

Preliminary Study on Remote Grid Computing for Real-time fMRI Processing

Christopher P. Cheng
B2 U7 Clairmont Townhomes, 309 Acacia Lane
Mandaluyong City, NCR, Philippines
+639165722238
soulfire@edsamail.com.ph

Tristan Joseph L. Raymundo
39D E. Abada St., Loyola Heights
Quezon City, Philippines
+639174171828
tj_raymundo@yahoo.com

ABSTRACT

It is preliminary research into developing a remote grid computing system for functional magnetic resonance imaging (fMRI) processing. This system would exploit the advantages of parallel computing along with an efficient general linear model to overcome some of the constraints concerning the analysis of whole-brain fMRI data in real time. The motivation behind this would be to allow a more cost effective way to process fMRI data efficiently in real time. A significant increase in computational speed can be achieved along with satisfying the computational requirements of the system. Preliminary research into other possible alternatives for the system is then conducted. The supporting software, referred to as Bax, was then analyzed and the possible areas of development have been studied. It has been determined that the cost of communications far outweigh the benefits of the speed up in computation. It has been concluded that this setup can be beneficial if the computing facility is located netographically close to the MRI facilities and the cost of communications is cheap.

General Terms

Experimentation

Keywords

Remote grid computing; real-time fMRI

1.INTRODUCTION

Magnetic Resonance Imaging is a technology which allows for production of high quality three dimensional images of the organs and bones within a human body. It has many uses in the fields of medical and scientific research. MRI is used to examine all organs of the body. This function is especially valuable for detailed imaging of the brain and the spinal cord. MRI examinations are very critical in detection, diagnosis, treatment planning, and follow-up of many diseases. For instance, the images can reveal the precise size and location of a tumor, contributing to more precise surgery and radiation therapy. MRI has become a routine method for diagnosis and treatment. It is still in rapid development. It is superior to other imaging techniques and has replaced several modes of invasive examination, thereby reducing the discomfort and risk of complications for many patients. In this case, MRI is used for detecting brain activity in response to controlled stimuli. The information derived from controlled test such as these would help develop further understanding of the brain processes and aid in mapping its functions.

MRI suffers from a few drawbacks. The first is that the scanners do not take the scans very quickly, which creates problems when the target moves even just a few millimeters. This means that the next image produced would not be correctly aligned. This is why the images must be realigned. Realigning the images is a very computationally intensive process, often requiring a dedicated super computer to process the scans and compute the realignment for each scan. Therein lies the second drawback. Each MRI facility must maintain its own computing facility which is often very expensive. One solution to this is to use grid computing to provide a cheaper alternative without sacrificing computing power.

We investigated the idea of a centralized computer grid setup such that any MRI facility can avail of computing power without having to house their own computing facility. If such a system were to be used, better or more cost-effective setups can be used as alternatives and possibly bring down the cost in time and money of MRI processing.

2.SCOPE

This preliminary study investigates the feasibility of the proposed remote grid system. It implements this by running the supporting software on different cluster configurations that are available to the study such as the Agila High Processing system, a 3 PC mini-cluster running on an 10/100 MBPS Ethernet switch, and on a 8x2 processor Myrinet cluster.

A cluster in Japan, which is being used to process the MRI data, is meant to be replicated locally. This should be set up such that it can be accessed and controlled remotely using Globus software, a remote management toolkit. Then performance comparisons are to be made against the native Linux system and other operating systems (specifically Windows) to test the feasibility of allowing real-time processing of the data.

Limitations to this study include the time period for conducting the study. It is understood that the main cluster which will be used for the actual implementation of the system will arrive near the end of this study's time frame such that only limited tests can be run to verify results. Due to resource limitations, it is not possible to exactly replicate a large cluster and perform full remote cluster interaction. It is attempted to therefore approximate the experiments and measurements instead where needed.

3. RELATED WORKS

Cluster computing / Grid computing is a popular emergent field. This study is the only recent attempt to integrate MRI scan data processing with clusters remotely across a grid.

However, there is material related to real-time MRI processing that allows for results to be displayed almost instantaneously. This involves using a CRAY T3E super computer to do the computations. This is done on site and not remotely. The website is located at:

<http://www.psc.edu/science/Goddard/goddard.html>

4. PROJECT DESCRIPTION

This thesis is an initial attempt to integrate grid computing with the processing of MRI scan data. The common approach to collecting and processing such data is by processing all the subjects before processing all the data in bulk at a later time period. This is done due to the heavy computational loads for each scan, of which the processing can range from a few minutes to many hours. All the data is then grouped into a batch and fed into the system while the testers wait for the results. While this may seem efficient due to the fact that all the subjects are already finished scanning by the time data processing starts, certain problems easily arise from the situation. Data which is determined to have poor quality cannot be resampled in time for processing and must subsequently be rejected. This can amount to huge losses in time and resources because the testers are waiting for naught. It is desirable to have the ability to process MRI data immediately after a subject undergoes scanning such that poorly sampled data can be resampled immediately. However, MRI facilities are in high demand and time slots for usage is extremely limited. There exists a huge amount of data that needs to be processed very quickly. Processing and checking is much too slow for it to be allowed immediately after scanning. It would then be ideal to have the ability to process the MRI data as the scan is being conducted such that data of poor quality can immediately be retested and processed again. This kind of immediacy requires great processing power in order to be implemented, power in the scale which is unreasonably forbidding in terms of cost.

