A Distributed Data Storage Architecture for Event Processing by Using the Globus Grid Toolkit

Han Fei[1], Nuno Almeida[1], Paulo Trezentos[1], Jaime E. Villate[2], and Antonio Amorim[3]

[1] ADETTI, Edificio ISCTE, University of Lisbon,
Avenida das Forças Armadas,
1600-082 Lisbon, Portugal
{Han.Fei, Nuno.Almeida, Paulo.Trezentos}@iscte.pt

[2] Department of Physics
School of Engineering,
University of Porto
villate@fe.up.pt

[3] Faculdade de Ciencias, University of Lisbon,
Campo Grande, Edificio C8, sala 8.3.05,
1749-016 Lisbon, Portugal
Antonio.Amorim@fc.ul.pt

Abstract. In this paper we discuss a Grid-based Event Processing System (GEPS). Data intensive problems broadly exist in many scientific computational areas; usually their needs for super storage and computing capacities are difficult to be fully satisfied. Meanwhile the Globus Toolkit has become the de facto standard of building high performance distributed computing environments. Event processing and filtering is a kind of data intensive problem in high-energy physics area. Using the Globus grid toolkit, we have constructed the GEPS system, which provides web-based access to grid computing environments for event processing. Performance result indicates that event processing and filtering can be effectively implemented on GEPS.

1 Introduction

In many scientific disciplines, the need for terabyte data storage, processing and transferring is emerging as a crucial problem; nevertheless large computing and storage facilities are always scarce resources. The storage and computing capabilities are often temporarily and geographically distributed unevenly, sometimes redundant in one place meanwhile scarce in other places. With the scientific and technical applications becoming more and more complicated and sophisticated, many researchers, working and living in different places, have to not only cooperate in the same research project but must also access distributed computing resources.

It is unlikely that conventional methods can meet the demands of providing and sharing these resources. A blueprint of computational grids leveled at addressing these difficulties has been proposed. [1]

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 267–274, 2003.

A Grid [2] is super-computing net, which can connect distributed mainframe computers, super-computers, as well as large numbers of desk top computing devices into easy-to-use computing facilities.

2 The LHC Computing Problem

In the Large Hadron Collider (LHC) accelerator at CERN, there are 10^9 collisions per second taking place per second. Each collision contains about 1 MB of information. One single collision is called an "event". Each event is recorded by surrounding particle detectors for later processing and filtering to pick out the physically interesting ones. Events are recorded at a typical rate of 100 Hz. Considering the data intensive aspect of event processing, computational grids are a possible solution.

2.1 Related Work

Gfarm (Grid Data Farm) is an event processing project [3][4] at KEK (High Energy Accelerator Research Organization) and ICEPP (International Center for Particle Physics, the University of Tokyo). A large scale distributed Gfarm file is divided into several fragments and distributed across the disks in the Gfarm file system. A Gfarm file is a logical aggregation of physical file fragments distributed over many CPU nodes. The processing jobs access the Gfarm files through the Gfarm parallel I/O library, and the job executes in parallel at each node where the physical file fragments reside. The Gfarm file system daemon runs on each node to facilitate remote file operation with access control. When a job is submitted into the Gfarm server, it is redistributed to nodes, which contain the fragment database files. When the job is finished, the results will be retrieved across the network.

Parallel ROOT (PROOF) [5] is another event processing system. The ROOT client session creates a master server on a remote cluster, and then the master server in turn creates slave servers on all the nodes in the cluster. All the slave servers execute user job in parallel. The master server distributes the event data packets to every slave server, carefully adjusting the packet size such that the slower slave servers get smaller data packets than faster slave servers. PROOF uses a TChain object to provide a single logical view of many geographically distributed physical files. The master server keeps a list of all generated packets per slave, so in case a slave failed then remaining slaves can reprocess its packets.

An application in Gfarm system need to use Gfarm I/O library to access Gfarm file, so already existing applications need to be changed and recompiled, and this change means the application will be Gfarm Only. In the case of PROOF, because the PROOF toolkit is relatively reliant on specific grid techniques, the application can’t always utilize the latest grid feature, which can be available only after PROOF provides a realization of that feature. For solving these inconveniences, we propose a distributed Grid-based approach, which facilitates intensive event raw data storage and processing, while providing a uniform application staging interface.

