

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. For functional parallelism a PVM algorithm ran user's software in parallel as a master / slave implementation. These approaches were applied to typical particle physics applications: hadronic tau decays, searching for anti-deuterons, and neutral pion discrimination using genetic programming algorithms. Studies of performance show considerable reduction of time execution using functional gridification.

1. INTRODUCTION

The GridPP collaboration [1][2] and several other e-science projects have provide a distributed infrastructure and software middleware in UK. The LCG (Large Hadrons Collider Computing Grid) [3][4][5] software, developed by an international collaboration centered at CERN, provides a system for batch processing for High Energy Physics (HEP) through hundreds of computers connected by the Internet. It can be seen as a homogeneous common ground in a heterogeneous platform.

The **testbed farm** available at University of Manchester [6] contains 1 resource broker, 1 information system, 1 storage element (SE) with 7GB, 1 computer element (CE), and 6 worker nodes. The **production farm** contains 1 CE and 28 WNs, sharing other systems with the testbed. It is used to run analysis software from BaBar data.

The BaBar experiment [7] 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.

This paper describes the approach for data gridification using EasyGrid (section 2), and for functional gridification (section 3) using real HEP analysis cases as benchmarks, followed by the conclusions.

2. Data gridification.

HEP Data Analysis can be divided in several hundreds of independent jobs running the same binary code in parallel over each dataset's data

file with thousands of million of events.

Data gridification is implemented through EasyGrid Job Submission [6] 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. easyvapp: 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.

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. 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.

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 that will store analysis software binary and large parametric files to minimize network traffic. EasyGrid performs all necessary procedures to replicate these 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 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.

Three benchmarks were developed. The first benchmark was eta(547) [8] 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 from 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.

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 [9].

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) [10].

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

3. Functional gridification algorithm.

Functional parallelism 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.

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 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. When one WN returns the results for a task, another task from the queue is send for processing.

The software was implemented using PVM commands [11], and can be changed to web services without any problem if necessary.

Functional gridification benchmark was genetic programming applied to evolve neutral pion discriminate function.

Genetic Programming (GP) [12]-[16] 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). Each fitness evaluation is an independent task to the gridification algorithm.

Genetic programming has been used to

determinate cuts to maximize event selection [17]-[19]. Genetic algorithms can also be associated with neural networks to implement discriminate functions [20] for Higgs boson.

Our approach is innovative because the mathematical model obtained with GP maps the variables hyperspace to a real value through the discrimination functions, 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. Otherwise, the pair is deemed to come from another (background) source.

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 were 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 Ge V (1,000 million electron-volts).

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

Table IV 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.

Conclusion.

In this paper implementations of data and 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 [6], and is fully operational and easily adaptable for any application and experiment.

Discrimination functions can be used to discriminate neutral pions from background with 80% accuracy and 91% purity. This will allow me compare experimental values with observable obtained from theoretical Standard Model.

