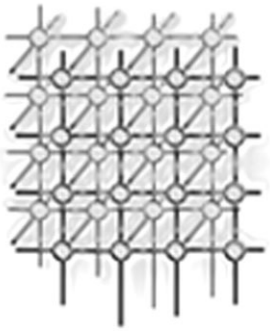
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **17**:1783–1798
Published online 28 June 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.888



Neuroscience instrumentation and distributed analysis of brain activity data: a case for eScience on global Grids

Rajkumar Buyya[1,*,†], Susumu Date[2], Yuko Mizuno-Matsumoto[3], Srikumar Venugopal[1] and David Abramson[4]

[1]*Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia*
[2]*Graduate School of Information Science and Technology, Department of Bioinformatics Engineering, Osaka University, Japan*
[3]*Department of Information Systems Engineering, Graduate School of Osaka University, Japan*
[4]*School of Computer Science and Software Engineering, Monash University, Melbourne, Australia*

SUMMARY

The *distribution* of knowledge (by scientists) and data sources (advanced scientific instruments), and the *need* for large-scale computational resources for analyzing massive scientific data are two major problems commonly observed in scientific disciplines. Two popular scientific disciplines of this nature are brain science and high-energy physics. The analysis of brain-activity data gathered from the MEG (magnetoencephalography) instrument is an important research topic in medical science since it helps doctors in identifying symptoms of diseases. The data needs to be analyzed exhaustively to efficiently diagnose and analyze brain functions and requires access to large-scale computational resources. The potential platform for solving such resource intensive applications is the Grid. This paper presents the design and development of MEG data analysis system by leveraging Grid technologies, primarily Nimrod-G, Gridbus, and Globus. It describes the composition of the neuroscience (brain-activity analysis) application as parameter-sweep application and its on-demand deployment on global Grids for distributed execution. The results of economic-based scheduling of analysis jobs for three different optimizations scenarios on the world-wide Grid testbed resources are presented along with their graphical visualization. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: Grid computing; eScience; neuroscience; Gridbus middleware; Nimrod-G; virtual instrumentation

*Correspondence to: Rajkumar Buyya, Department of Computer Science and Software Engineering, The University of Melbourne, ICT Building, 111, Barry Street, Carlton, Melbourne, VIC 3053, Australia.
†E-mail: raj@cs.mu.oz.au

Received 10 September 2003
Revised 5 February 2004
Accepted 9 March 2004

1. INTRODUCTION

The emergence of high-speed networks has made it possible to share geographically distributed resources such as supercomputers, storage systems, databases and scientific instruments in order to gather, process and transfer data smoothly across different administrative domains. Aggregations of such distributed resources, called computational Grids [1], provide computing power that has made it possible to solve large-scale problems in science, engineering and commerce. Biological sciences have several computational and data-intensive problems that can be tackled better by utilizing large-scale distributed computing and storage resources. The analysis of data gathered by monitoring activity in the brain is one such problem.

Brain activity analysis

Brain activity is measured by the magnetoencephalography (MEG) instrument which measures the magnetic fields generated by the electrical activity in the brain. This method is more accurate than others such as electroencephalography (EEG) and electrocorticography (ECoG) [2]. Another advantage of MEG is that it is non-invasive.

The MEG instrument consists of a number of sensors which record information about brain activity. Currently, MEG helmets with over 200 sensors are already used to detect magnetic brain fields by means of a sensitive transducer technology called the Superconducting Quantum Interference Device (SQUID) [3]. This provides a doctor with a host of data offering the finest temporal and the highest spatial resolution. Specialists can detect a disorder by observing the complex brain wave form and analyzing the frequency content. The doctor has to separate the MEG data into signal classes. Each of these contains a certain frequency band which allows the localization of the signal's source. To this end, wavelet cross-correlation analysis, developed at the Osaka University by Dr. Mizuno-Matsumoto *et al*. [4], is used. Unlike the traditional Fourier-based analysis, wavelet-based analysis has the capability to explore the frequency content without losing the time information of the original brain data. This approach requires analysis of data from each pair of sensors. This analysis lacks the time information of the original brain data, but provides the similarity between a pair of brain data for every frequency spectrum. By focusing on a spectrum, one could track a signal that is within the spectrum. This enables the mapping of brain activity to patient behavior and is useful in diagnosing several diseases.

However, there are certain problems with MEG. Due to the high cost of equipment and the required shielding against stray magnetic fields, there are only limited numbers of MEG instruments around the world. Plus, the instrument generates a huge amount of data, all of which are not analyzed due to lack of computing resources. For example, a 64-sensor MEG instrument would produce 0.9 GB of data over a period of an hour. Such a task generates 7 257 600 analysis jobs and would take 102 days on a commodity computer with a Pentium III 500 MHz processor and 256 MB of memory. Proper recovery of the patient depends on the results being available as soon as possible so that the doctor can start the drug regimen in the shortest possible time. While it is possible to use a supercomputer to analyze the brain data, every medical facility cannot be expected to have access to such a machine. Also, access to a much larger pool of resources and, consequently, larger amounts of computational power is possible if the application can be deployed on the Grid. Furthermore, there is also the advantage that the medical data can be shared easily among the partnering doctors. These requirements make

a clear case for considering it as an eScience application that calls for Grid-enabled global scientific investigation through seamless and secure integration and utilization of distributed resources such as medical instruments, computational resources, datasets, application components, and people (domain experts, medical doctors, and also patients in this case).

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the flow diagram for the analysis. Section 4 describes the architecture for the execution. Section 5 explains the process of Grid-enabling this application. The scheduling experiments and their results are detailed in Section 6 and the final section summarizes the paper along with suggestions for future work.

2. RELATED WORK

There are number of paradigms and platforms, such as message passing (using MPI [5] or PVM [6]), for developing and executing parallel applications on distributed systems. However, using such programming paradigms/platforms for creating parallel/distributed applications involves significant software engineering effort. These approaches are effective for tightly coupled systems such as massively parallel processing (MPP) machines or on loosely coupled but controlled systems such as clusters. The inherent challenges in Grid computing environments such as the load volatility, high network latencies and high probability of failure of individual nodes make it difficult to adopt a programming approach which favours tightly coupled systems. The parameter sweep model of computation is considered to be ideal as it allows the composition of applications for distributed execution just by parameterizing their input data files and they can be easily deployed on global Grids using Grid-scheduling systems such as Nimrod-G [7]. The Grid enabling of this application as a parameter sweep application, as described in this paper, took about three weeks of effort whereas a parallel implementation [2,8] using MPICH-G [9] consumed over six months of development effort.