3 GEPS Prototype

In GEPS system we make use of the Grid infrastructure, a back-end database, LDAP directory query, and PHP script web interface. The scalability of GEPS can be easily obtained through freely adding into or picking out any grid computing and storage node. GEPS works like a portal. Behind the friendly appearance of GEPS, many Grid related details are well hidden. Geographically distributed physicists can easily cooperate over the same event processing project, share dispersed events data file, stage jobs, query job status, share computing resources, transfer data file, and visualize events filtering results.

3.1 Introduction of the Events Application

Event processing application is programmed in C++ by using the Root Toolkits [6]. Root is an object-oriented framework, aimed at solving the data analysis challenges of the high-energy physics discipline. It provides a large collection of specific utilities to manage information in an efficient way. Root provides not only an application programming interface (API), but a integrated Root tree class data file visualization environment. The creation of the Root data file has several steps. The first step is to create a structure to store all the raw information of the events. This process consists of the creation of a shared library, which contains all the variables of the event, track, vertices, as well as relation objects.

If the shared library produced in the first step works well, then the next step is to create a Root tree to storage all the objects presented in the raw information file. The Root tree class is optimized to reduce storage space usage and enhance accession speed. Inside the Root tree there is one branch with all events, inside this branch are all event variables that include the tracks, vertices, and relations.

After all the information appears in the Root tree, now it is the time for scrutinizing, one by one, which event will be the candidate that meets the processing standard. The calibration procedure based on the processing standard will be done on each event, then the result will be stored in a new tree with the same structure.

Based on the Grid-enabled computing net, we can divide event raw data into different storage parts, which can be stored in geographically distributed Grid resource nodes. After that, we can stage processing and filtering procedures in a parallel manner, monitor the application running status, collect results, merge the different results data into final data file, and visualize the final data.

3.2 The GEPS Architecture

Figure 1 describes the GEPS architecture. GEPS has an easy-to-use and friendly interface, which is programmed in the PHP script language. No matter where the end user is, the services of GEPS can be easily approached through Internet. After logging into the main-page, several optional functions can be chosen by the end user. The user can request the summarized or detailed information of the available Grid resources. The user can simply fill in a job description form to express some needed information about the job, such as: what is the executable, where does the executable reside, to which Grid nodes is the executable going to be submitted, where is the raw data file, where does the end user want the output to be stored. After filling in the specification of one job, the user can continue to describe another job. The next step is simply push the "submit" button, then jobs will be submitted to grid nodes. After submitting all jobs, the user can continuely monitor the running status of submitted jobs.

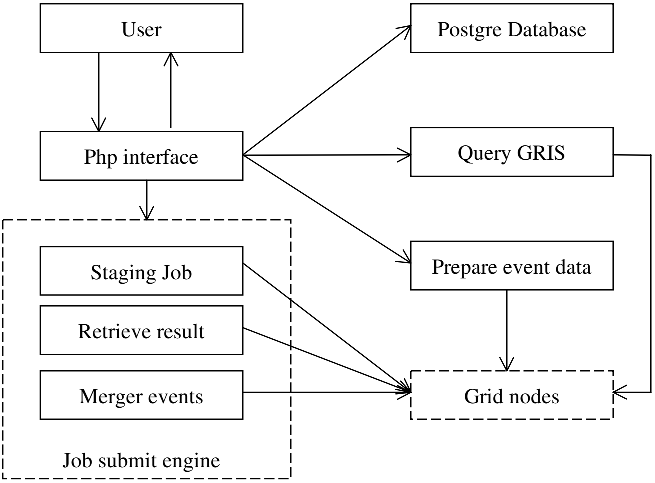

Fig. 1. The GEPS architecture. The user can use GEPS through a PHP scripted interface, which hides many realization details from the outside user.

When the jobs have done, the distributed event result data files will be automatically merged to form the final result, which will be stored in the user specified site. Finally the user can utilize the Root visualization tool to see the event processing and filtering result.

The end user interface is programmed in the PHP script language. It will receive the job descriptions provided by the end user, and it will insert the new job into the PostgreSQL database. If the end user wants to query information about the grid computing environment, the PHP scripts will call a function to query a GRIS ldap server.

Before submitting the executables, the event raw data needs to be copied to the grid nodes, according to the demands of the job specification.

The grid job submission engine will parse the job specification tuple in the PostgreSQL database, analyze the job executing environment and raw event data distribution demands, synthesize the RSL sentences, submit the jobs, monitor the status of submitted job.

4 The GPES Prototype Implementation

4.1 The Computational Grids Environment

The Globus Grid toolkit evolved out of the I-WAY high performance distributed computing experiment [7]. Before the Globus Grid became the de facto high performance computing environment, there were other candidate grid architectures, include using object-based technology and web technology [8].

