Grid computing in High Energy Physics using LCG: the BaBar experience

James Cunha Werner

University of Manchester

Abstract

This paper presents accounts of several real case applications using grid farms with data- and functional-parallelism. For data parallelism a job submission system (EasyGrid) provided an intermediate layer between grid and user software. Performance results suggest new directions of research for optimizing throughput. For functional parallelism a PVM algorithm ran user's software in parallel as a master / slave implementation. This was applied to typical particle physics applications using real data and Monte Carlo simulation: hadronic tau decays, searching for anti-deuterons, and neutral pion discrimination using genetic programming algorithms. (This discrimination is used to obtain high purity samples to compare experimental results with theoretical predictions.) Studies of performance show considerable reduction of time execution using functional gridification.

1. INTRODUCTION

The GridPP collaboration [7][8] and several other e-science projects have provides a distributed infrastructure and software middleware in UK. The LCG (Large Hadrons Collider Computing Grid) [4][5][6] software, developed by an international collaboration centred at CERN, provides a system for batch processing for High Energy Physics (HEP) through hundreds of computers connected by the Internet.

In HEP processing each event is independent, and Monte Carlo simulation and data analysis can be divided in several hundreds of independent jobs running in parallel over each data file with thousands of million of events. It is therefore suitable for data parallelism, but less so for functional parallelism. The merged results provide physicists with information to study fundamental matter constituents. To manage this complex process, a reliable job submission system for data parallelism integrating resources to user's software was developed (EasyGrid) and is described in this paper.

Functional parallelism [1][2][3] is a technique where functions in the program are executed independently in parallel more efficiently. There is a master program (or client) that request slave programs (or servers) for some service and coordinates effort and synchronization.

To use LCG in scientific processing with functional parallelism, an efficient and secure communication mechanism is necessary to allow data transfer between jobs in different worker nodes. The time needed to establish the partition and transferring of information cannot be bigger than the processing time, or the solution will be inefficient (it will take longer than running in one computer).

Another important concern is the server's reliability. If any server goes down, the master program must re-submit the task to another server, and not stay waiting forever. To overcome these difficulties, a gridification algorithm was developed and described in this paper. Data and functional gridification algorithms can be used together.

To test the gridification algorithm, this paper addresses the problem of neutral pions discrimination obtained by genetic programming. Genetic programming (GP) [9][10][11][12][13] is an artificial intelligence algorithm that mimics the evolutionary process to obtain the best mathematical model given an economic function to evaluate performance (called the fitness function).

Genetic programming has been used to determinate cuts to maximize event selection [14][15][16]. Genetic algorithms can also be associated with neural networks to implement discriminate functions [17] for Higgs boson.

Our approach is innovative because the mathematical model obtained with GP maps the variables hyperspace to a real value through the discriminator function, an algebraic function of pions kinematics variables. Applying the discriminator to a given pair of gammas, if the discriminate value is bigger than zero, the pair of gammas is deemed to come from pion decay. (The neutral pion, with a mass of 135 MeV/c^2, decays to two photons.) Otherwise, the pair is deemed to come from another (background) source. This approach has been successfully applied to medical diagnostics [18][19][20][21].

This paper starts describing LCG architecture (section 2), moving to Data gridification (section 3), how EasyGrid works (section 4), and its performance (section 5). Functional gridification is described in section 6, followed by its installation procedures (section 7), Genetic programming (section 8), the BaBar experiment (section 9), neutral pion discrimination (section 10), and finally functional gridification performance (section 11).

2. LCG grid architecture.

Grid middleware, working over the Internet, provides the necessary hardware infrastructure grouped in functional modules. It can be seen as a homogeneous common ground in a heterogeneous platform.

LCG grid implementation has been used in HEP experiments due its characteristics of independent parallel processing. There are many worker nodes (WN), managed by Compute Elements (CE), running jobs distributed by resource brokers (RB).

