JOB INTERACTIVITY USING A STEERING SERVICE IN AN INTERACTIVE GRID ANALYSIS ENVIRONMENT

Arshad Ali[4], Ashiq Anjum[4], Julian Bunn[1], Richard Cavanaugh[5], Frank van Lingen[1], Richard McClatchey[3], Harvey Newman[1], Conrad Steenberg[1], Michael Thomas[1], Ian Willers[2] , Muhammad Adeel Zafar[4]

[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,zafar.adeel}@niit.edu.pk
[5]*University of Florida, USA*
Email: cavanaug@phys.ufl.edu

Abstract

Grid computing has been dominated by the execution of batch jobs. Interactive data analysis is a new domain in the area of grid job execution. The Grid-Enabled Analysis Environment (GAE) [1] attempts to address this in HEP grids by the use of a Steering Service. This service will provide physicists with the continuous feedback of their jobs and will provide them with the ability to control and steer the execution of their submitted jobs. It will enable them to move their jobs to different grid nodes when desired. The Steering Service will also act autonomously to make steering decisions on behalf of the user, attempting to optimize the execution of the job. This service will also ensure the optimal consumption of the Grid user's resource quota. The Steering Service will provide a web service interface defined by standard WSDL.

In this paper, we have discussed how the Steering Service will facilitate interactive remote analysis of data generated in Interactive Grid Analysis Environment.

INTRODUCTION

The grid couples a wide variety of geographically distributed computing resources. This includes storage and computational systems which are heterogeneous in nature, are owned by different individuals and organizations, and have their own policies with different access and cost models. Users of the interactive Grid-Enabled Analysis Environment (GAE) will have defined quotas to access these resources in the grid. Large data collections are stored and replicated on distributed resources to enhance storage capability and efficiency of access. Grid systems today are mostly used as batch processing systems. Grid users submit batch jobs composed of many atomic tasks, where the dependencies between tasks is known in advance. One significant difference between interactive analysis and batch analysis is that in the interactive session the execution of one task is based on the results of the previous task, such that the dependencies between tasks in the final job structure is not known in advance. This is in contrast to the case of batch analysis, where the task dependencies are known in advance.

Grid Analysis Environment

The Grid-Enabled Analysis Environment (GAE) as described in Figure 1 is an end to end system for scalable distributed multi user (batch and interactive) analysis. Clarens [2] web service hosts are the backbone of this GAE. Clarens offers a consistent framework for hosting the GAE web services, for providing common set of services for authentication, access control, and for service lookup and discovery. It will enable users and services to dynamically discover other services and resources within the GAE through a peer-to-peer based lookup service.

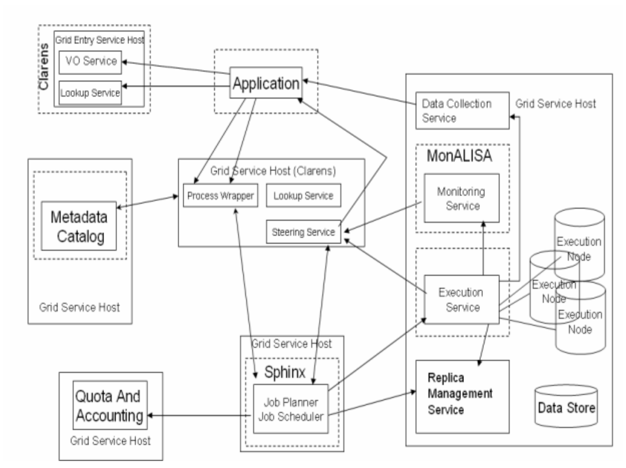


Figure 1: Different services within GAE being developed by Caltech and its collaborators

Need for Steering Service

The heterogeneous and distributed nature of grids makes it much more difficult to perform interactive analysis than on a single machine. A single dataset may be replicated in many locations. Competition for resources is much more apparent. It is much more difficult, if not impossible, to take a "snapshot" of the current state of the Grid, making the best choice of how and where to execute a task difficult to determine.