Table 1. Globus components in GEPS.

Component	Usage
GRAM	Executable staging
GRIS in MDS	Query Grid node information
GASS	Transfer raw data, retrieve remote results

Table 1 lists the grid components used in GEPS. In the Grid job submission engine, the new job specification tuples are selected from the back-end PostgreSQL database. For each new job, by parsing the job specification tuple, a job Resource Specification Language (RSL) sentence is formulated, then a raw data file is transferred (by using GASS components) in accordance with the setting of relevant resources, and then the GRAM component (globus-gram-client) is used for remotely submitting and managing job. The run time stdout and stderr is defined in the RLS sentence. After all submitted jobs having finished, GASS file access functions are used for retrieving distributed event results.

4.2 Query GRIS LDAP Server

Monitoring Discovering Service (MDS). The Globus Toolkit has provided an information Monitoring and Discovery Service (MDS)[9], which acts as a resource information registry and discovery agent. The MDS includes a standard, configurable information provider framework called a Grid Resource Information Service (GRIS). GRIS is implemented as an OpenLDAP[10][11] server. Each Grid node can run a local GRIS.

Through GEPS, the end user can query properties of the grid nodes, such as how many processors are available at this moment, what bandwidth is provided, etc. The MDS provides two interfaces: interactive and programmatic. By default, a GRIS service is automatically configured to port 2135. In our GEPS, the grid-info routine obtains the overall Grid node information by querying this port through the LDAP protocol. The PHP script will call the grid-info routine to get the results.

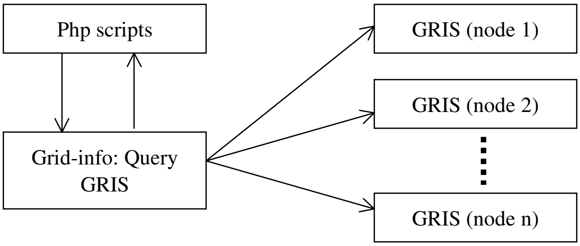

Fig. 2. Querying Grid node resource information through the LDAP protocol. In GEPS the end user selects interesting objects or just simply makes a choice of default objects. The PHP scripts then call Grid-info. Grid-info will send LDAP queries to the GRIS in each Grid node and get the available resources list.

5 Experiment and Results

From August to October in 2002, we tested 13 groups of raw event data, and with a total of 130 experiment executions (for decreasing the effect of system and network latency in executable staging and data transfer). The current GEPS demonstration prototype temporarily consists of two server, gandalf and hobbit. Because the GEPS topology structure has the feature of scalability, in the future more nodes can be easily incorporated. One of the advantages of computational grids is that any part can be easily changed without any global effect.

Different granularities of event data will dramatically affect the overall performance of the GEPS system. This is reasonable, because with many smaller files of raw event data, the portion of system cost dedicated to raw data transfer will become larger in total execution time. Based on the event data file size, Figure 3 gives the relation between running only on hobbit and running in parallel between gandalf and hobbit. The unit on Y-axis is time cost in second, and the unit in X-axis is the number of events in raw event data file. In raw event file each event is about 1M bytes in size. From the illustration we can easily see that the data file size of approximate 2000 events is a watershed. Data files consisting of less than 2000 events run in tightly coupled computing environments will have better performance. But usually our event raw data files can be easily much larger than 2000 events. From the results illustrated in Figure 3 we know that to some extent our GEPS has provided better performance.

The GEPS network connection is fast Ethernet. The user defines the raw event data distribution by using the RSL sentence. Before a job can be submitted to the grid gatekeeper through the grid client API, the raw event data will firstly be transferred to the grid nodes in accordance with the raw event data distribution specification. GEPS currently uses Globus GASS file access API for transferring raw data and result file between gird nodes. Figure 3 only gives the comparison of processing time cost between GEPS and hobbit.