The testbed platform available at University of Manchester [22] contains the following 2 CPUs per node and 100Mbits/s network cards computers: User interface (UI): It is the module that allows users interact with grid. It contains the EasyGrid program to submit jobs and commands to the grid, check job status, recover outputs, etc; 1 **Resource broker** (RB): It is responsible for matching the jobs with resources, implementing policies, and sending jobs for processing in a remote resource CE. Resources definition is written in the Job Description File (JDF) using ClassAds; 1 **Computer element** (CE): it is the batch manager (PBS) that allocates resources in the farm to run users' software; 6 **Worker Nodes** (WN): computers that will run users' software. They contain the packages installed, libraries, and can access necessary data for processing; 1 **Storage Element** (SE): it is the grid mass storage element , with 7 Gigabytes; 1 **Information system** with the Berkeley Database (BDII): the catalogue of resources published in the grid. It is distributed in two levels. The first catalogue all CE BDII entries. The second, in the CEs, contains the information about the farm managed by the CE. This computer also contains the Proxy manager (PX) which stores proxies for long jobs in the grid, and the Grid monitor database; 2 NFS file servers with 1.7 Terabytes and a 1 Gigabits/s network card, accessible through the farm.

The production farm contains 1 CE and 28 WNs, sharing other systems with the testbed. The BaBar virtual organization has its own LCG File Catalogue (LFC) in Italy.

3. Data gridification using EasyGrid/LCG.

HEP requirements have pushed grid establishment in data parallelism. HEP experiments measure the results of collisions between relativistic particles, called an event, and store a large volume of data (hundreds of Terabytes) for further analysis.

The total event sample is separated into different *datasets* with hundreds of data files according to simple criteria, and a physicist will be interested in performing analysis on a selected dataset.

Data gridification is implemented through EasyGrid Job Submission [22] software. It is an intermediate layer between Grid middleware and user's software. It integrates data, parameters, software, and grid middleware doing all submission and management of several users' software copies to grid.

EasyGrid's commands are: a. easymoncar: run Monte Carlo Events generation. b. easysub: run Raw data analysis. c. easygftp: run Generic data access applications using gridftp. d. easyapp: run Generic applications performing data gridification. e. easyroot: run Root application performing data gridification. f. easygrid: perform jobs' follow up, recover results in user's directory, and recover crash information for further analysis.

4. How EasyGrid works (users do not need to know it!).

EasyGrid's first task is to find what event files are in the dataset (bookkeeper system), and what WNs have access to them (LFC metadata catalogue).

To manage the files of each dataset, there is the **bookkeeping system**. Physicists can obtain from it a list of dataset file names that match their analysis requirements. These requirements, from grid point of view, are:

a. Select the necessary number of data files that contains some number of events. The number of events defines the processing time and can optimise performance.

b. Select data files from some date. Users want to update their selected events dataset since the last processing with new data, without have to do all over again.

c. Some systems provide a remote query, which allows users know what datasets are available in a remote site, for remote job submission. This is a wise procedure to increase reliability and reduce bottlenecks.

d. Define the initial number of sequence for pointer's data files. Users will be able to have a history directory with all pointer files that already were processed.

The Bookkeeping system is updated every night to guarantee synchronization between all sites and the central experiment.

There are data distribution policies to guarantee redundancy and availability according to demand, geographic distribution and security. **LFC metadata** has a metadata file for each dataset name and its distribution around the world. This method might be also used to store file handlers of files stored in dCache or other file system, providing the link between the logical file name and its physical storage in the remote site. These physical files are registered in the LFC with the SE name.

When EasyGrid submits a job using the clause *InputData* in the JDL file, only the CE with closest SE with data available will be selected.

VO tags describing available software releases and packages complete the necessary information to distribute user's software to CEs for processing.

The list of CEs defines the SEs/NFS that will store analysis software binary and large parametric files to minimize network traffic. EasyGrid performs all necessary procedures to store files remotely and recover them efficiently.

