

Enabling Queries Using the Grid-brick Approach: A Distributed Data Storage Architecture

Han Fei, Nuno Almeida, Paulo Trezentos, A. Amorim
ADETTI, FCUL and CFNUL, Universidade of Lisboa
Han.Fei@iscte.pt, Nuno.Almeida@iscte.pt,
Paulo.Trezentos@iscte.pt, Antonio.Amorim@fc.ul.pt

Abstract

This paper presents a Grid-based parallel Event Processing System (GEPS) that is applicable to all the domains where large collections of separate blocks of data (images) need to be processed, stored and analyzed. Data intensive applications are becoming increasingly important in many science areas. In many domains the need for future computing and data management capabilities is difficult to accommodate within the expected technology improvements and the usage of scalable solutions, distributed over a large number of nodes, becomes essential. The processing, storage and analysis of High Energy Physics collisions illustrates the main challenges of this research area. Using the Globus grid toolkit, we have developed the GEPS system that provides a framework for data storage, processing and analysis that is based on the Grid-brick approach. The main concept of this approach is that individual farm nodes mirror the same architecture to provide all the available services for the data stored locally that is to each node. The data is split over different nodes that are responsible for processing the local queries. Performance results indicate that event processing and filtering can be effectively implemented on GEPS, encouraging the continued effort to improve the GEPS prototype.

1. Introduction

In many scientific disciplines, the need for terabyte data storage, processing and transfer are emerging as crucial problems. While large computing and storage facilities are always scarce, problem solving usually needs cooperation of many researchers in different places that may have access to computing resources distributed over several centers. The storage and computing capabilities are distributed unevenly, being sometimes redundant and sometimes scarce. With the scientific and technical applications becoming much more demanding and sophisticated, one needs new technologies to allow the distributed groups of researchers to efficiently take

part in problem solving by sharing the available resources.

From the realization that it is unlikely that the conventional methods of high performance computing can meet this challenge, a blueprint of a computational grid, leveled at addressing these difficulties, has been proposed[1].

The computing Grid is based on the integrated set standard services that are available on the net and that are able to solve the most demanding problems by sharing the resources of Grid centers in many geographic locations. The Grid is supposed to take advantage of temporary free resources in some locations and use them to help other resource stringent entities. To enable effective and reliable usage, the Grid must provide security mechanisms to protect the integrity of sensitive resources.

Currently the Globus toolkit[2] is becoming the most used software infrastructure to build the computational grid, even though there are several other alternatives [3][4][5].

In this paper we discuss the GEPS structure, depict progress made to date in GEPS project. This paper is structured as follows: Firstly, we analyze the feature of event processing problems in high-energy physics computing discipline; secondly, we describe the GEPS construction; finally we discuss the experiment results.

2. The LHC computing problem

In the physics experiments at the Large Hadron Collider (LHC) at CERN, about one petabyte of raw data will be produced each year. The only proposed way to process these petabytes of data in a reasonable time is by using a massively distributed computing infrastructure. In the LHC accelerator, each experiment has to cope with 10^9 collisions per second each containing about 1 MB of information. One single collision of particles in the accelerator is called an "event". Information on the particles emerging from the collision point is recorded by surrounding particle detectors. The events are then processed and filtered to pick up the physically interesting ones, which are then recorded at a typical rate of 100 Hz for later analysis. With the experiments sharing

critical requirements on the storage, processing and analysis of vast amounts of event data, the prospects of resorting to a Globus Grid meta-computational network seems the only foreseeable solution.

2.1. Related work

The European DATAGRID [6] and CROSSGRID [19] programs are developing and testing the middleware that lies above the Globus toolkit, and will support the LHC computing Grid. The US GriPhyN [20] and the e-science grid project [21] in the UK are other strong initiatives that also develop and test Grid software and maintain large scale test-beds.

The paradigm for the architecture of the Grid data centers at different levels involves always a physical separation between the data repository and the computing infrastructure that accesses data from the disk or tape servers. The access to the vast amounts of data could be made simpler by clustering the events on the local machines that also provide the CPU power. The present work explores this possibility in an integrated way by making each Grid-brick a service provider that enables all processing and analysis over local data for which this node is responsible. The idea of sharing the data among the different nodes was already described before but without any assumption on the association of the services and the data in each node.