There have been various efforts in Grid-enabling different applications. They include AppLeS parallel tomography [10], FightAIDS@Home [11] and Cactus [12]. AppLeS approach for Grid-enabling applications involves the development of an application-specific scheduler, whereas our work separates the task of *composition* of the application for processing on the Grid from the *scheduling* system. This means that for a number of applications with the same processing model, we can utilize the same scheduling system for deploying them on global Grids. FightAIDS@Home is based on Entropia's distributed computing network and Scripps Research Institute's docking application. In this system, volunteers need to download Entropia's screen saver program that runs in the background on the volunteer computer. The volunteer PC contacts the Entropia server to download the data to perform docking. When docking on an assigned data is completed, it uploads the results to the server. This execution model is different from our model where the scheduler (Nimrod-G) assigns the work to computers that are available and initiates the execution. Cactus is a framework for the numerical solution of Einstein's equations and it is Grid enabled by implementing it as a MPI application.

3. GRID-BASED ANALYSIS MODEL

The NeuroGrid project aims to convert the existing brain activity analysis application into a parameter sweep application for executing jobs that perform wavelet cross-correlation analysis for each pair of

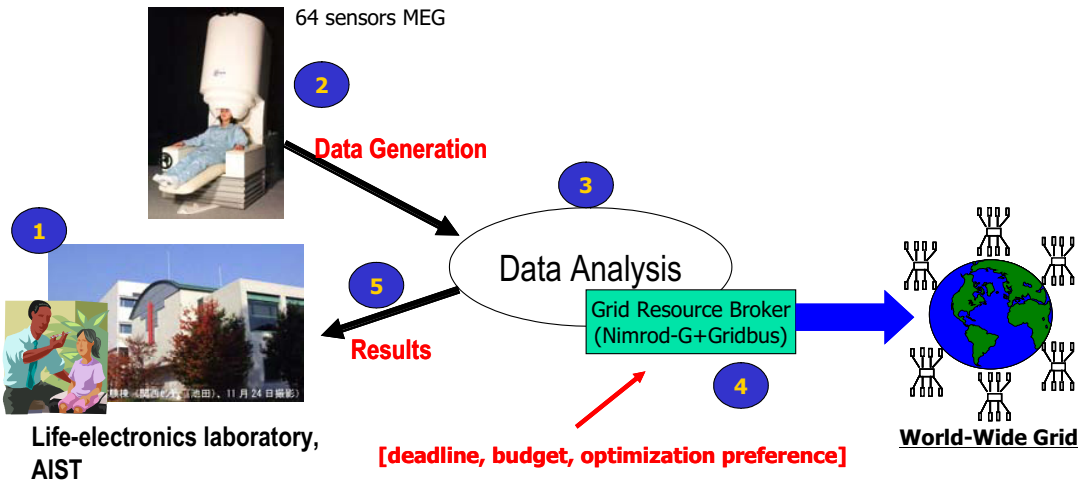


Figure 1. A model for brain-activity analysis on global Grids.

sensors in parallel on distributed resources. The jobs are both computationally and data intensive. Also they are independent of each other, thus making them perfect for being composed as a parameter sweep application. The flow diagram for distributed analysis of brain activity is shown in Figure 1.

The medical staff who are dealing with the diagnosis order a MEG scan of the patient's brain (step 1). The request is sent to the instrument, which takes a MEG scan and collects data about the activity in the brain (step 2). This data is then collated and presented to the Grid resource broker for analyzing on the Grid (step 3). The broker discovers the resources and looks up in the Grid market directory (GMD) for the corresponding service costs associated with those resources. Using the user-defined quality-of-service (QoS) parameters, two of which are the deadline and the budget, the scheduler distributes the jobs based on the optimization method chosen. The optimization method could be one of three: cost, time or cost–time. The data and the analysis code are dispatched to the remote node and the results are collected (steps 4 and 5).

4. ARCHITECTURE

The architecture followed in this project is shown in Figure 2. It consists of the brain-activity analysis application, parameterization tools (Nimrod-G parameter specification language), resource broker (Nimrod-G with Gridbus scheduler), GMD (Gridbus), and low-level Grid middleware (Globus [13]). The resources are Grid-enabled using Globus software deployed on them. The application has been Grid-enabled by composing it as a parameter sweep application using the Nimrod-G parameter specification language. The GMD has been used as a register for the publication of resource providers and services.

