to be greater than foreseen, and that Tier2 and 3 systems taken together will be a substantial source of simulated and perhaps also reconstructed event data.

The hierarchical model is the basis on which data will be distributed. However, the unpredictable patterns of physics analysis as a whole will lead to data and jobs being moved around between different peers in an essentially chaotic fashion. Although it is possible to make predictions on how best to organize data, the unpredictable, chaotic analysis patterns tend to defeat simple rationalization. Depending on how "hot" (popular) or "cold" (infrequently requested) data is, attention will shift between the different data sets. Furthermore, multiple geographically distant users might be interested in data that is geographically dispersed among the different tiers. Because of this, data will need to be replicated. And so it will not always be possible to move data around so that it fits into a strict hierarchical model.

The data movements and job movements outside the hierarchical model, although relative small compared to the data movement within the hierarchical model, will be significant consumers of resources (network, CPU, storage). When users submit jobs to a peer, middleware will discover which resources are available on other peers that can satisfy the job requests. Although a large number of job requests will fit with the hierarchical model, others will not. The more powerful peers (super peers) will receive more job and data requests, and will host a wider variety of services.

ARCHITECTURE

The first step within the GAE project was to identify key services needed to perform distributed analysis and the interaction between these services. This service design is described in detail in [8] and displayed in

Figure **1**: A client discovers catalogs and workflow management services. After selecting a dataset (or multiple datasets), the user submits its job to a workflow management service which sends the different jobs (with different execute dependencies) to a grid scheduler. Based on the status of the different computing farms and the quota the user has it will submit the jobs to particular farms through a local execution service, which will interact with a replica management service. During this process and execution of the job a steering service keeps

track of the job status and gives periodically feedback to the user. If necessary the user (or the steering service on behalf of the user) can decide to stop, move, or pause, the job. Once the job has finished the data collection service notifies the client and if necessary collects the results to a storage location accessible by the user. Some of the key services that have been identified are well known within the Grid community. GAE does therefore not only focus on new development but also on integration of existing components.

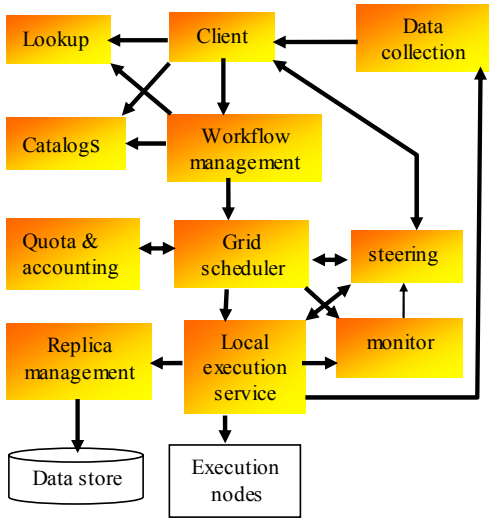

Figure 1. Architecture overview

DEVELOPMENT

The second step consisted of mapping existing applications that contain the functionality described in the GAE design to these services (see [8]). As the focus of the GAE is the CMS experiment this puts restrictions on what kind of applications we use within our deployment. For example within CMS, POOL catalog [7], and RefDB [10] are used as catalogs. Furthermore, MCRunJob [9] and RefDB contain functionality that can be mapped to a workflow management service. Integration and exposure of current CMS applications as web services is important within the GAE deployment for CMS. The approach taken by the GAE team, for these applications is bottom up: Analyze the applications and develop (if possible) a "neutral" web service interface for these applications. Furthermore, publish these interfaces in a language neutral format (e.g. WSDL) such that users/applications can access these services even when they are not completely compliant with "standard" interfaces.

For development and deployment the Clarens [4] web service framework is used, which is a high-performance wide-area network system for web service deployment that includes powerful features for managing access control to web services, and dynamic service discovery. The choice for Clarens as the GAE backbone does not prevent services within the GAE to access other web service outside the Clarens environment e.g. Globus Toolkit [12], as both are based on XML-RPC and the SOAP protocol. The Clarens web service framework is available as a Python and a Java implementation. At the time of writing there were approximately 20 known deployments of Clarens: Caltech (5), University of Florida (4), Fermilab (3), CERN (3), Pakistan (2+2), INFN (1).