Among these previous studies one finds the Gfarm (Grid Data Farm) project [8][9] project 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 the nodes which contain the fragment database files. When the job is finished, the results will be retrieved across the network.

Another service interface that shares the load of processing many different events over a collection of nodes is the parallel ROOT Facility, PROOF [10]. This system is designed for the parallel interactive analysis of terabytes of data on clusters of heterogeneous computers but implicitly assumes that the event data has to be moved between the nodes while it is being processed. The PROOF implementation is based on the facilities available in the ROOT framework, such as TTree object

containers, object streaming, scripting, networking classes, remote file access, etc. 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.

3. The Grid-brick approach

Events processing job is a data intensive scientific application. An application in Gfarm system need to use Gfarm I/O library to access Gfarm file, so already existed large amount of 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 rely on the bases of specific grid technique, so the application always can't 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 Grid-brick approach, which facilitates intensive events raw data storage and process with distributed grid computing and storage nodes, and provides a uniform application staging interface. Figure 1 illustrates the Grid-brick topology structure. Every grid node includes its local events raw data storage. Job submit server will receive the job configuration from the PHP script interface, and store the job configuration into PostgreSQL database. This topology structure allow an ongoing either increasing or decreasing grid nodes, which become the bases of metadata storage environment, each grid node acts like a brick.

Computational Grids provide a bundle of utilities. It provides command-line interface, as well as application programming interface (API). By using the command-line interface, if a user wants to submit an application to grid network nodes then the user have to log in each site in turn and memorize the arcane commands needed to allocate computing resources. For transferring and retrieving distributed data the user need to do almost the same thing, besides this, the user still need to keep trace with distributed data storage, some resource can be unusable in the immediately future, to memorize the site name as well as the data file name, lest inadvertently damaging or losing. In our events processing case, the event processing and filtering applications will be automatically submit to large amount of distributed

computing nodes for approach of the distributed events raw data storage.

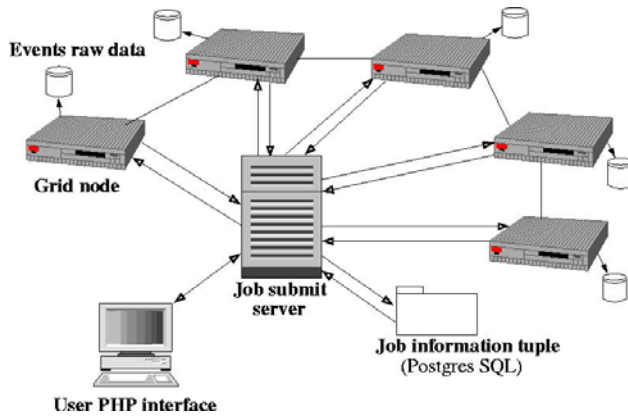


Figure 1. Grid-brick topology illustration

Our Grid-enabled Event Processing System (GEPS) utilizes Grid infrastructure, back-end database, LDAP directory query, and PHP script web interface. Each computing grid node like a "brick". The scalability of GEPS can be easily obtained through freely adding into or picking out any grid computing and storage brick. By using GEPS the end-user can get approach to a uniform and integral utilization of distributed computing and storage environment for event processing. Beside this, meanwhile GEPS works like a portal, which as an internet neologism usually refers to a start-point, from which, for a well known group of people having common interests to approach specific resources or relevant services. Behind the friendly appearance of GEPS, many Grid related details are well hidden. Geographical distributed physicists can easily cooperate over the same events processing project, share dispersed events data file, stage jobs, query job status, share computing resources, transfer data file, and visualize events filtering results.

4. Grid-brick facilitated GEPS prototype

4.1. Event processing application

Event processing application is programmed in C++ by using the Root Toolkits [11]. Root is an object-oriented framework, aimed at solving the data analysis challenges of 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 there is all of event variables that include the tracks, vertices, and relations.

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

During the first year of the accelerator run in LHC, an order of 10^{16} collisions will be observed and 10^9 events will be recorded. Events process is not only computing time consuming, but also eat up very large raw data foodstuff, besides this it still produces large amount of output. 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 procedure in a parallel manner, monitor the running, collect results, merge the different results data into final data file, and visualize the final data.