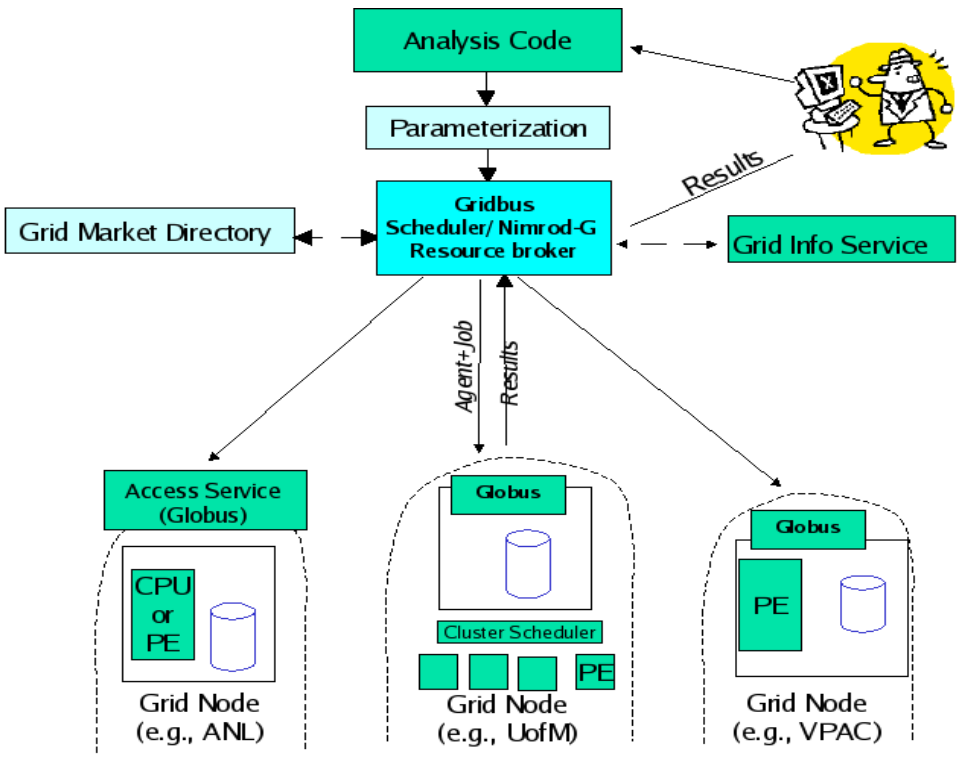


Figure 2. Architecture of brain-activity analysis on the Grid.

4.1. Analysis code

The analysis code was developed by the Cybermedia Centre, Osaka University, Japan. The raw data obtained from the sensors in the MEG instrument is analyzed in two phases as shown in Figure 3. In the first phase, the raw data from the brain goes through the wavelet transform operation. This phase gives the time-frequency data of the output. In the next phase, cross-correlation analysis is performed for each pair of wavelet transforms. This output displays the similarity between a pair of brain data for every frequency spectrum. By focusing on a specific frequency spectrum, the sensor that first detected the signal with that focusing spectrum can be localized. This means that the path of a signal can be found as it travels through the brain. The code was written in C and is highly portable. We have been able to compile it for different platforms such as PC/Linux, Sun/Solaris, SGI/Irix and DEC/Alpha.

4.2. Grid resource broker and scheduler

For executing our application on the Grid, a combination of the Nimrod-G [7] resource broker developed using the Globus [13] middleware and our own Gridbus scheduler was used.

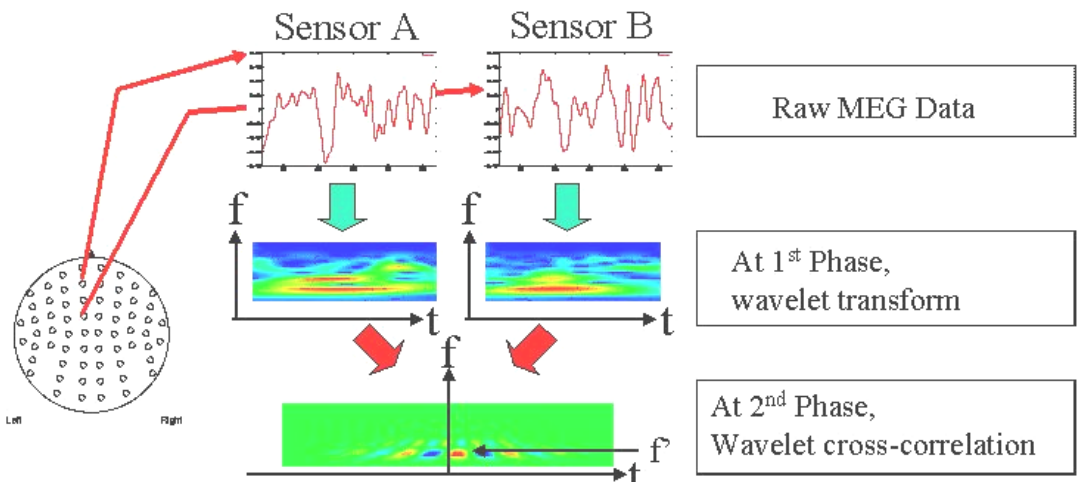


Figure 3. Wavelet cross-correlational analysis.

The Nimrod-G resource broker identifies the user and application processing requirements and selects the combination of Grid resources in such a way that the user requirements are met. It performs resource discovery, selection, and dispatching of MEG jobs to remote resources. It also starts and manages the execution of jobs and gathers the results back at the home node. The following components of Nimrod-G have been used in our experiments:

- a persistent task-farming engine;
- a Grid explorer for resource discovery.

The Gridbus scheduler developed as a plugin scheduler for Nimrod-G has been used instead of the default Nimrod-G scheduler since it has been designed to utilize the GMD [14]. The default Nimrod-G scheduler calculates the processing cost based on the CPU time that is used to execute a job on a remote node. To support the notion of application services and pricing based on application operation (AO) instead of vanilla CPU service, the GMD already allows the Grid service providers (GSPs) to publish application services along with their AO service price. An AO service price describes how much it costs to execute an application once on a remote host regardless of the amount of resources it takes. The Gridbus scheduler has been enabled to utilize the GMD services and perform resource allocation based on AO cost model. In this model, the user is charged a price for execution of each job on the resource. Thus, the resource owner may offer the application as a service and charge a fixed price for executing it. By periodically updating the resource information from the GMD, the Gridbus scheduler ensures that the scheduling is done based on the most recent cost-price for the resources.

The Gridbus scheduler implements three algorithms: cost minimization, time minimization and cost–time optimization. All three algorithms are constrained by two parameters: the deadline by which the experiment is required to complete and the budget that the user has. Time minimization tries to execute

the project within the shortest time while keeping within the budget. Cost minimization tries to complete the execution with the least cost while keeping to the deadline. Cost–time optimization gives jobs to the cheapest servers, but performs time optimization among those. Cost–time optimization was only simulated until recently and it has been implemented for the first time in the Gridbus scheduler. More information about these scheduling algorithms can be found in [15,16].

4.3. GMD

The GMD [14], developed by the Gridbus project in the University of Melbourne, allows service providers to publish their services along with the costs associated with them. The consumers will be able to browse the GMD to identify services that meet their requirements. The GMD is developed using standard Web services technologies such as the simple object access protocol (SOAP) and XML, therefore it can be queried by other client programs. To provide an additional layer of transparency, a client application programming interface (API) has been provided that could be used by programs to query the GMD without the developers having to concern themselves with the details of SOAP. The GMD infrastructure was used to maintain a registry of participants along with the details of their contributed resources as part of the global Grid testbed [17] collaboration. The Gridbus scheduler interacts with the GMD to discover the testbed resources and their high-level attributes such as access price.