DEPLOYMENT

This section discusses several scenarios that have been successfully executed on a set of distributed hosts on which we deployed the Clarens server and GAE services. The scenarios are a preparation for the end to end analysis scenarios that allow multiple users to perform physics analysis.

Figure 2 shows the use case of a client that accesses several catalogs and queries these catalogs to identify datasets of interest to be used for physics analysis. The client code contains no “hard coded” urls to services except for several entry points to discovery services. Step 1 deals with locating the POOL catalog and Refdb service. Multiple catalogs are deployed in the distributed environment, containing potentially information on different datasets. Once the client receives the service urls from the discovery service it queries the multiple catalogs using asynchronous calls. Based on several queries it selects one or more datasets and uses a grid scheduler to submit an analysis job.

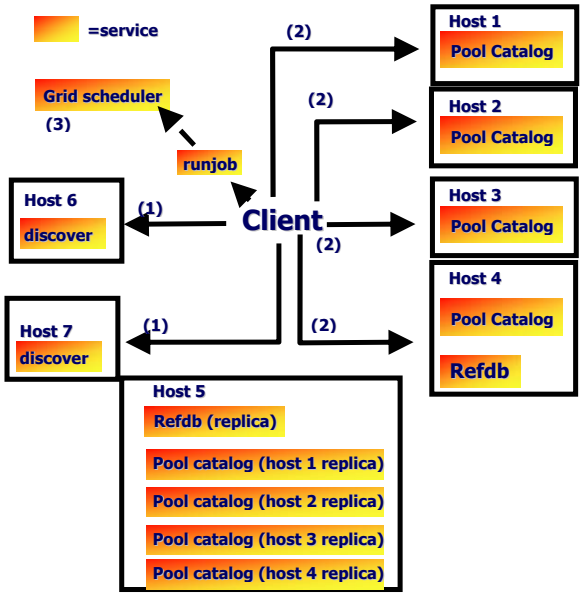

Figure 2. Distributed Querying for Datasets

Figure 3 shows the use case of jobs being scheduled to available resources using a push model. The current Sphinx [11] implementation is able to analyze monitor information from MonALISA [3] (step 2), after which the scheduler determines on which farm to run the job (step 3). On each farm BOSS [13] is deployed to provide a uniform interface to the local schedulers and to keep track

of the progress of the different jobs. Currently Sphinx job submission to the uniform BOSS interface is being implemented.