4.2. GEPS framework and structure

Figure 2. describes the GEPS user interface. GEPS has a easy-to-use and friendly interface, which is programmed in PHP script language. No matter wherever is the end user, who can easily approach the services of GEPS through internet. After log into the mainpage, there are several optional function menu can be choiced by the end user. The user can query about the summarized or detail information of the available Grid computational resources. The user can simply fill in a job describing form to express some needed information about the job, such as: what is the executable, where does the executable locate, which of Grid nodes is the executable going to submit, where is the raw data file, where does the end user like the output to be stored, and after filling in the specification of one job if the user would like then he can continue to describe another job. The next step is simply push the "submit" button, after submitting all jobs

the end user can continually monitor the running status of submitted jobs. When the jobs have done, the distributed events result data file will be automatically merged to formalize the final result which will be stored in the user specific site, finally the user can utilize the Root visualizational tool to see the events processing and filtering result.

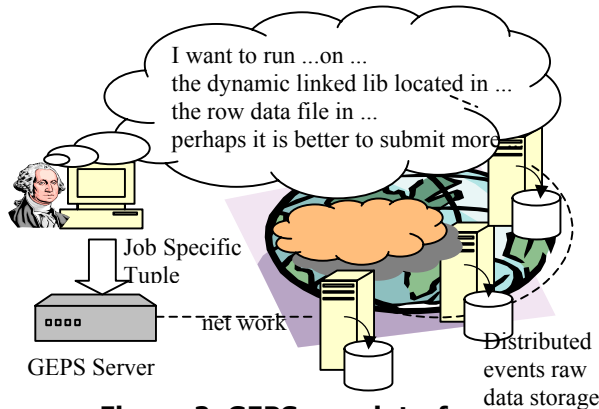


Figure 2. GEPS user interface illustration

Behind the GEPS there are many realization details, such as Grid job submit environment setting, back-end database, LDAP client routine, and data file transfer. All these realization relevant details, infrastructure, and middleware have been hidden from the end user. The end user will be aware only the environment illustrated in Figure 2.

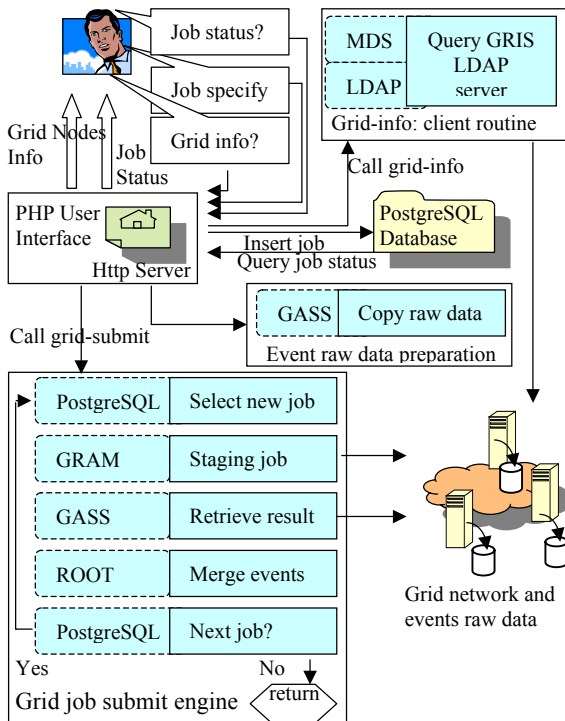


Figure 3. GEPS structure and core technology

Figure 3 illustrates the GPES realization details. There are 6 main parts in GPES prototype, which are user interface, postgresQL database, grid job submit engine, event raw data preparation, gid-info query client routine, and the grid computing network environment.

The end user interface is programmed in 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 the grid-info routine to query GRIS ldap server for reply of needed information.

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

The grid job submit engine will parse the job specification tuple in 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.

All this detail and specifics can't feel or see by the end user.

5. GPES prototype implementation

5.1. The computational Grids environment

Globus Grid evolved out of I-WAY high performance distributed computing experiment [12]. 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 [13].

