JOB MONITORING IN AN INTERACTIVE GRID ANALYSIS ENVIRONMENT

ArshadAli[4],Ashiq Anjum[4],Julian Bunn[1],Richard Cavanaugh[5],Frank van Lingen[1],Richard McClatchey[3],Harvey Newman[1],Waqas ur Rehman[4],Conrad Steenberg[1],Michael Thomas[1],Ian Willers[2]

[1]California Institute of Technology
Pasadena, CA 91125, USA
Email: {fvlingen,newman,conrad,thomas}@hep.caltech.edu, Julian.Bunn@caltech.edu
[2]CERN, Geneva, Switzerland
Email: Ian.Willers@cern.ch
[3]University of the West of England
Bristol, UK
Email: Richard.mcclatchey@uwe.ac.uk
[4]National University of Sciences and Technology
Rawalpindi, Pakistan
Email: {arshad.ali, ashiq.anjum,waqas.rehman}@niit.edu.pk
[5]University of South Florida, USA
Email: cavanaug@phys.ufl.edu

Abstract

The grid is emerging as a great computational resource but its dynamic behavior makes the Grid environment unpredictable. Systems and networks can fail, and the introduction of more users can result in resource starvation. Once a job has been submitted for execution on the grid, monitoring becomes essential for a user to see that the job is completed in an efficient way, and to detect any problems that occur while the job is running. In current environments once a user submits a job he loses direct control over the job and the system behaves like a batch system: the user submits the job and later gets a result back. The only information a user can obtain about a job is whether it is scheduled, running, cancelled or finished. Today users are becoming increasingly interested in such analysis grid environments in which they can check the progress of the job, obtain intermediate results, terminate the job based on the progress of job or intermediate results, steer the job to other nodes to achieve better performance and check the resources consumed by the job. In order to fulfill their requirements of interactivity a mechanism is needed that can provide the user with real time access to information about different attributes of a job. In this paper we present the design of a Job Monitoring Service, a web service that will provide interactive remote job monitoring by allowing users to access different attributes of a job once it has been submitted to the interactive Grid Analysis Environment [1].

INTRODUCTION

Grid computing provides a mechanism of sharing the heterogeneous computing and storage resources of organizations and individuals distributed across the globe to form a massive computing environment through which complex and large scale problems can be solved. The aim is to create an illusion of a large and powerful virtual computer with immense computing power. The great computing potential of grids has proven to be so significant that scientists working to solve many of the difficult scientific problems have started to utilize these systems to solve complex scientific problems. Physicists working with the Compact Muon Solenoid (CMS) at European Organization for Nuclear Research (CERN) are making use of computational and data grids to perform complex and time consuming data mining operations. Current grids tend to be used to perform unattended batch analysis jobs. The use of grids for performing interactive data analysis is a relatively new concept which is currently being investigated.

Interactive Grid Analysis Environment

Grids have emerged as a next generation computing and analysis platform. As the grid concept matures, people are moving towards interactive grid environments. Interactive grid environments will enable users to harness heterogeneous collections of computer components to form massive analysis environments while giving them full control over their jobs. Users in such an environment will be able to monitor their jobs while they are executing, to direct the execution of jobs to different nodes to increase the performance, and to communicate directly with the job while it is running. But in order to fulfill these requirement for interactive analysis certain applications and services are needed that can support job submissions, job scheduling, job monitoring and job steering. Fig. 1 describes the set of interacting web services that comprise one such analysis environment currently under development, the so-called interactive Grid Analysis Environment (GAE).