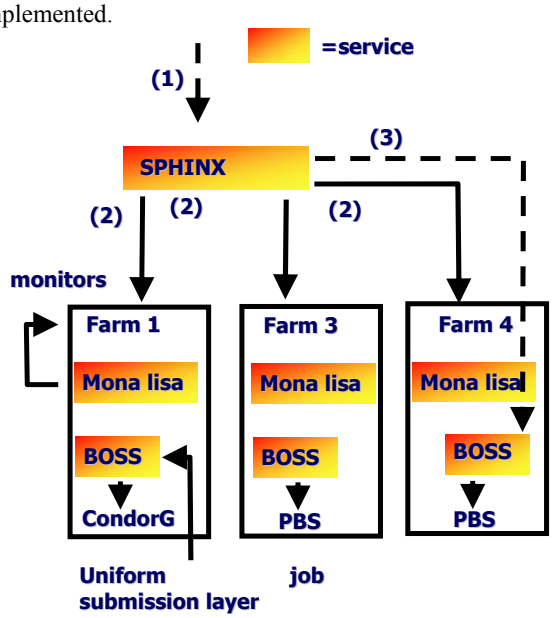

Figure 3. Scheduling: Push Model

Figure 4 shows a prototype framework that is being developed for grid enabled analysis within the CDF experiment. A client discovers (step 1) a global manager. Currently several global manager are deployed to decrease the load on a single global manager. The client submits a request to the global manager (step 2). This request contains a (or multiple) data sets a user wants to analyze and a specification of how much resources this analysis potentially needs. The global manager queries the CDF catalog (represented by the CDF SAM system) to get a list of farms where these datasets resides (step 3 and 4). The global manager will then request these local managers (step 5) for available resources to do the analysis. Once the global manager receives a list of resources available, it determines on which farm to run the analysis. The global manager contacts the associated local manager (step 7), who will start a data transfer using a Clarens dcache service. The client receives a unique session id and url (step 7) of the local manager it needs to submit its job to. The job submission service periodically checks if the dcache service has moved the data and if ready will start the analysis job (step 9). During execution analysis jobs will submit a life sign to the associated local manager (step 10). The local manager checks if any jobs failed to submit life signs and if so (within a certain interval) will terminate the job. The local manager submits monitor data on the number of jobs running, terminated, and finished, to the MonALISA framework.

At the time of writing, the service framework has been developed within the Clarens service framework [4]. The framework has been successfully deployed on 4 hosts. Current work involves linking the catalog and dcache service to the CDF SAM database and the dcache applications on different sites. Other work involves clients being able to submit Proof jobs (parallel root [6]).

Although the work for CDF (Figure 4) started after the GAE architecture document was completed and semantics of the services used are different there is conceptual overlap with the components described in the GAE architecture [8] and deployment of CMS components (Figure 2,Figure 3). The global manager can be compared with the global scheduler (SPHINX) and the local manager can be compared with the CMS BOSS service. It also shows that although it is possible to identify commonalities between the CMS and CDF deployment, there are also differences in for example interface and functionality. It is therefore not only important to design standard interfaces and implementations but also to work closely with the different physics experiments to construct and deploy the proper services for different experiments.

Within the Clarens framework service interfaces can be published in multiple ways (name, WSDL) via the Clarens grid portal creating a transparent environment not only for developers but also for users to discuss the different interfaces used within different analysis environments.

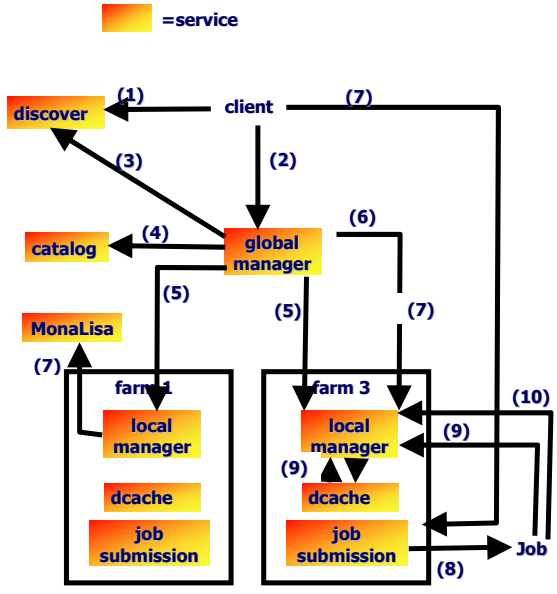

Figure 4. Scalable Job Submission

FUTURE WORK

Future work of the GAE project will focus on developing "high level" services. High level services can be described as services that take input from the "low level" passive services and proactively take decisions on where to perform an analysis job, when to replicate data, etc… Part of this phase is the development of an accounting service that allows for "fair" sharing of resources. The accounting service, together with other high level services prevent scenarios in which one user allocate all resources for a very long time (without the organizations consent).

Additionally, monitoring the GAE as it is used is of crucial importance in order to make intelligent decisions on data distribution, job execution and data management. The MonALISA monitoring framework is capable of satisfying this need, and it has been deployed on many sites to monitor disk, and CPU usage. In the future it will be able to perform complex trend analysis on GAE resource access patterns, and so eventually enable automatic resource co-scheduling.