For these reasons it is necessary to have a grid service that is able to make reasonable choices among a range of possible job execution strategies, autonomously or interactively, while a job is running. Decisions made by these services will be based on a more complete range of information about the current and predicted future grid 'weather'.

RELATED WORK

The concept of Monitoring, Steering and Optimization is used by different projects but has never been implemented as a single generic Grid Service. The noteworthy efforts include MonALISA [3] which is currently being used to perform Monitoring, Steering, and Optimization for the VRVS [4] reflector network. G-Monitor [5] is a web portal that allows a user to monitor, control, and steer the execution of application jobs on Global Grids. G-Monitor does not provide an interface for Applications to interact with the system. Falcon [6], from the College of Computing, Georgia Institute of Technology Atlanta, was created to steer parallel programs through online monitoring, allowing Monitoring and Steering but not Optimization. The goal of the project is to enhance the end-user's insight into the application level behavior of the high-performance program under study. In order to steer the execution of a program using Falcon, the program has to be coded using the steps described by Falcon system. The Vase System [7] is another effort which focuses on the steering of applications by human users. This project lacks autonomous decision-making and algorithmic program steering. NetLogger [8] is a Toolkit for Distributed System Performance Tuning and Debugging. This project provides methodology to get the detailed end-to-end application and system level monitoring and tools for visualizing the log data and real-time state of the distributed system.

STEERING SERVICE

The Steering Service is one of the components of the GAE architecture that allows users to interact with submitted jobs. The Steering Service will be hosted on the Grid Service host Clarens. This service provides users with the real-time control of their job submissions, allowing users to kill, restart, or reschedule their jobs. The MonALISA [3] Monitoring Service continuously collects job information and sends it back to the client.

INTERACTIVITY

The Steering Service provides one way for users to interact with their submitted jobs. Listed below are the types of interaction a client can have with a job.

Job Feedback

Users need responsiveness in an interactive Grid Analysis environment. To satisfy this need the user's jobs must be continuously monitored, and the monitoring information must be relayed back to the user in real-time. A job may contain multiple tasks and each task may be scheduled on a different site. The Steering Service will collect the monitoring information from all of the monitors of a single user job and will send this monitoring information back to the client.

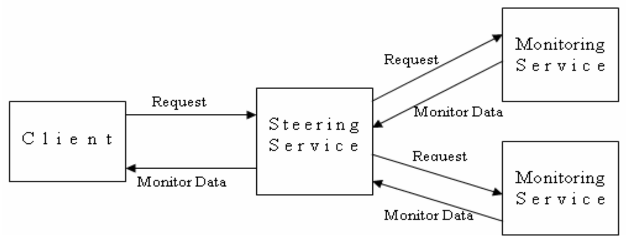


Figure 2: Collection of monitoring information from various monitors

User-Driven Directions

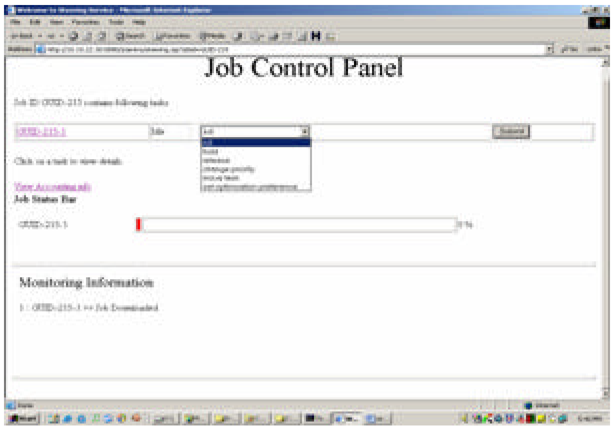
The Steering Service provides a set of APIs for sending commands to running tasks in order to tune the operation of the entire job. The tuning can be in the form of moving a running task from one execution site or replicating data to sites so that it can be made immediately available to more execution nodes. This is done to speed up the execution time or to reduce the resource usage for a user so that they don't exceed their resource quota. While the grid scheduler attempts to optimize the initial job plan, the Steering Service is used to make adjustments to the job plan after the job has started running. The user will submit requests to the Steering Service specifying actions to be taken, and on which jobs/tasks actions will be performed. The Steering Service interacts with the Execution Service to perform these actions on behalf of the user. The Steering Service will respond back to the client with the status of their steering request.