The development of the Globus Project is lead by Argonne National Laboratory and the University of Southern California's Information Sciences Institute [14]. Globus provides the basic software infrastructure for computational Grid. Grids are super Internets with geographically and organizationally distributed physical structure for high-performance computing. It can include worldwide collections of high-end resources, such as storage, information, special resources.

In Grid job submit engine, the new job specifying tuples are selected from back-end PostgreSQL database. For each new job, by parsing the job depicting tuple, job Re-source Specification Language (RSL) sentence is formu-lated, then raw data file is transferred (by using GASS components) in accordance with the setting of relevant re-sources, and then GRAM component (globus-

gram-client) is used for remotely submitting and managing job. The running time stdout and stderr is defined in RLS sentence. After all submitted job having finished, GASS com-ponents (gass file access functions) are used for retrieving distributed events results.

Table 1. Globus components in GEPS

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

5.2. Query GRIS LDAP server

Computational Grid enables geographical distributed user sharing of different computational resources and information resources. In this setting, the discovery, summarization, and monitoring of resources and services become challenging problems due to the fairly diversity, large amount, heterogeneity, and distribution of computational grids environment.

Metacomputing Directory Service (MDS). Globus Toolkit has provided an information Monitoring and Discovery Service (MDS)[15], which acts as a resource information registering and discovering agent. MDS includes a standard, configurable information provider framework called a Grid Resource Information Service (GRIS). GRIS is implemented as an OpenLDAP server back end. Each Grid nodes 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. MDS supports the concept of "virtual organizations", where geographical distributed researchers engaged in collabo-rative activities share resources with each other. A virtual organization that integrates and provides coherent resource information spanning virtual organizations is necessary in order for facilitating applications to configure themselves and adapt to dynamic grid available resources.

To date MDS can provide information query for static host information, dynamic host information, storage system information, and network information. By using LDAP protocol, the GEPS grid-info routine query GRIS for specific information, the GRIS being queried calls the information provides. Feedback results from an information provider are filtered by GRIS to eliminate any objects, that don't match exactly the query filter specification.

MDS provides two interfaces: interactive and programmatic. By default, a GRIS service is

automatically configured to port 2135. In our GEPS, the grid-client routine obtains the overall Grid nodes information by querying this port through LDAP protocol.

By using LDAP protocol we provide Grid metacomputing environment information to web-end user. At present implementation web-end user can send out specific query command or default query command, then the customized resource information query result will be printed in a web page window.

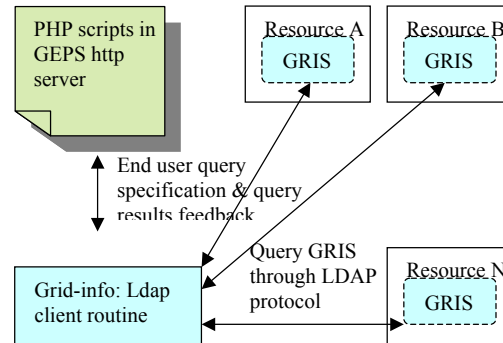


Figure 4. Query Grid nodes resource information through LDAP

Figure 4 illustrates the information query mechanism in GEPS. The MDS information directory structure utilizes the LDAP model, which include a Directory Information Tree hierarchy and object class definitions. In GEPS the end user select interested objects or just simply make a choice of default objects (which will be main resources information briefing) from a lists programmed in PHP script language. The PHP script will call grid-info routine to get the results. Grid-info is a LDAP client routine, which will send query to GRIS LDAP back end server built in each relevant resource node, the feedback information has been customized by GRIS in accordance with the query specification. After obtaining all feedback, then the query result will be print by PHP scripts in a web page text window.

5.3. Query Grid info through LDAP protocol

In GEPS Grid-Info query routine use LDAP [16][17] protocol send request message to GRIS in distributed Grid nodes for specific local resource information. A directory is a listing of information about resource object, all the information are arranged in some order that gives details about each resource node object. MDS use a directory structure, which consists of a directory information tree and object class definitions. The basic

structure of LDAP is a simple information tree. Starting at root node, it consists of a hierarchical view of all its objects and attributes attached with the specific object. LDAP directory information tree, unlike the multi megabyte file in ordinary relation database reservoir, the nodes in which are usually much smaller. The contents of the directory are called object classes and entries. Object class depicts what information can be stored in the directory. Objects are named by their position in the tree. Each node in the tree calls an entry, or Directory Service Entry (DSE). The entry contains the attributes of the real or abstract object in the Grid computing network. The content of an entry is stored as attribute/value (<name, value>) tuple.