The next stage is generation of all necessary information to submit the jobs on the Grid. **GEnerator of Resources Available** (GERA) produces the Job Description Language (JDL) files, the script with all necessary tasks to run the analysis remotely at a WN, and some grid dependent analysis parameters. The JDL files define the input sandbox with all necessary files to be transferred, and a WN balance load algorithm matches requirements to perform the task optimally.

When the **task is delivered in the WN**, scripts start running to initialize the specific environment, and user's software binary is downloaded from closest the SE. Data files are made available through transfer or providing any access method to the application, and run user's software.

Users can follow up the process querying job status. If the job is done, a task recovering results in the user's directory is performed automatically. If the job was aborted in the process, the diagnostic listing is stored in the history file for further analysis.

EasyGrid was developed using the RAP (Rapid Application Prototyping) methodology. Several versions were developed, covering different approaches and functionalities, and three real applications were used as proof-of-case, which allowed us to evaluate each strategy and acquire information to write the production system specification [22].

5. LCG grid benchmark and performance.

The benchmarks were developed to study data transfer between file providers and applications, test over large numbers of events, and test under large numbers of jobs.

The Grid paradigm states applications should go where data is available. However, there is a potential bottleneck in a site's data distribution architecture. The software will be delivered to a CE with WNs that can access data in some way (NFS, xrootd, gridftp, etc).

The first benchmark was eta(540) [28] reconstruction to test what is the best approach to data distribution: copy data file locally and read the file by application, or use a remote file access such as NFS. The software reads 1.4 gigabytes and produces several histograms for further analysis. Fig 1 shows the performance for different approaches in data access. In Figure 1a data is read directly from the local WN disk, in ideal conditions without overload and traffic. In Figure 1b it is first transferred through a Storage Element. Transferring data produces an iowait in the initial part of the job due channel contention, and reading the data during execution looks better and more efficient.

However, this solution is not scalable as result of network's channel contention (Fig 1 c).

Iowait increases to 50% and cpuload decreases to 50% with performance reduction.

The problem becomes even more significant when many nodes compete to access network (see Table I and II), increasing execution time from 600 s when data is local, to 2522 s with 12 cpus, and 6971 s with 56 cpus. The number of events analysed per second (EPS) decreases, which is a massive waste of resources, and shows the implemented paradigm may not be suitable for data grids because its efficiency is dependent on network availability. Using storage systems such as dCache in the WN [31] are also subject to this potential problem.

Efficiency could be improved if the job submission system submits the jobs directly to specific worker nodes that store specific data files. There would not have file transfer, and the execution would be local with high efficiency. This solution is feasible because HEP data files never change. They become available after reconstruction, and very sporadically will suffer any maintenance. From economical point of view, storage hard drivers are available with hundreds of megabytes, and are used only to store operating system and temporary data (less then 5 Gigabytes of data). However it does require the data to be structured so that jobs will not require data from datasets stored on different nodes, and the job submission system has to know the physical location of the data.

The second benchmark was tau decays to neutral pions. This benchmark selected events over 482 million real events and generated 5 million MC events using EasyGrid [29].

The third benchmark was search for anti-deuterons in all events available in Run 3 (1,500 million events, in one week using 250 computers in parallel) [30].

There were no missing jobs, and few aborts were related with application problems. There were problems in grid catalogue when more then 250 jobs access at once.

6. Functional gridification algorithm.

The gridification algorithm is a library with several functions to run conventional software on the grid doing functional parallelism, with minor changes in the source code. (It is possible at same time to apply data parallelism using EasyGrid.)

The algorithm implements a master / slave architecture. The master manages a task queue that contains elementary tasks each slave can perform independently. One task can store data for a set of individual cases (service string) to overcome problems with communication delays between master/slaves.

The **master software** was implemented using PVM commands [23][24], and can be changed to web services without any problem if necessary.