5. GRID-ENABLING THE APPLICATION

The existing analysis code was composed as a task-farming, parameter sweep application for execution on Grids using the Nimrod-G parameter specification language [7]. The Nimrod-G farming engine and dispatcher along with Gridbus scheduler is used for deploying and processing it on global Grids.

The brain activity analysis software suite, developed at the Cybermedia Centre at Osaka University, consists of two separate programs. The first, *raw2wavelet*, is a program that performs wavelet transform over the raw data and the second, *wavelet2cross*, is a program that does the cross-correlation between the wavelet transforms. A job consists of executing these two programs in sequence for a pair of unique sensors and for a particular offset. For 64 sensors and an offset ranging over zero to 29 750, this would generate $(64 \times 63)/2 \times 29\,750$ jobs. Although this application contains numerous jobs, each individual job is fine-grained in nature—in certain data scenarios it can be less than a minute. The overhead associated with initiating each task on a separate node and collecting its results after it has finished execution would have radically decreased the efficiency of distributed execution. So, coarse-grained jobs had to be created by grouping the fine-grained jobs so that the computation to I/O ratio would work in favour of distributed analysis. For this purpose, a meta-program (*metameg*) was written that would perform pairwise analysis for all sensors for a particular range of offsets. The entire temporal shift region was divided into offset ranges. A job would consist of performing analysis over a temporal shift range for all the sensors. The pseudo-code for this meta-job program, called metameg, is listed in Figure 4.

The variable **time_offset_step** decides the size of the meta job as it divides the offset range into regions. If it is 1, then a job is the wavelet cross-correlation analysis for all sensors for one particular offset. If it is same as the limit of the temporal shift region (i.e. maximum offset), then it represents

```
For time_offset from time_t1 to (time_t1+time_offset_step)
begin
    For sensor_A from 1 to max_sensors
    begin
        For sensor_B from 1 to max_sensors
         begin
                if sensor_A NOT EQUALS sensors_B then
                begin
                    Execute raw2wavelet sensor_A sensor_B time_offset meg_data_path
                    Execute wavelet2cross sensor_A_output sensor_B_output
                end
         end
    end
end
```

Figure 4. Pseudo-code for the meta program.

an aggregation of all the jobs. As the size of the meta job increases, the number of jobs generated decreases according to the following equation:

$$\text{Meta job size} = \frac{\text{Maximum offset}}{\text{Number of meta jobs for Grid analysis}}$$

The executables of the three programs that represented the analysis code (metameg, raw2wavelet, and wavelet2cross implemented in C language) were small in size and hence easily transferable at runtime if the code is not deployed on remote resources. Each meta job requires the full raw data captured by all (64) sensors. However, transferring this raw data, over 24 MB, to each remote node during the execution of any job would constitute a significant I/O load over wide-area networks. Unless an analysis is carried out on the MEG data with a few sensors generating small amounts of data, it is preferable to pre-stage the entire raw dataset at each node. We also had to change the meta program (metameg) and the raw data to wavelet conversion (raw2wavelet) program to enable access to data on a path on the remote node.

A Nimrod-G plan for the single-program multiple-data (SPMD) style execution of the above analysis operation as parametric execution is shown in Figure 5. As the Grid comprises heterogeneous resources with different architectures, we have compiled the analysis code for different architectures and renamed the executables with an extension same as the operating system (OS) name. This means that any reference to the program with .$OS in the plan file indicates that the Nimrod-G broker is able to select the correct executable depending on the target architecture automatically at runtime. $HOME represents the home directory of the user at the remote node and the directory $HOME/alphawave is where the brain data is stored on the remote machine. As the data passed around has no information related to patient identification, there is no need to handle privacy issues explicitly.

The output of the each pair-wise analysis is represented in image data, which can be further studied using a visualization program jointly developed by the Osaka University and NEC, Japan. All the output files of meta job are grouped (archived using the tar command) and transferred back to the remote node.

```
parameter meg_sensors_count label "no. of MEG sensors" integer default 64;
parameter offset_max=29750;
parameter MetaJobSize label "time_offset step size" integer default 10;
parameter time_offset label "MEG Data Temporal Region Shift" integer range from 0 to offset_max step
MetaJobSize;
task nodestart
    #copy meg_data.tar node:$HOME/alphawave
    copy raw2wavelet.$OS node:raw2wavelet
    copy wavelet2cross.$OS node:wavelet2cross
    copy metameg-data-path.$OS node:metameg
endtask
task main
    node:execture./metameg-data path $time_offsets $MetaJobSize $meg_sensors_count $HOME/alphawave
    node:execute $HOME/tar cvf output.tar *.ase *.ppm
    copy node:output.tar output.tar.$jobname
endtask
```

Figure 5. Plan file for brain activity analysis on the Grid.

6. APPLICATION DEPLOYMENT AND EVALUATION

The brain activity analysis was demonstrated at *Supercomputing 2002* held in Baltimore, MD, U.S.A. We used the resources provided by the Supercomputing 2002 Global Grid Testbed Collaboration [17], which we were part of, plus those of the World Wide Grid [15] (see Figure 6) to schedule analysis jobs on globally distributed resources. After the *Supercomputing 2002 Conference*, most of the resources that were part of this testbed have discontinued their participation as the original objective has been achieved. This illustrates the ever changing nature of collaborations on the Grid. Different organizations are able to come together and share their resources to achieve an objective and thus create virtual organizations [18].

However, at the time of writing this paper (at the end of January 2003), we continued to have access to some of the testbed resources. We utilized these resources for our analysis experiments whose results are reported in this paper. The list of machines that we have used to carry out our experiments is shown in Table I.