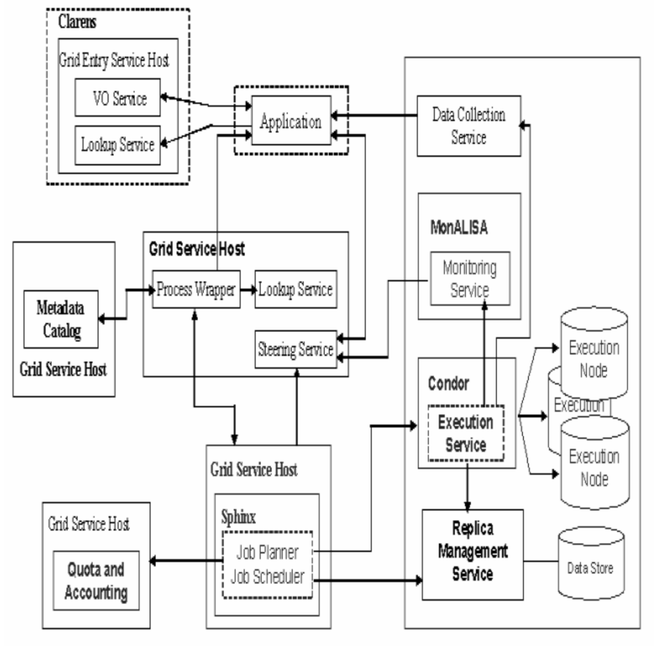

Figure 1: Grid Analysis Environment

The GAE focuses on the construction of an infrastructure that allows scientists to interactively perform the analysis and to submit small jobs in quick succession, depending on the output of previous jobs, instead of submitting one large batch job. Clearly it is not as simple to run interactive analysis on the Grid as it is on a single machine. The same data may be replicated in many locations, competition for resources may be much more complex, the number of higher-priority tasks that one owns may not be automatically known, and therefore the best choice of how and where to execute a task is hard to determine. For these reasons, new forms of Grid services that are able to make reasonable choices among a range of possible job-execution strategies, autonomously or interactively, are needed. Decisions made by these services will be based on a more complete range of information about the current Grid 'weather' and 'weather forecast'.

In such a complex environment it will be difficult for users to manually 'steer' the usage of resources. Instead, the selection and allocation of resources will have to be automated to optimize their usage. At the same time, constant monitoring of resources will have to be done to avoid overloading or underutilization of Grid resources.

Need for Job Monitoring

Large distributed systems such as Computational Grids require a large amount of job monitoring data for a variety of tasks such as fault detection, performance analysis, performance tuning, performance prediction, and scheduling. The ability to monitor and manage distributed computing components is critical for enabling high-performance distributed computing. As Computational Grids become bigger, more complex, and more widely distributed, it becomes important that this monitoring and management be automated. CMS has also identified the need to monitor the large numbers of jobs that are being executed simultaneously at multiple remote sites. Our approach to this need for job monitoring problem is to develop a job monitoring service which is designed for use in a Grid Analysis Environment and it provides monitoring information that will enables other services within the Grid Analysis Environment to optimize the resource utilization and throughput.

JOB MONITORING SERVICE

The interactive Grid Analysis Environment consists of an ensemble of web services cooperating to form an analysis environment. The GAE is based on service oriented architecture. The Job Monitoring Service is also implemented as a web service. The purpose of the Job Monitoring service is to provide real-time job monitoring and status feedback to a steering service while operating in close interaction with an execution service, such as Condor, to provide interactivity, fault tolerance and error detection.

Job Monitoring Services will monitor a single job that has been submitted for execution in the interactive Grid Analysis Environment and provide an easy-to-use API for retrieval of job monitoring information such as job status, remaining time, elapsed time, estimated run time, queue position, priority, submission time, execution time, completion time, CPU time used, amount of input IO and output IO, owner name and environment variables.

Architecture

The architecture of the Job Monitoring Service is based on XML-based open standards, such as the Web Services Description Language (WSDL) and the Simple Object Access Protocol (SOAP). The Job monitoring service will be implemented as a web service which will be hosted on the Clarens [2] web service framework. It will interact with the execution service to collect monitoring data and then this data will be stored in the data store. Monitoring data will be provided to the clients such as the steering service once it has been requested. The monitoring data will also be published to a MonALISA server for statistical summaries.