The first task in the master software is pointer and data structure initialization, to manage the distributed algorithm. It performs the distributed software initialization defining internal characteristics (pvm_setopt function), obtains the total number of slaves available in the cluster (pvm_config function), and starts new slaves processes (pvm_spawn function).

The number of processes depends on the number of machines available in the cluster, and a machine’s cpu load. Sometimes, one computer can host several slaves with optimal performance, combining CPU bound with IO bound services.

After the handshake, the master must confirm a slave’s processes resources availability, keeping record of them to deliver tasks from the queue.

The Master process initializes the application and prepares data buffers with necessary information to run each slave process independently.

The Master initializes the communication channel (pvm_initsend), pack the data (pvm_pkDATA_TYPE, where DATA_TYPE is float, etc), and sends the data to the slaves. Task state and id are stored in the master in case of re-submission.

After all slaves have received the packages, the master starts a probe loop (pvm_probe) in the communication channel, listens for results from slave processes, or waits for timeout from any slave processor. A delay is necessary to avoid a traffic jam in the communication channel, and will contribute to the total communication time between master/slaves. Every cycle a timeout counter is increased.

Tasks are stored in a queue. Every time one slave sends results, it receives a new package of data. This implementation does not require a loading balance algorithm because the master holds all remain processes and provides idle slaves with more tasks.

Every time a new process is sent to one slave, the timeout counter is initialized. If there is a timeout signaling, the task has to be sent to another slave (while there are slaves available) and the slave is marked as out-of-order to avoid

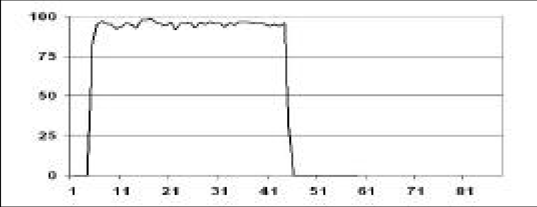
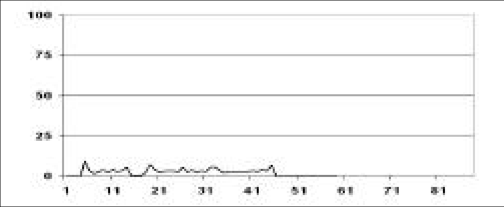
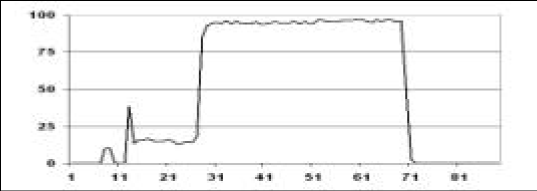
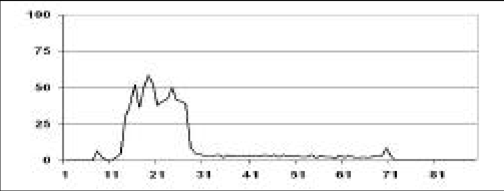
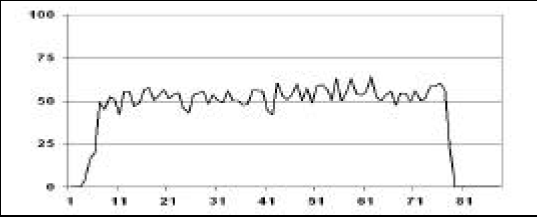
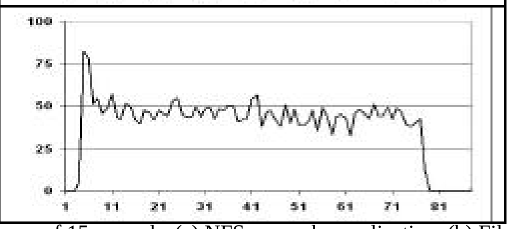
	Percentage of CPU use (%user)	Percentage of IO wait (%wait)
a.NFS 100Mb/s Bf40		
b.SE Bf40		
c.NFS 1Gbs Arthur		

Fig. 1 CPU load and IO wait performance for data transfer paradigm in time frames of 15 seconds. (a) NFS access by application. (b) File copy locally and later access. (c) NFS access with jammed network.