In Table I, G$ stands for Grid dollars, the unit used to indicate the cost of leasing a CPU for one second in a dedicated mode. G$ can be thought of as tokens that are allocated in HPC centre for its users who can then spend them while executing their applications on supercomputers. The number of tokens charged for consumption of CPU power per second is determined by the cost of providing resources. However, in the experiment, the Grid node access cost in G$ is made out artificially and aimed at reflecting the differentiated pricing for services similar to the real world marketplace. For example, pricing the user or collaborator owned resources lower than the external resources to encourage the higher utilization of local resources. The experiments were carried out on Thursday 30th January 2003, from 10:00 am to 1:00 pm AEDT. The three scheduling algorithms were tried out and their results tabulated. The experiment was performed for two sensors, with a maximum offset of 100 and

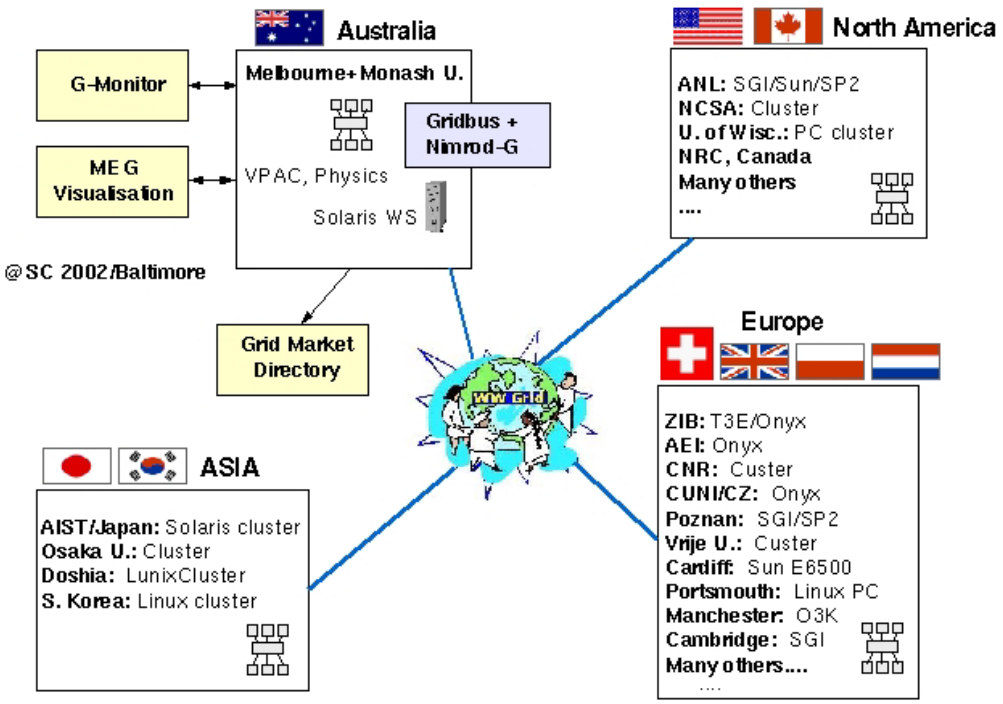


Figure 6. Brain-activity analysis at the HPC Challenge Demo at *Supercomputing 2002*.

Table I. Grid resources utilized during the analysis of brain activity data.

Organization	Node details (architecture, number of nodes, hostname)	Relative strength	G$/CPU (s)
University of Melbourne, Australia	Linux, *broker machine*, lem.ph.unimelb.edu.au	N/A	N/A
Vrije Universiteit, Netherlands	Linux Cluster, 144 nodes (32 available), fs0.das2.cs.vu.nl	4	2
N*Grid Project, Korea	Linux Cluster, 24 nodes, node1001.gridcenter.or.kr	3	3
N*Grid Project, Korea	Linux Cluster, 16 nodes, node2001.gridcenter.or.kr	2	1
Osaka University, Japan	Linux Cluster, 2 nodes, date1.ics.es.osaka-u.ac.jp	1	1

Table II. Summary of scheduling experiment statistics.

Scheduling strategy	Start time (ST)	Completion time (CT)	Elapsed time (CT − ST)	Budget utilized (G$)
Time	10:00 am	10:29 am	29 min	399
Cost	10:35 am	11.54 am	79 min	204
Cost–time	12:10 pm	12.52 pm	42 min	330

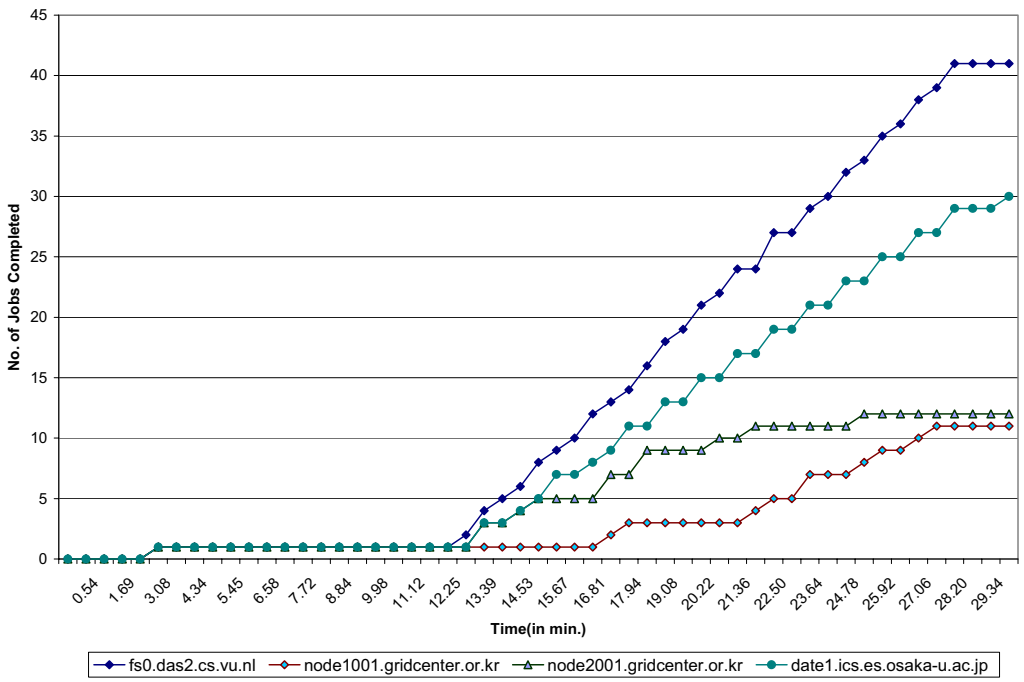


Figure 7. Scheduling with time minimization: cumulative graph of the number of jobs completed versus time.

meta-job size of one which produced 100 jobs. All the experiments were started with a deadline of 2 h, and a budget of G$1990. The summary of the results of these experiments is shown in Table II.