For computational Grids, the root in Directory Information Tree (DIT) is "o=grid", where "o" is a descriptor for "organization". The DIT branches out following this root by adding organizations (o), organization units (ou), various resources, etc. Every node in the DIT structure uses its unique path as an unambiguous name to the entry. The unique path of each node is named Distinguished Name (DN), so the DN provides the path from leaf to root entry in the DIT. The DN is the full name of the entry within the DIT. Normally the DN is the very first attribute of the entry. To query the entry of specific object in DIT, you must use the DN. Figure 5 illustrates the Grid DIT structure in Gandalf server of GEPS.

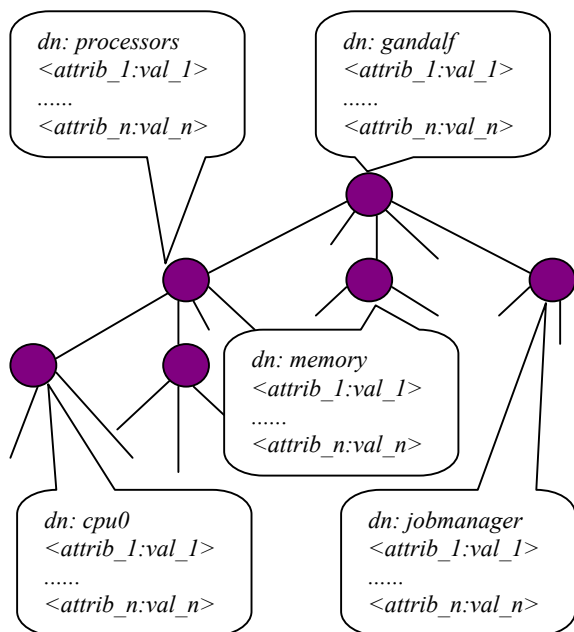


Figure 5. GRID Directory Information Tree (DIT) structure in GANDALF server of GEPS

6. Experiment and results

From August to October 2002, we test 13 group of events raw data, and with totally 130 times of experiment execution (for decreasing the effect of system and network latency in executable staging and data transferring) in our GEPS, which for the reason of at present being a demonstration prototype temporarily consists of two server, gandalf and hobbit, because GEPS topology structure has the feature of plug-in-and-play, like grid "brick", so in the future more distributed resource will be incorporated into GEPS. Meta computational environment is the one of advantages of computing grids, any parts of the grids can be easily changed without any global effect.

Different coarse-grained scale level of events data will dramatically affect the overall performance of GEPS system. This is reasonable, because with smaller Events raw data the portion of system consumption dedicated to raw data transfer will become larger in total execution time. With the same Events Data file grain, Figure 6 give out the relation between running only in Hobbit server and parallel executing in distributed Gandalf and Hobbit server. From the illustrator we can easily know that the data scale of approximate 2000 events is a watershed. Data file consisting of events less than 2000 runs in tightly-coupled computing environment will have better performance. But usually our events raw data file can be easily much larger than 2000 events. From the results illustrated in Figure 6 we know that to some extent our GEPS has given out a good performance. Figure 7 and Figure 8 illustrate the system latency in Gandalf and Hobbit. In these two Figures, the total consumption is the time from staging until finishing. With different grain scale of events raw data file, the latency is nearly invariable (this latency not including raw data transfer time between distributed grid nodes). Figure 9 describes the staging cost, which doesn't change with different Events raw data file. From the analyzing we can draw a conclusion that the bandwidth in raw data transferring can be a key bottleneck to overall system performance. At present the Events raw data is transferred by using gass-file-access API toolkits, in the future we will try to use GridFTP and dynamic caching to further improve the performance.