TABLE I DATA DISTRIBUTION ANALYSIS. PERFORMANCE WAS DEFINED AS (100 * EPS/EPS_LOCAL), WHERE EPS IS EVENTS PER SECOND. EPS_LOCAL (1577) IS NUMBER OF EVENTS PER SECOND, USING LOCAL STORED FILES AND TAKES 600 SECONDS.

# Jobs	c.NFS Arthur		a.NFS Bf40		b.SE 100Mb	
	EPS	Perf	EPS	Perf	EPS	Perf
1	855	54	1574	100	1193	76
3	481	31	1457	92	949	60
6	492	31	1388	88	725	46
12	412	26	1372	87	495	31

TABLE II: AVERAGE EXECUTION TIME OF 500 JOBS IN 12 AND 56 CPUS IN PARALLEL WITH FILE TRANSFER AND REMOTE ACCESS.

500 Jobs		File Transfer	Exec	Total
12 CPUs	Time	00:15:00	00:27:00	00:42:02
	Seconds	900	1620	2522
56 CPUs	Time	01:43:52	00:12:19	01:56:11
	Seconds	6232	739	6971

new submissions to it.

Every time new messages come from the channel, the result is stored and a new data string is sent to the slave.

When the task queue is empty, the master does the next cycle in the application and returns to feeding the task queue, repeating all distribution process.

The **slave software** running in each worker node performs the following tasks. First, it get its task id (pvm_mytid), master process (pvm_parent), set distributed environment (pvm_setopt), checks the resources available and prepares the handshake with the master.

While the slave does not receive a package finishing the process, it listens to the communication channel for new data packages. When a new data package is available, it gets the packaged information and type (pvm_bufinfo).

Package labels define what type of task has to be performed by slave process. For example, type 2 is answer master handshake, type 3 is perform some mathematical routine and send results back to master, and type 5 is end of processing.

7. PVM installation with LCG grid middleware.

We have studied gridification algorithm implementations using Parallel Virtual Machine (PVM) package running at LCG worker nodes. System managers should perform the following configuration to use it:

a. Install PVM in a shared NFS system and export directory (/etc/exports) to all computers' farm. Every computer farm should mount PVM directory (auto-mount or /etc/fstab).

b. Set PVM variables in ~/.bashrc: PVM_SSH should point globus ssh (/opt/globus/bin/ssh.d/ssh) and PVM_ROOT should point the mounted NFS directory.

c. All farms computers should have an entry in the ~/.rhosts file.

d. All farms should have a ~/.ssh/authorized_keys file (no passphrase) with all worker nodes public keys (.ssh/id_dsa.pub) created with *ssh-keygen -t dsa*.

e. A copy of compiled master and slave programs should be stored in pvm3/bin/LINUX.

The package is compatible with LCG software and Globus Toolkit 4.

8. Genetic Programming

GP is an optimization algorithm that mimics the evolution and improvement of life through reproduction. Each individual contributes with its own genetic information to the building of new ones (offspring) adapted to the environment with higher chances of surviving. This is the basis of genetic algorithms and programming. Specialized Markov Chains underline the theoretical bases of this algorithm, changes of states and searching procedures.

Chromosome representation. The chromosome represents the model of the problem solution using trees. A tree is a model representation that contains nodes and leaves.