REFERENCE

- [1] GridPP site: <http://www.gridpp.ac.uk/>
- [2] 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
- [3] CERN site: <http://public.web.cern.ch/Public/Welcome.html>
- [4] LHC site: <http://lhc.web.cern.ch/lhc/>
- [5] LCG site: <http://lcg.web.cern.ch/LCG/>
- [6] Werner, J.C.; "HEP analysis, Grid and EasyGrid Job Submission Prototype: Babar/CM2 showcase" at <http://www.hep.man.ac.uk/u/jamwer/>
- [7] BaBar Collaboration; "The BaBar detector", Nuclear Instruments and Methods in Physics Research A479(2002) 1-116.
- [8] Tavera, M.; Private communication on eta reconstruction from TauUser data, PhD Thesis.
- [9] Werner, J.C.; "Neutral Pion project" <http://www.hep.man.ac.uk/u/jamwer/pi0alg5.html>
- [10] Werner, J.C.; "Search for anti-deuteron" <http://www.hep.man.ac.uk/u/jamwer/deutdesc.html>
- [11] Parallel Virtual Machine site: http://www.csm.ornl.gov/pvm/pvm_home.html
- [12] Holland, J.H. "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence." Cambridge: Cambridge press 1992.
- [13] Goldberg, D.E. "Genetic Algorithms in Search, Optimisation, and Machine Learning." Reading, Mass.: Addison-Wesley, 1989.
- [14] Chambers, L.; "The practical handbook of Genetic Algorithms" Chapman & Hall/CRC, 2000.
- [15] Koza, J.R. "Genetic programming: On the programming of computers by means of natural selection." Cambridge, Mass.: MIT Press, 1992.
- [16] Werner, J.C.; "Active noise control in ducts using genetic algorithm" PhD. Thesis- São Paulo University- São Paulo- Brazil- 1999.
- [17] Cranmer, K.; Bowman, R.S.; "PhysicsGP: A genetic programming approach to event selection" Computer Physics Communications 167 (2005) 165-176.
- [18] Focus Collaboration, "Application of genetic programming to high energy physics event selection" Nuclear instruments and methods in physics research A 551 (2005) 504-527.
- [19] Focus Collaboration; "Search for $L+c \rightarrow pK+p-$ and $D+s \rightarrow K+K+p-$ using genetic programming event selection" Physics letters B 624 (2005) 166-172
- [20] Mjahed, M.; "Search for Higgs boson at LHC by using genetic algorithms" To be published in Nuclear Instruments and Methods in Physics Research.

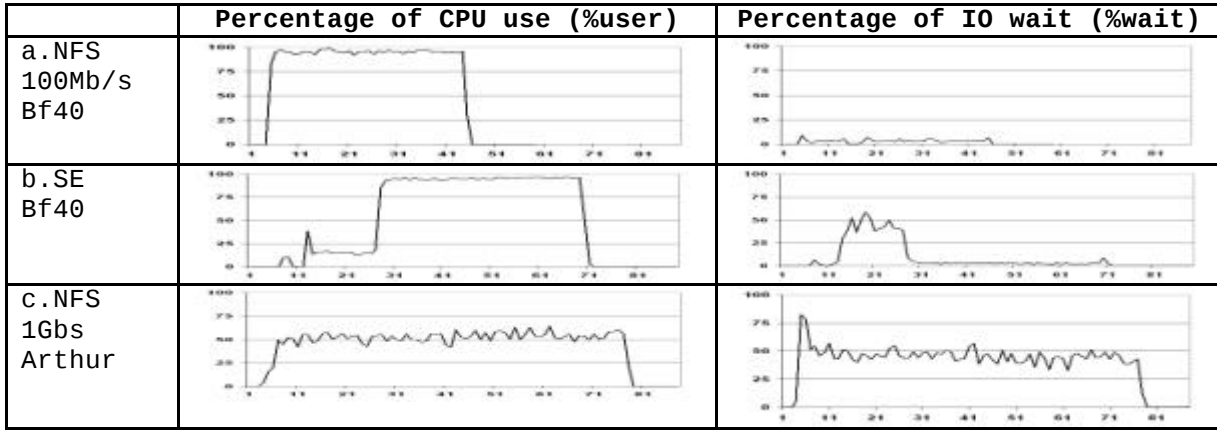


Fig. 1 CPU load and IO wait performance for data transfer paradigm in time frames of 15 seconds. (a) NFS access by application. (b) File copy locally and later access. (c) NFS access with jammed network.

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 1 Gbps		a.NFS		b.SE 100Mbps	
	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

TABLE III TRAINING AND TESTS RESULTS FROM DISCRIMINATE FUNCTION OBTAINED USING GENETIC PROGRAMMING WITH DIFFERENT DATASETS.

	Real	Case 1: Forecast		Case 2: Forecast		Case 3: Forecast	
		D>0	D<0	D>0	D<0	D>0	D<0
Training 57992 records	1	23299	4819	23368	4750	22491	5627
	0	3093	26781	3040	26834	2731	27143
	Accuracy	86		86		85	
	Efficiency	82		83		80	
	Purity	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
	Accuracy	81		81		80	
	Efficiency	74		74		70	
	Purity	90		90		91	

TABLE IV 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%