The graph in Figure 7 shows the progress of the execution using time minimization scheduling algorithm. As is depicted, the scheduler makes use of all the resources to ensure that the experiment completes in the fastest possible time. Most of the jobs were executed by the fastest machine (fs0.das2.vu.nl). However, it was also not the cheapest machine and so the cost of computation increased.

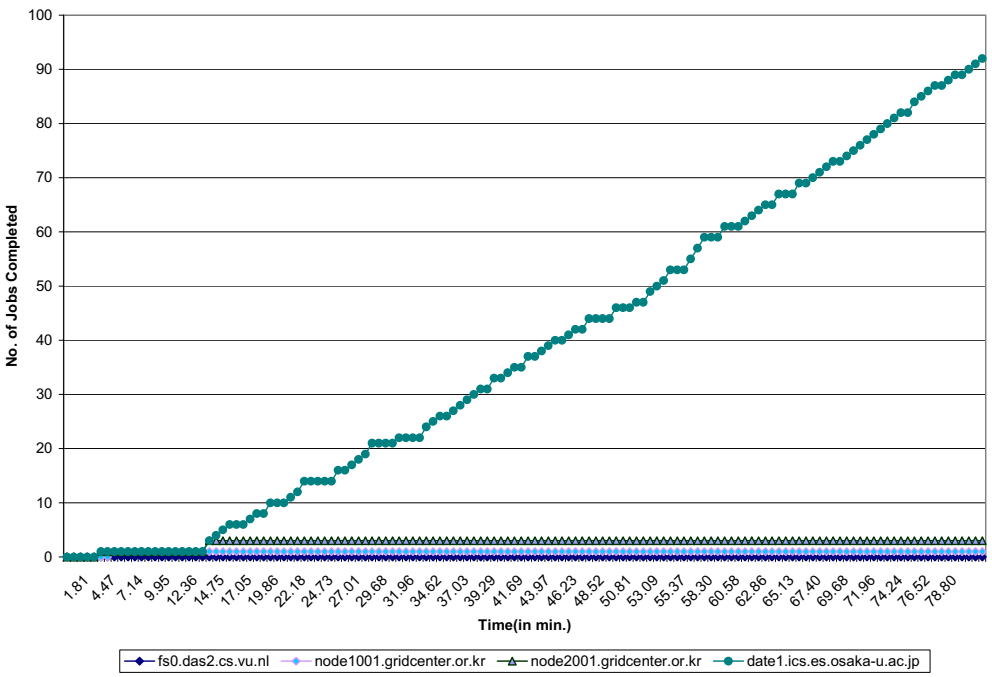


Figure 8. Scheduling with cost optimization: cumulative graph of the number of jobs completed versus time.

The graph in Figure 8 is of the same experiment using cost minimization scheduling. Here, most of the load is borne by one of the cheapest machines (date.ics.es.osaka-u.jp). The scheduler allocates all the jobs to the machine with the least cost to minimize the cost of computation. However, in the beginning, it does give some jobs to the faster, expensive machines before it can be sure that the Osaka University machine will be able to execute the jobs within the specified deadline.

The graph in Figure 9 relates to the experiment performed with cost–time optimization. Here, most of the computation was handled by the cheapest nodes (node1001.gridcenter.or.kr and date1.ics.es.osaka-u.jp). However, as these two were initially busy, the scheduler allocated the maximum number of jobs to the expensive machines. As the availability of the other two improved, the share of jobs given to the costly machines decreased as jobs were moved to the cheaper resources. The graph does not show the behavior expected of the algorithm that was observed in simulation [15]. This underscores the volatility of the Grid environment due to the variation in factors such as bandwidth availability, machine load, etc.

The results of our scheduling experiments show that it is feasible to conduct the analysis using the parameter sweep model of distributed computing. Furthermore, this approach allows medical personnel to determine the pace of the analysis based on the urgency of their requirements. If the urgency is low,

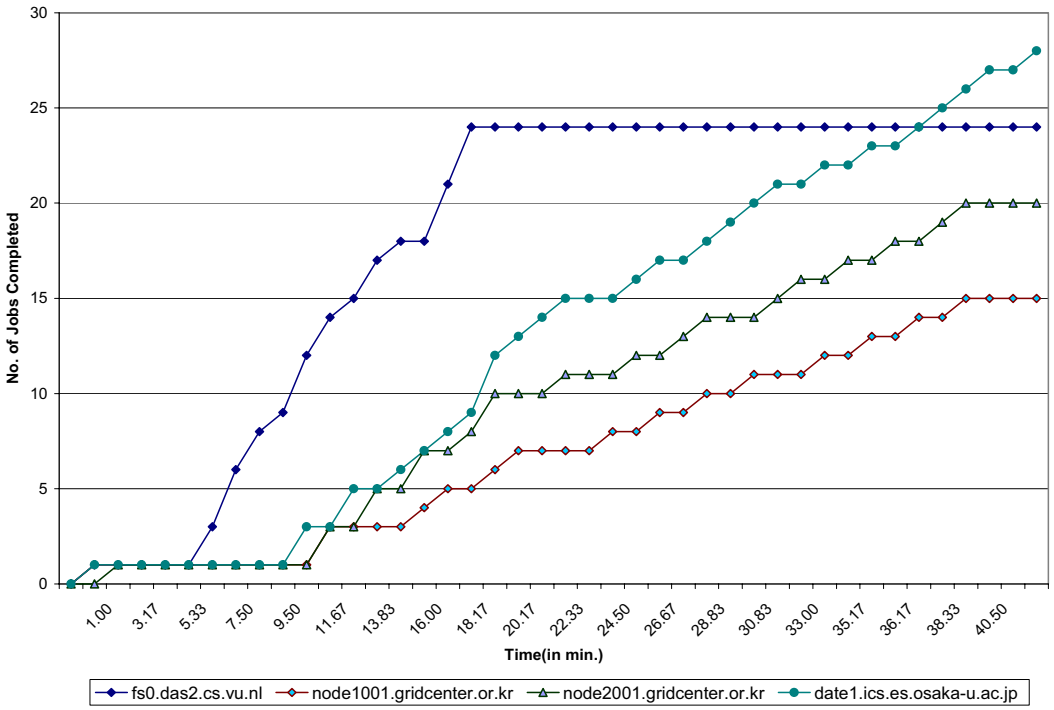


Figure 9. Scheduling with cost–time optimization: cumulative graph of the number of jobs completed versus time.

for example, the analysis might be conducted as a part of a study or for a report, then it is possible to reduce expenditure by using cost-optimization and a relaxed deadline. However, if the results are required in the shortest possible time, then the analysis can be performed using time optimization algorithm.