Nodes are mathematical operators. We have used multiplication, addition, subtraction, and division. Leaves are terminals (the attributes of the dataset and random numbers). The discriminator function in a GP context is a tree using operators and leaves (or so called Terminals). Let us consider the following discriminator function:

$$X_1+3.14 \cdot X_2+5.3 / X_3$$

In the tree representation it can be rewritten as following:

$$(+\ X_1\ (+\ (\cdot\ 3.14\ \ X_2)\ (/\ 5.3\ \ X_3)))$$

where X_1, X_2, and X_3 are the terminals, and multiplication(·), addition(+), subtraction (-), and division(/) are the operators. Replacing the values in the equation results in a number that should be positive or negative.

Genetic operators. Trees are manipulated through genetic operators. The crossover operator points a tree branch and exchanges it with another branch and obtains new trees. The mutation operator changes the branch for a random new branch. The length of the chromosome is variable.

The probability of crossover is 60% and the probability of mutation is 20%. We adopt a high value of the mutation probability to spread the population over all solution space.

Fitness function. The Fitness function defines the quality of chromosome as a solution to the problem. It is a numerical positive value. The dataset is divided in two parts: one is for training and the second for validation. The training dataset is used to obtain the model and the validation dataset is used to measure the accuracy of the model with data that was not used in training.

The fitness function evaluates how accurate the mathematical model coded in chromosome is, over all the training dataset counting the number of times the discriminator function is correct.

Receiver Operating Characteristics (ROC) evaluates the accuracy using the number of true negative (N_{TN}), true positive (N_{TP}), false negative (N_{FN}), and false positive (N_{FP}):

$$a = \frac{N_{TP}}{N_{TP}+N_{FN}} \quad b = \frac{N_{TN}}{N_{TN}+N_{FP}} \qquad (1)$$
$$g = \frac{N_{TP}+N_{TN}}{N_{TP}+N_{TN}+N_{FP}+N_{FN}}$$

where a is the *Sensitivity*, ß is the *Specificity*, and γ is the accuracy. Sensitivity is the probability that a test result will be positive when the condition is true (true positive rate, expressed as a percentage). *Specificity* is the probability that a test result will be negative when the condition is false (true negative rate, expressed as a percentage). Accuracy is the probability of correct forecasts.

9. The BaBar High Energy Experiment.

The BaBar experiment [25][26][27] studies the differences between matter and antimatter, to throw light on the problem, posed by Sakharov, of how the matter-antimatter symmetric Big Bang can have given rise to today's matter-dominated universe. High energy collisions between electrons (matter) and positrons (antimatter) produce other elementary particles (tau leptons, pions, kaons, etc), giving tracks and clusters which are recorded by several high granularity detectors and from which the properties of the short-lived particles

TABLE III TRAINING AND TESTS RESULTS FROM DISCRIMINATE FUNCTION OBTAINED USING GENETIC PROGRAMMING WITH DIFFERENT DATASETS. α, β, AND γ DEFINED IN EQUATION (1).

		Case 1:Forecast		Case 2: Forecast		Case 3: Forecast	
Training 57992 records	Real	D>0	D<0	D>0	D<0	D>0	D<0
	1	23299	4819	23368	4750	22491	5627
	0	3093	26781	3040	26834	2731	27143
	γ	86		86		85	
	α	82		83		80	
	β	89		89		90	
Test 302374 records	Real	D>0	D<0	D>0	D<0	D>0	D<0
	1	117268	41037	117215	41090	111999	46306
	0	14153	129916	13870	130199	12543	131526
	γ	81		81		80	
	α	74		74		70	
	β	90		90		91	

can be deduced.

10. Genetic programming gridification to obtain a Neutral Pion discriminator function.

Genetic programming expends most computational effort evaluating fitness functions. Each generation hundreds of individuals have their chromosome decoded into the problem solution that is tested against data. The service we will distribute in grid will be the evaluation, in parallel by many WNs, using Monte Carlo events.

Experimental analysis uses Monte Carlo (MC) generators with particle decays + detector system transfer function. MC events contain all information from each track particle and gamma radiation, which allows event selection for training dataset with no mistakes.