Figure 6. Performance in GEPS & Hobbit with different coarse-grained level of events raw data

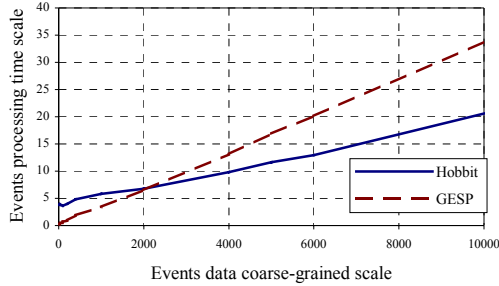


Figure 7. System latency in Gandalf

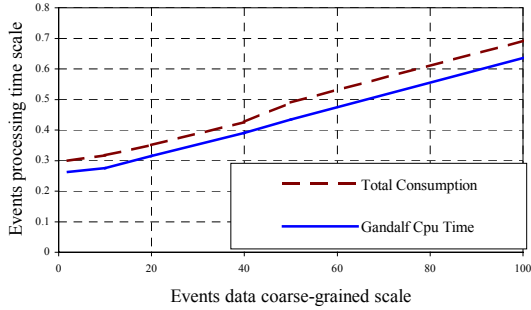


Figure 8. System latency in Hobbit

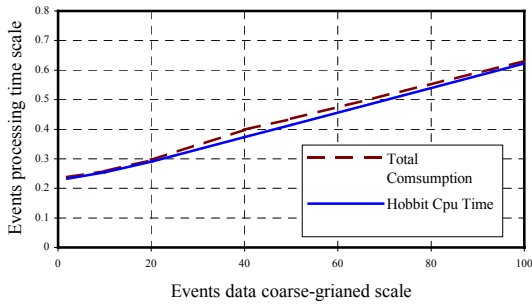
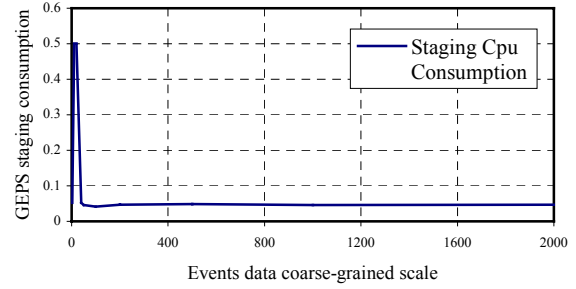


Figure 9. Staging consumption



7. Conclusions and future work

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

The small 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 with adding GridFTP into our prototype. Because multiple TCP streams and proper TCP buffer sizes are very important to obtaining better in TCP wide area links [18], and caching and latency-tolerant method are also very critical method, so we are trying to add this feature into GEPS prototype. We are also exploring the feasibility of solving other physics problems in GEPS prototype environment.

8. Acknowledgements

This work was supported by Fundação da Ciência e Tecnologia under the grant CERN/P/FIS/43719/2001.

9. 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] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr., Legion: The Next Logical Step Toward a Nationwide Virtual Computer, Technical Report No. CS-94-21 June, 1994, Department of Computer Science, University of Virginia

[4] Grimshaw A.S., Wulf W.A., French J.C., Weaver A.C., Reynolds P.F., The Legion Vision of a Worldwide Virtual Computer. CACM 40, 1997.

[5] Almond J., Snelling D., Unicore: Uniform Access to Supercomputing as an Element of Electronic Commerce. Future Generation Computer Systems 613(1999), 1-10.

Conference on Computing in High Energy and Nuclear Physics), 699-701

[9] 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, Technical Report, TR-2001-4*. <http://datafarm.apgrid.org>

[10] 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)*, 704-707

[11] <http://root.cern.ch>

[12] 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, pp. 562-571, 1997.

[13] 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.

[14] The globus project: <http://www.globus.org>

[6] The European DATAGRID project <http://eu-datagrid.web.cern.ch/eu-datagrid/>

[7] Andrei E. Chevel, Vladimir Korhkov, Deployment of Globus tools at St.Petersburg(Russia), Proceedings of CHEP 2001 (International Conference on Computing in High Energy and Nuclear Physics), 661-662

[8] 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*

[15] 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, pp. 365-375, 1997.

[16] Heinz Johner, Michel Melot, Harri Stranden, Permana Widhiasta. LDAP Implementation Cookbook. SG24-5110-00, IBM. International Technical Support Organization, <http://www.redbooks.ibm.com>

[17] 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>

[18] 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.

[19] <http://www.crossgrid.org/>

[20] <http://www.griphyn.org>

[21] <http://www.escience-grid.org.uk>