INTERACTION WITH CLIENT

The client will be able to interact with the Steering Service in two ways: through the use of a graphical user interface and through the use of a Web Service API.

User Interface

A user interface (UI) will be provided which will contain a list of all jobs that are to be steered by that Steering Service instance for a particular client. This UI (a part of it is shown in Figure 3) will also provide a way to interact with all the tasks of a job. This UI will also provide a progress bar for each job with detail progress of each task in that job. Since the Steering Service has the estimate of the total time the job / task will take to complete, this progress bar can be displayed based on this estimate and the time the job has taken at that point in time. Moreover, this UI contains the list of actions that a client may take on a job. Complete job monitoring information including job status, failure notifications, job redirection alerts and submission failure, will be displayed directly in the GUI.


Job Control Panel

Job ID: GUID-215 contains following tasks

GUID-215-1 Idle kill [Submit]

Click on a task to view details

View Accounting info

Job Status Bar

GUID-215-1 0 %

Monitoring Information

1 : GUID-215-1 >> Job Downloaded

Figure 3: Steering Service Job Control Panel

API

The Steering Service will provide a Web Service API for monitoring information retrieval and job control.

API for Job Monitoring

The API for Job Monitoring will allow the client to get information regarding the submitted jobs such as job status, remaining time, elapsed time, estimated run time, queue position, priority, submission time, percentage completed, execution time, completion time, CPU time used, amount of input IO and output IO, owner name, environment variables, and any normal and error output produced by the job.

API for Job Control

The API for Job Control will allow the client to kill a running job, kill a queued job, hold a job, resume a held job, and change priority of a job, move job to specific computing element and change the optimization preference of a job.

API for interaction with Scheduler

The API for Scheduler Interaction will allow the Scheduler to give a copy of the job plan to the Steering Service. It also provides a method to contact the scheduler to reschedule the task to a specified Computing Element.

ARCHITECTURE

Figure 4 shows the different components of the Steering Service and its interaction with other services in the GAE. The Steering Service is a web service that is hosted on Clarens.

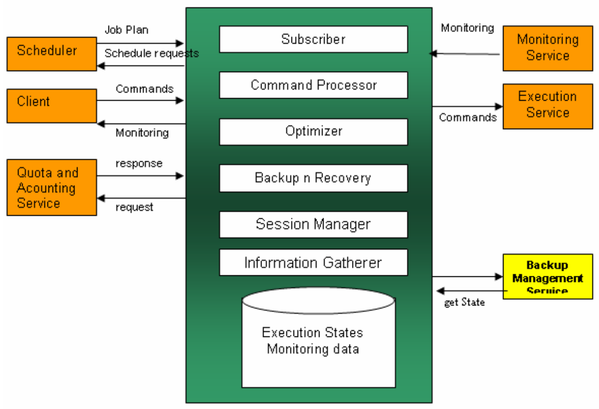


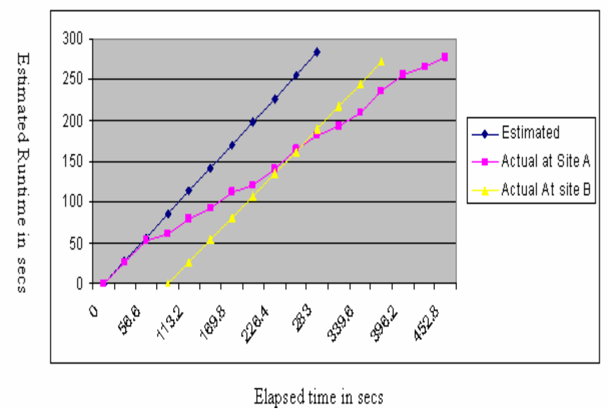
Figure 4: Steering Service (in green)