Two datasets were built, one for training with 57,992 records, and one for test with 302,374 records. Events with one real neutral pion were selected and marked as 1. Events without real pions and invariant mass reconstruction in the same region of real neutral pions where also selected and marked 0.

Kinematics data from each gamma used in the reconstruction were written in the datasets: angles of the gamma ray, 3-vector momentum, total momentum, and energy in the calorimeter. To avoid unit problems, we use sine, cosine and tangent values for each angle measured in the genetic trees. All other attributes are measured in GeV (1,000 million electron-volts).

Table III shows the results for training and test of 3 different runs. All results where in agreement and shows high specificity, fundamental to study observable variables from neutral pion particles. High specificity means there will be low non-neutral pions contamination in the sample (less then 10%). Sensitivity of 83% means there will be a lost of 17% of real neutral pions from the sample, with decrease in number and increase of statistical error.

If the large dataset is used in training, the discriminator function obtained by genetic programming is:

D = 3*ener1+ener2+sinteta2+sinteta1-2.5428

And the analysis can be seen in table IV. Accuracy was 82%, sensitivity 81%, and specificity 83% - equation (1).

A better performance could perhaps be obtained by including knowledge of the kinematics of pion decay, but for this analysis we make no such prior assumptions and rely entirely on the training and the algorithm.

11. Functional grid performance.

Table V shows the time expended running standalone and with several numbers of slaves, with good performance: 10 slaves should reduce the time in ideal conditions to 10%, and our implementation achieved 24% despite all necessary communication overheads.

12. Conclusion.

In this paper implementations of data and

TABLE IV TRAINING RESULTS FROM DISCRIMINATE FUNCTION OBTAINED USING GENETIC PROGRAMMING.

		Forecast	
		D>0	D<0
Real	1	129169	29136
	0	24110	119959

TABLE V EXECUTION TIME FOR THE SAME SOFTWARE WITH DIFFERENT NUMBER OF SLAVES AND NODES.

	Standalone	1node / 2 slaves	5 nodes / 10 slaves
Time(1,000s)	80	47	19
Improvement		58%	24%

functional parallelism using LCG/PVM grid environment are discussed and applied for several real case studies. A reliable job submission system (EasyGrid) manages all aspects of integration between user's requirements and resources for data grid. Functional gridification algorithm was implemented in client server architecture with good performance.

All software is available from the Internet [22], and is fully operational and easily adaptable for any application and experiment.

Discriminator functions can be used to discriminate neutral pions from background with 80% accuracy and 91% specificity. This will allow the study of observable and check with values obtained from theoretical Standard Model, such as energy, differential cross section and momentum distribution, from a sample of events with little contamination.

The main bottleneck in data gridification processes is related with data transfer from storage system to the client. To overcome this difficulty, we suggest data files could be stored in a WN disk partition. A catalogue linking files and WN names could be implemented using the approach described in section 4. Files with high demand could be replicated in more than one WN. The job submission system should drive jobs to WNs with the data file available locally, without need of data transfer. This is a major change in LCG approach, but efficiency improvement could justify its development.

The author thanks GridPP and PPARC for funding this project, and the BaBar collaboration for granting access to their data.

REFERENCE

[1] Kameda, H., et al; "Optimal load balancing in distributed computing systems"; Springer, 1996

[2] Fonlupt, C.; Marquet, P.; Dekeyser, J.; "Data-parallel load balancing strategies" Parallel Computing 24 (1998) 1665-1684

[3] Cao, J. et al; "Grid load balancing using intelligent agents" Future generation computer systems 21 (2005) 135-149.

[4] CERN site: http://public.web.cern.ch/Public/Welcome.html

[5] LHC site: http://lhc.web.cern.ch/lhc/

[6] LCG site: http://lcg.web.cern.ch/LCG/

[7] GridPP site: http://www.gridpp.ac.uk/

[8] The GridPP Collaboration: P J W Faulkner et al "GridPP: development of the UK computing Grid for particle physics" 2006 J. Phys. G: Nucl. Part. Phys. 32 N1-N20 doi:10.1088/0954-3899/32/1/N01

[9] Holland,J.H. "Adaptation in natural and artificial systems: na introductory analysis with applications to biology, control and artificial intelligence." Cambridge: Cambridge press 1992.