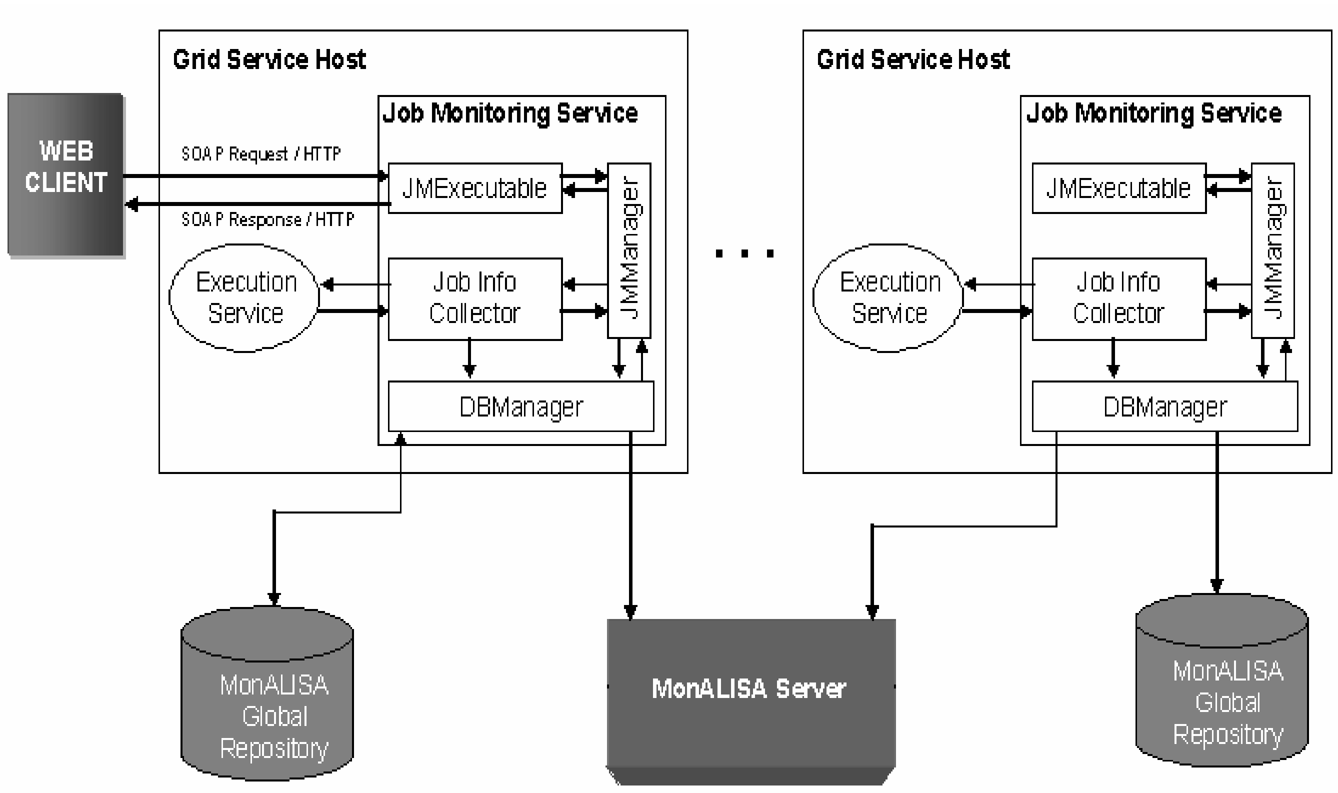

Figure 2: Job Monitoring Service Design

Design

The interaction of Job Monitoring Services with other components within GAE is shown in Figure 1. Once a job has been submitted in GAE users can interact with a steering service to obtain monitoring information and steer the execution of job. The Steering service will access the Job Monitoring Service to provide monitoring information to the user. The Job Monitoring Service will continuously monitor the jobs that have been submitted and whenever the state of a job changes the Job Monitoring Service will update the repository and MonALISA [3]. The main components of the Job Monitoring Service and their interaction are shown in Figure 2.

JMExecutable

JMExecutable is the main interface of the Job Monitoring Service. JMExecutable is implemented as a web service module within the Job Monitoring service. All of the clients that require the monitoring information will access the published methods of JMExecutable. The requests of clients will be forwarded to JMManager by JMExecutable.

Job Information Collector

The role of the Job Information Collector module is to monitor the different jobs that have been scheduled. The Job Information Collector interacts with Condor and provides real time job monitoring information. The Job Information Collector functions in two ways

- It monitors the job execution and whenever the job is completed or terminated due to an error, it sends an update request to the DBManager for that job.
- It provides the monitoring information of the running jobs to the JMManager when requested. The JMManager requests the information from the Job Information Collector if it does not find the required information in the database

A separate module for information collection increases the performance and also allows the flexibility of changing the execution service by incorporating minor changes in Job Information Collector.