IMPLEMENTATION AND TEST RESULTS

The Steering Service currently is in the development phase at NUST in collaboration with Caltech and CERN. The current development of the Steering Service allows the user to get the constant feedback of the job on a webpage (and through the web service API as well), control the job, move the job to a specific site during execution, and set optimization preferences. The Steering Service creates backups of the jobs as specified by the job plan and optimizes the resource quota and execution of job autonomously. The backup is created by storing the job executables with the steering service so that in case of failure, the job may be resubmitted. We have provided the web service API that will be used by Scheduler and the Client / UI of the Steering Service.

The Steering Service finds the cheapest site when specified by the user to do so and then recommends the scheduler to submit the job on that site. Currently the mechanism used by Steering Service is to "kill the job on site A" and "resubmit on site B" when it is moved from Site A to Site B. The current implementation of the Execution Service supports Condor as the job execution system. If the support of flocking (Condor-specific feature) is enabled between site A and site B, it will no longer be needed to manually kill the job at A and resubmit it on B while moving the job from A to B. Also for "checkpoint-able" jobs, the problem of restarting the job from its start will be solved and the job will be started from the recent checkpoint. In case of optimization of slow running jobs, tests and results have shown that even if the job is restarted from the beginning, it can complete faster as compared to its current slow execution.

The following graph shows the completion time of the jobs at different scenarios.



The blue line shows the job completing as estimated. Currently this estimate is calculated by running the job many times on different machines which have negligible CPU load. The estimates come out to be 283 seconds; hence ideally the job should complete in 283 seconds. The purple line shows the job that is running on site A under significant CPU load. The Steering Service has monitored the progress of this job and has decided to move this job based on its slow execution rate. This job has been rescheduled on some new site B and has completed before the original job as indicated by the yellow line. For testing purpose, the job was allowed to continue running on site A.

Estimated Runtime	283 seconds
Time taken at Site A	452 seconds
Time taken at Site B	369 seconds

The job can be completed even quicker than 369 seconds if it is checkpoint-able and the flocking is enabled between site A and Site B. A critical factor that affects the job completion time is the time at which the decision to move the job is taken. The quicker the decision is taken, the better the chance that it will complete quicker. Another important factor is the time taken to transfer the data files needed by the job. All of these factors must be taken into account when deciding whether a job should be transferred or allowed to run to completion.

CONCLUSIONS

In this paper, we have discussed the ways in which the Steering Service provides interactivity in GAE. Besides discussing different ways, the architecture of the Steering Service in GAE was discussed by highlighting the role of different components within the Steering Service. Different components of GAE allow the user to monitor, control and steer the execution of their jobs and provide users with fine control over their jobs. The Steering Service is one of the services that will help achieve a self-organizing grid and autonomous behavior of grid services. The Steering Service will also have the ability to make decisions on behalf of the user. Based on the information provided by the Monitoring Service and the policies for that particular job, the Steering Service may make decisions to move individual tasks or even an entire job to some other computing nodes to optimize the resource usage quota of the user as well as the execution of the job.

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] Virtual Room Videoconferencing System http://www.vrvs.org
[5] G-Monitor : A Web Portal for Monitoring and Steering Application Execution on Global Grids
http://www.gridbus.org/papers/gmonitor.pdf
[6] Falcon: On-line Monitoring for Steering Parallel Programs
w.cc.gatech.edu/systems/papers/schwan/GuCPE.pdf
[7] D. Jablonowski, J. Bruner, B. Bliss, and R. Haber. VASE: The Visualization and application steering Environment. In Proceedings of Supercomputing '93, pages 560{569, Portland, OR, November 1993.
[8] NetLogger : A Toolkit for Distributed System Performance Tuning and Debugging
http://www-didc.lbl.gov/NetLogger/