[10] Goldberg,D.E. "Genetic Algorithms in Search, Optimisation, and Machine Learning." Reading, Mass.: Addison-Whesley, 1989.

[11] Chambers,L.; "The practical handbook of Genetic Algorithms" Chapman & Hall/CRC,2000.

[12] Koza,J.R. "Genetic programming: On the programming of computers by means of natural selection." Cambridge,Mass.: MIT Press, 1992.

[13] Werner,J.C.; "Active noise control in ducts using genetic algorithm" PhD. Thesis- São Paulo University-São Paulo-Brazil-1999.

[14] Cranmer,K.; Bowman,R.S.; "PhysicsGP: A genetic programming approach to event selection" Computer Physics Communications 167 (2005) 165-176.

[15] Focus Collaboration, "Application of genetic programming to high energy physics event selection" Nuclear instruments and methods in physics research A 551 (2005) 504-527.

[16] Focus Collaboration; "Search for L+c -> pK+p- and D+s -> K+K+p- using genetic programming event selection" Physics letters B 624 (2005) 166-172

[17] Mjahed, M.; "Search for Higgs boson at LHC by using genetic algorithms" To be published in Nuclear Instruments and Methods in Physics Research.

[18] Kalganova,T.; Karol, I.M.; Werner,J.C.; Silkou,N.I.; Lipnitskaya,N.G.; Probability prediction method of throat cancer with use of discriminate function (in Russian) 2nd International Belarusian-Polish Conference on Otorhinolaryngology: Actual Problems in Otorhinolaryngology, Grodno, 29-30 May 2003

[19] Werner,J.C.; Kalganova,J.C.; Disease modeling using Evolved Discriminate Function. LNCS 2610, Proceedings 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003.

[20] Werner,J.C.; Fogarty,T.C.; Severe diseases diagnostics using Genetic Programming. Intelligent Data Analysis in medicine and pharmacology IDAMAP2001; September 4th, 2001 London

[21] Werner,J.C.; Fogarty,T.C.; Genetic programming applied to Collagen disease & thrombosis. PKDD 2001 Challenge on Thrombosis data Germany/ Freiburg September 3-7

[22] Werner,J.C.; "HEP analysis, Grid and EasyGrid Job Submission Prototype: Babar/CM2 showcase" at http://www.hep.man.ac.uk/u/jamwer/

[23] Parallel Virtual Machine site: http://www.csm.ornl.gov/pvm/pvm_home.html

[24] Geist,A. et al; "PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing" MIT Press, 1994 available from http://www.netlib.org/pvm3/book/pvm-book.html

[25] BaBar Collaboration, "The BaBar experiment home page", http://www.slac.stanford.edu/BFROOT/

[26] Harrison,P.F.; Quinn,H.R.; The BaBar Physics Book" SLAC Report 504, October 1998, available at http://www.slac.stanford.edu/pubs/slacreports/slac-r-504.html

[27] BaBar Collaboration; "The BaBar detector", Nuclear Instruments and Methods in Physics Research A479(2002) 1-116 available at http://www.hep.man.ac.uk/u/jamwer/babarnucl.pdf

[28] Tavera, M.; Private communication on eta reconstruction from TauUser data, PhD Thesis.

[29] Werner,J.C.; "Neutral Pion project" http://www.hep.man.ac.uk/u/jamwer/pi0alg5.html

[30] Werner,J.C.; "Search for anti-deuteron" http://www.hep.man.ac.uk/u/jamwer/deutdesc.html

[31] Forti, A.; "Cluster distributed dynamic storage" CHEP06, Mumbai, India