JMManager

The JMManager is the most important component of the Job Monitoring Service. The JMManager handles the flow of information within the Job Monitoring Service. It provides transparent access to monitoring information to the JMExecutable. The JMManager gets the monitoring information either from the DBManager or from the Job Information Collector. It first queries the DBManager and if the information is not found in its repository the request is forwarded to the Job Information Collector. On successful retrieval the information is forwarded to the JMExecutable.

DBManager

Each Job Monitoring Service instance has a database repository. The access to this repository will be controlled by the DBManager. The DBManager will update the database local to each Job Monitoring Service instance and will send the updated info to the MonALISA server also. All the queries that are destined for the database are handled by the DBManager. Deploying a separate module that controls the access to the database allows flexibility of having multiples types of the databases.

Job Monitoring Database

Each Job Monitoring Service has a local database that is being used to store the job information. This database is maintained by Clarens. Clarens has been designed to be able to use any database for storing its ACL's, proxy certificates, and other information. For this implementation Clarens was configured to use the MySQL database which will be shared by the Job Monitoring Service.

MonALISA Server Module

The DBManager also publishes the job monitoring information to MonALISA. For this purpose a job monitoring module has been integrated with the MonALISA server. This module intercepts the information published by each Job Monitoring Service instance and formats it into a form that can be stored in MonALISA repository.

IMPLEMENTATION

Currently the Job Monitoring Service is in the development phase at NUST in collaboration with Caltech and CERN. The current implementation of the Job Monitoring Service allows the user to get the real-time job monitoring information through the web service API. The Job Monitoring Services is hosted on Clarens web service framework. The Job Monitoring Service is also publishing the monitored values to the MonALISA server. In order to publish job monitoring information to the MonALISA server, a new monitoring module named JobInfo has been developed that runs on each MonALISA server and provides the job monitoring information to the MonALISA. JobInfo provides the mechanism to collect the values from each Job Monitoring Service instance, to parse the output and to generate a result object.

RELATED WORK

Grid information systems such as Globus MDS [4] and R-GMA [5] provide some monitoring but they do not support all monitoring scenarios. To overcome these limitations application monitoring solutions have been proposed such as NetLogger[6], OCM-G[7], Autopilot[8] and Nimrod. Application monitoring has been done in the CMS production tools IMPALA , BOSS[9], and McRunJob [10].

CONCLUSIONS

In this paper, we have identified the need for a Grid analysis environment to perform interactive analysis. Besides discussing the need, the architecture of the Job Monitoring Service in the GAE was discussed by highlighting the role of a Job Monitoring Service within the GAE. The Job Monitoring Service provides an API that allows the clients to access the job monitoring information of their jobs.

REFERENCES

[1] Proposal for: a Grid Analysis Environment Service Architecture (Authors: Julian Bunn, Dimitri Bourilkov, Rick Cavanaugh, Iosif Legrand, Harvey Newman, Suresh Singh, Conrad Steenberg, Michael Thomas, Frank van Lingen) http://ultralight.caltech.edu/gaeweb/gae_services.pdf

[2] The Clarens Web Services Architecture (Authors: Conrad D. Steenberg and Eric Aslakson, Julian J. Bunn, Harvey B. Newman, Michael Thomas, Frank van Lingen) http://clarens.sourceforge.net/index.php/docs

[3] MonALISA (MONitoring Agents using a Large Integrated Services Architecture). http://monalisa.cacr.caltech.edu/

[4] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: Grid Information Services for Distributed Resource Sharing. Proc. 10th IEEE International Symposium on High Performance Distributed Computing, San Francisco, California, August 2001

[5] S. Fisher et al. R-GMA: A Relational Grid Information and Monitoring System 2nd Cracow Grid Workshop, Cracow, Poland, 2003.

[6] B. Bali´s, M. Bubak, W. Funika, T. Szepieniec, R. Wism¨uller: An Infrastructure for Grid Application Monitoring. Proc. 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002

[7] Zoltan Balaton, Gabor Gombas: Resource and Job Monitoring in Grid

[8] Anand Natrajan, Michael P. Walker: Monitoring Remote Jobs in a Grid System

[9] www.bo.infn.it/cms/computing/BOSS

[10] www.uscms.org/scpages/subsystems/ DPE/Projects/MCRunjob/McRunjob.html