The visualization of wavelet analysis results of selected sensors is shown in Figure 10 and cross co-relation (pair-wise) analysis of two different sensors data is shown in Figure 11. The viewer also has the ability to invoke wavelet analysis module on the local machine if the results corresponding to the selected sensor are not available. Wavelet analysis decomposes the frequency components of original data over the time. This is the reason why this analysis is promising in comparison with other frequency analysis such as Fourier. Most of traditional analysis methods for the analysis of brain function loses the time information of the original data. Therefore, medical doctors had much difficulty in investigating the change in frequency of MEG data over the time. The visualization was performed so that maximum value of the results is visualized as red, while minimum value of the results is visualized as blue. It is expected that medical doctors can get the intuitive understanding of results at a glance.

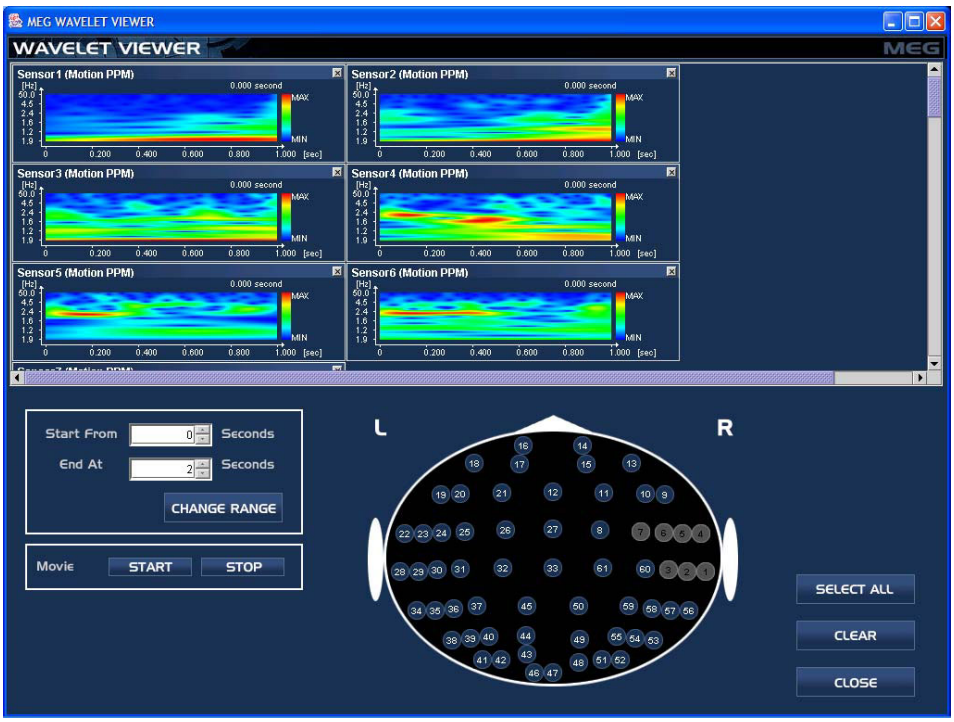


Figure 10. Visualization of the results of wavelet analysis for selected sensors. The numbers within oval shape diagram indicate the sensor number.

7. CONCLUSION AND FUTURE WORK

As the new generation of medical instruments turns out to be more precise and accurate, the medical field will sooner or later have to find the means to deal with large volumes of data generated by them. Grid computing can satisfy their needs for a large amount of processing power, but would have to deal with issues of tight deadlines, small turnaround time, consistency in performance, and reliability of computation before any large-scale adoption. The economy based approach of processing brain activity data as illustrated in this paper would help in enforcing QoS requirements of medical applications and hence would enable adoption of Grid technologies by the bio-instrumentation field.

It has been observed that Grid enabling the application using the message-passing model (MPI) took significantly more time to develop when compared with the approach described in this paper, as explicit parallelization of the application using MPI as a master–*worker* application requires more effort. Hence, it can be safely said that the application developed and deployment time was reduced considerably by adopting a parametric model for the distributed execution of the brain-activity analysis application. Additionally, the approach presented in this paper offered QoS-based deployment of analysis jobs on global Grids.

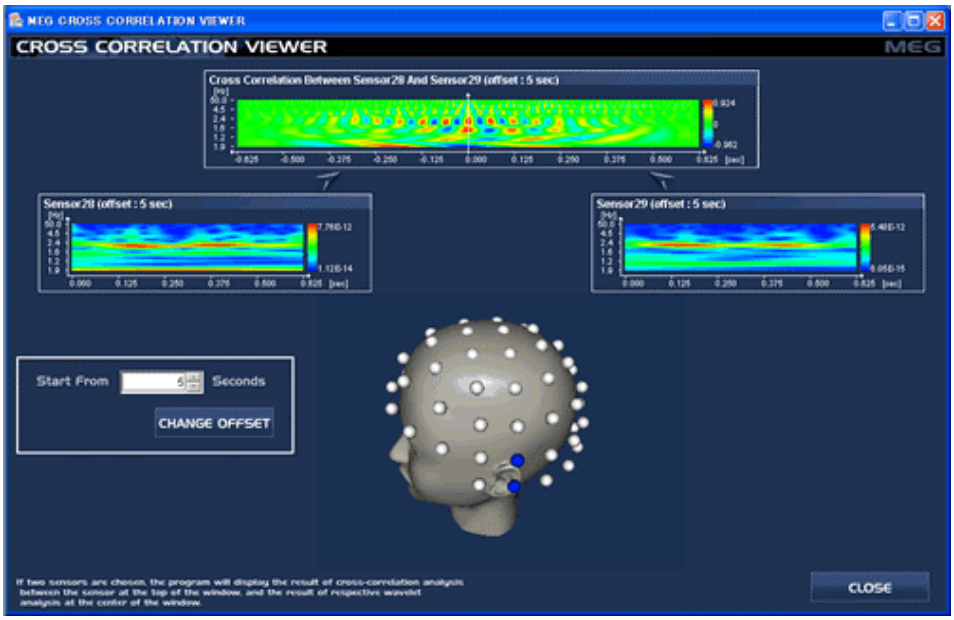


Figure 11. Visualization of the results of two different sensors data cross-correlation analysis.