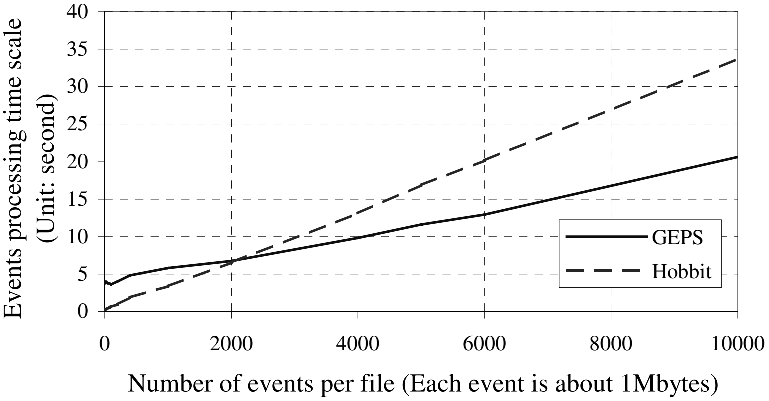

Fig. 3. Performance in GEPS & hobbit with different event raw data file sizes.

6 Conclusions and Future Work

We have described the GEPS prototype, which provides an integrated meta computing environment for event processing and filtering. In GEPS, Grid related detail and relevant middleware specifics have been hidden from the end user. GEPS facilitates the scalability of intensive event data storage. Using GEPS, physicists can easily administer and share distributed data and take advantage of distributed computing resources. This prototype has incorporated to date innovative Grid concepts and mechanisms.

The smaller bandwidth and the larger latency due to the geographical distribution of the Grid computational resources are the main reason of parallel inefficiency. We are working on adding GridFTP into our prototype. Because multiple TCP streams and proper TCP buffer sizes are very important to obtaining better performance in TCP wide area links [12], we are trying to add this feature into the GEPS prototype. We are also exploring the feasibility of solving other physics problems in the GEPS prototype environment.

Acknowledgements. This work was supported by Fundação da Ciência e Técnologia under the grant CERN/P/FIS/43719/2001. The first author gratefully acknowledges the postdoctoral fellowship by the FCT. The third author would like to thank ADETTI (Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática) for their support to this work.

References

1. I.Foster and C. Kesselman: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
2. S. Barnard, R. Biswas, S. Saini, R. Van der Wijngaart, M. Yarrow, L. Zechter, I. Foster, O. Larsson: Large-Scale Distributed Computational Fluid Dynamics on the Information Power Grid using Globus. Proc. of Frontiers '99 (1999)
3. Y.Morita, O.Tatebe, S.Matsuoka, N.Soda, H.Sato, Y.Tanaka, S.Sekiguchi, S.Kawabata, Y.Watase, M.Imori, T.kobayashi: Grid Data Farm for Atlas Simulation Data Challenges, Proceedings of CHEP 2001 (International Conference on Computing in High Energy and Nuclear Physics) (2001) 699–701
4. Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, Hiroyuki Sato, Yoshio Tanaka, Satoshi Sekiguchi, Yoshiyuki Watase, Masatoshi Imori, Tomio Kobayashi: Grid Data Farm for Petascale Data Intensive. Electrotechnical Laboratory, Techinical Report, TR-2001-4. http://datafarm.apgrid.org
5. René Brun, Fons Rademakers: Distributed Parallel Interactive Data Analysis Using the Proof System. Proceedings of CHEP 2001 (International Conference on Computing in High Energy and Nuclear Physics) (2001) 704–707
6. http://root.cern.ch
7. I. Foster, J. Geisler, W. Nickless, W. Smith, S. Tuecke: Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment. Proc. 5th IEEE Symposium on High Performance Distributed Computing (1997) 562–571
8. S. Brunett, K. Czajkowski, S. Fitzgerald, I. Foster, A. Johnson, C. Kesselman, J. Leigh, S. Tuecke: Application Experiences with the Globus Toolkit. Proceedings of 7th IEEE Symp. on High Performance Distributed Computing, July 1998
9. S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke: A Directory Service for Configuring High-Performance Distributed Computations. Proc. 6th IEEE Symposium on High-Performance Distributed Computing (1997) 365–375
10. Heinz Johner, Michel Melot, Harri Stranden, Permana Widhiasta: LDAP Implementation Cookbook. SG24-5110-00, IBM. International Technical Support Organization, http://www.redbooks.ibm.com
11. Heinz Johner, Larry Brown, Franz-Stefan Hinner, Wolfgang Reis, Johan Westman. Understanding LDAP. SG24-4986-00, IBM. International Technical Support Organization, http://www.redbooks.ibm.com
12. J. Lee, D. Gunter, B. Tierney, B, Allcock, J. Bester, J. Bresnahan, S. Tuecke: Applied Techniques for High Bandwidth Data Transfers Across Wide Area Networks. Proceedings of International Conference on Computing in High Energy and Nuclear Physics, Beijing, China, September (2001)