CONCLUSIONS

The current GAE services that have been developed and deployed within the testbed together with clients allow execution of certain physics scenarios as described in the deployment section. It also raised a set of issues that are important for distributed computing: graceful handling of exceptions are very important within a distributed environment in which you can have complex service compositions, and you potentially are not aware which service instance you are using. Furthermore, time out when accessing a service (or a collection of services) is important to prevent the scenario of a service composition that takes a long time to execute because one or several services in this composition do not respond. Having a discovery service allows GAE service developers to create location independent service compositions. Services only need an entry point into the distributed system that allows them to discover available services. A problem associated with discovery, are services that have the same (or similar) names and/or interfaces. Within GAE we use the concept of VOs to distinguish between such services but this does not prevent having two services with the same name in one VO (we assume that within one VO there is a convention to prevent this from happening).

In order for the GAE to be successful it is important to have a good synergy with the scientific experiments as to be aware what tools and applications they use, and to know how their analysis processes are organized. This specific domain knowledge (if necessary) can than be integrated into the GAE development and deployment. On the other side, experiments have sometimes adapted specific (in house developed) applications to tackle parts of very "generic" problems, suitable for their application domain, such as workflow management, and (meta) data catalogs. These applications might need to be "translated" and adapted to the more generic syntax, structure and semantics as is used within the Grid environment. Such efforts require not only *technical engineering* but also *social engineering*, enabling GAE developers to offer Grid middleware applications that fit the needs of experiments.

ACKNOWLEDGEMENTS

This work is partly supported by the Department of Energy as part of the Particle Physics DataGrid project and by the National Science Foundation. 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] CMS: http://cmsdoc.cern.ch/cms.html

[2] J. Bunn, H. Newman, "*Data Intensive Grids for High Energy Physics*", p859-p906, in "Grid Computing", edited by F. Berman, G. Fox, A. Hey, 2003, Wiley

[3] Legrand, I., Newman, H., Galvez, P., Voicu, E., Cirstoiu, C., "*MonaLISA: A Distributed Monitoring Service Architecture*", Computing for High Energy Physics, La Jolla, California, 2003

[4] Steenberg, C., Aslakson, E., Bunn, J., Newman, H., Thomas, M., Van Lingen, F., *"The Clarens Web Service Architecture"*, Computing for High Energy Physics, La Jolla, California, 2003

[5] Newman, H. Branson, J. "*CMS Analysis: an Interactive Grid-Enabled Environment (CAIGEE)*" Submitted to the 2002 NSF Information and Technology Research Program Proposal #6116240

[6] Ballintijn, M., Brun, R., Rademakers, F., Roland, G., "*The PROOF Distributed Parallel Analysis Framework based on ROOT*", Computing in High Energy Physics, California 2003

[7] Duellman, D. "*POOL Project Overview*", Computing for High Energy Physics, La Jolla, California, 2003

[8] Bunn, J., Bouriklov, D., Cavanaugh, R., Legrand, I., Muhammad, A., Newman, H., Singh, S., Steenberg, C., Thomas, M., Van Lingen, F., "*A Grid Analysis Environment Service Architecture"* , http://ultralight.caltech.edu/gaeweb/gae_services.pdf

[9] Bertram, I., Evans, D., Graham, G.E., Love, P., Walker, R. "*McRunjob: A High Energy Physics Workflow Planner for Grid Production Processing*"Proceedings of UK e-Science All Hands Meeting 2003 2-4th September, Nottingham, UK

[10] Lefebure, V., Andreeva, J. "*RefDB: The Reference Database for CMS Monte Carlo Production*" Computing in High Energy Physics, La Jolla, California, 2003.

[11] In., J., Avery, P., Cavanaugh, R., Kulkarni, M., Ranka, S.,"*SPHINX: A Scheduling Middleware For Data Intensive Application on a Grid*," To appear in the proceedings of Computing in High Energy Physics, Interlaken, Switzerland, 2004

[12] Foster, I., Kesselman, C. "*Globus: A Metacomputing Infrastructure Toolkit*" Intl. J. Supercomputer Applications, 11(2):115-128, 1997

[13] Grandi, C., Renzi, A., "*Object Based System for Batch Job Submission and Monitoring (BOSS)"*, CMS note 2003/005

[14] DC04: http://www.uscms.org/s&c/dc04/