Although Grids provide basic security services for the protection of the Infrastructure, i.e. preventing unauthorized access to nodes in the Grid network, but they are very limited when compared with the standards required in the medical field. Medical applications demand advanced security services that support policy-based access control, data integrity, and the protection of patients' privacy. These are subjects of future research in Grid computing.

In addition, a number of advances in application scheduling on global Grids are required to support a large number of applications. We are currently working on scheduling with grouping of jobs at the broker level to obviate the need of developing a mechanism for creating meta-jobs (coarse-grained jobs) at user-level. This also enhances the utilization of large data files that are common to many jobs and reduces the communication cost.

ACKNOWLEDGEMENTS

We would like to thank participants of the *Supercomputing 2002* Global Grid Testbed Collaboration for providing us access to the resources that have made large-scale experiments possible. We are grateful to all the system administrators of the machines that were part of the collaboration who have responded to our requests and have made it possible for our applications to execute on their systems. We thank Ed Seidel of AEI-Potsdam, Germany for inviting us to join this testbed collaboration. We would also like to thank Slavisa Garic of Distributed Systems Technology Centre, Monash University, Australia for having made changes to Nimrod-G so that we could use as

many resources as possible of those that were available at our disposal. This study was partly supported by the IT-program of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan; and we would like to thank Life-Electronics laboratory, the National Institute of Advanced Industrial Science and Technology (AIST) for the use of the MEG. The equipment donation and research grant from Sun Microsystems, U.S.A. is gratefully acknowledged.

We thank anonymous reviewers for their constructive comments that greatly helped in improving the quality of paper.

REFERENCES

1. Foster I, Kesselman C. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann: San Francisco, CA, 1999.
2. Mizuno-Matsumoto Y, Date S, Kadobayashi Y, Shimojo S, Tatsumi S, Shinosaki K, Takeda M, Inouye T, Tamura S, Miyahara H. Evaluation system for brain function using MEG, Internet, and high performance computing. *Proceedings of the 12th International Conference on Biomagnetism (Biomag2000)*. Helsinki University of Technology: Espoo, Finland, 2000.
3. Itozaki H. SQUID application research in Japan. *Journal of Superconductor Science and Technology* 2003; **16**:1340–1343.
4. Oscaka University MEG Grid Project Web site. http://www.ais.cmc.osaka-u.ac.jp/Project/megrid/base.htm [January 2003].
5. Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. *MPI: The Complete Reference*. MIT Press: Cambridge, MA, 1996.
6. Sunderam VS. PVM: A framework for parallel distributed computing. *Concurrency: Practice and Experience* 1990; **2**:315–339.
7. Abramson D, Giddy J, Kotler L. High performance parametric modeling with Nimrod/G: Killer application for the global Grid? *Proceedings of the 14th International Parallel and Distributed Processing Symposium IPDPS 2000*, Cancun, Mexico. IEEE Computer Society Press: Los Alamitos, CA, 2000.
8. Kaishima T, Mizuno-Matsumoto Y, Date S, Shimojo S. The experience and practice of developing a brain functional analysis system using ICA on a Grid environment. *Proceedings of the 7th Asian Computing Science Conference (ASIAN'2002)* (*Lecture Notes in Computer Science*, vol. 2550). Springer: Berlin, 2002.
9. Foster I, Karonis N. A Grid-enabled MPI: Message passing in heterogeneous distributed computing systems. *Proceedings of the IEEE/ACM SuperComputing Conference (SC'98)*, Orlando, FL. IEEE Computer Society Press: Los Alamitos, CA, 1998.
10. Smallen S, Casanova H, Berman F. Applying scheduling and tuning to on-line parallel tomography. *Proceedings of the IEEE/ACM SuperComputing Conference (SC 2001)*, Denver, CO. IEEE Computer Society Press: Los Alamitos, CA, 2001.
11. Entropia. FightAIDS@Home Project: A joint effort of Entropia and Scripps Research Institute. http://www.fightAIDSatHome.org [January 2003].
12. Allen G, Benger W, Goodale T, Hege H-C, Lanfermann G, Merzky A, Radke T, Seidel E, Shalf J. The Cactus code: A problem solving environment for the Grid. *Proceedings of the 9th International Symposium on High Performance Distributed Computing (HPDC-9)*, Pittsburgh, PA. IEEE Computer Society Press: Los Alamitos, CA, 2000.
13. The Globus Project. http://www.globus.org [January 2003].
14. Yu J, Buyya R. Grid Market Directory: A Web and Web services based Grid service publication directory. *Technical Report GRIDS-TR-2003-0*, Grid Computing and Distributed Systems (GRIDS) Laboratory, The University of Melbourne, Australia, January 2003.
15. Buyya R. Economic-based distributed resource management and scheduling for Grid computing. *PhD Thesis*, Monash University, Australia, April 2002.
16. Buyya R, Giddy J, Abramson D. An evaluation of economy-based resource trading and scheduling on computational power Grids for parameter sweep applications. *Proceedings of the 2nd Workshop on Active Middleware Services (AMS 2000)*, Pittsburgh, PA. Kluwer Academic Press: Boston, MA, 2000.
17. SC2002 Global Grid Testbed Collaboration. http://scb.ics.muni.cz/static/SC2002/ [November 2002].
18. Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 2001; **15**:200–222.