However, grid computing presents the possibility of rapid real time processing of data without the prohibitive cost required to maintain a high-end computing facility per scanning device. All the computations are currently done on a dedicated cluster for such purposes. It was proposed that it might be able to achieve real time processing on a remote cluster, which is not immediately within proximity of the MRI scanner. This way, a researcher only needs to remotely connect to the cluster from a local machine, transfer the input data as it is received from the scanner, and return in a processed form, ready for analysis. Eventually, should this be a feasible idea, multiple MRI scanners can linkup with a single cluster. This could amount to time efficiency, yet avoid the need to have a cluster dedicated for this purpose for each MRI machine.

This thesis is experimental in nature. The cluster being used in Japan, as well as the cluster that was shipped here, though significantly faster, is also quite expensive, owing to superior Myrinet infrastructure. It was investigated if performance comparable to the Myrinet cluster can be achieved with cheaper Ethernet clusters. Through experiments and measurements, we

tried to determine if it was actually feasible to process MRI data on a remote cluster given the currently utilized Globus system of remote execution management. We also investigated the economic aspect of the proposed system, specifically cost-effectiveness of the entire setup.

5. ARCHITECTURAL DESIGN AND DESCRIPTION OF COMPONENTS

BAX – Stands for Brain Activation eXplorer. It takes as input a data set of a 30 slices (each I.xxx file being one slice) worth of 64 x 64 pixel images. The resulting 122880 voxels are equally distributed to all nodes. The volume is then realigned relative to the first image volume then smoothed. This is necessary since the subject's head may move during the scanning process, therefore causing a misalignment. During this preprocessing, computing nodes may need to communicate with each other since they may need values from other nodes since realignment and smoothing are operations relative to the entire brain. After preprocessing, some quantities are computed and incorporated to the previous values. The updated results are then sent back to the master node. The master node stores the updated results and reads the next volume in the time series (next 30 slices) and the process is repeated. Essentially, it utilizes the Message Passing Interface standard (MPI) to communicate between nodes in a cluster.

The BAX takes in the input one volume at a time. Each volume is composed of 30 slices worth 16096 bytes each. Therefore, an entire volume amounts to 482880 bytes. The output is in terms of volume as well. The more significant parts of the output are the image of the value plus adjusted calculations for it, worth 983040 bytes each, for a total of 1966080 bytes of data output per volume.

The BAX program is divided into several stages of work. The more important parts, which were subsequently measured in our experiments were as follows; Statistics Computation (CS) wherein the statistics and results of the data is computed. Volume Realignment (RA), wherein, each volume of 30 slices is split up among the nodes and realigned with respect to a fixed volume, with interprocess communication for interdependent data. Smoothing (SM), where smoothing and convolving of the aligned volume is performed, there is also interprocess communication done. Cluster Communication (CC) pertains to the time spent in passing data to each node/process excluding the ones within Realignment and Smoothing. This is usually the data which is resent after an operation such as realignment or smoothing is done.

The Globus Toolkit is a suite of applications which facilitate resource and data management over a grid. For the purposes of this system, the resource management module GRAM (Globus Resource Allocation Module) is used. Its primary function is to handle the remote execution of the program that prepares the transfer of MRI data, as well as BAX itself.

The Ethernet cluster on which the benchmarks were run on were composed of three Pentium 2.42 GHz with 512Mb of RAM each, connected to a 10/100mbps switch. They were setup to dual boot between Linux Red Hat 9 and Windows XP Professional. Under Linux, it used the LAM MPI implementation to handle the Message Passing Interface system. Under Windows, the MPICH implementation was used.

The Myrinet cluster is high speed network infrastructure. The Myrinet cluster is equipped with 16 nodes, each running on a 2.8

GHz Intel Xeon dual processor. Each node also has 1 GBytes of RAM. It is 1000 times faster than regular Ethernet networks. The core of this speed lies in superior inter-node connections. Data transfer is immensely faster. It is currently composed of 8 computers running on Red Hat 7 and using the SCore high performance parallel computing environment.

6.METHODOLOGY

- Learn the workings of MPI.
- Understand the workings of the BAX program.
- Analyze algorithm used
- Assemble a local cluster running MPI
- Analyze time comparison results given different cluster platform configurations.
- Analyze performance comparisons between other parallelized code
- Look into optimization of the code.
- Set up a Globus system in order to allow for remote cluster coordination
- Analyze feasibility of remote execution of processing program.
- Investigate possible areas of development

7.IMPLEMENTATION DETAILS

Upon receipt of the BAX program and the corresponding MRI data to be processed for testing, we started to work on getting it to execute on the Agila High Performance Cluster. It allowed us to learn more about the workings of MPI and how to use the program correctly.

However, due to limitations of the Agila cluster, we required our own cluster where we can perform more thorough and reliable tests. Hence we were eventually able to secure three computers and setup a Beowulf cluster. Taking after the MPI system of the Agila cluster, we decided to use LAM, which is an MPI implementation. The BAX program needed a few modifications in order to port it the MPI system. Performance analysis of the system was then conducted. This focused on studying the benefits of parallelization of the processing system.

The BAX program was examined in order to investigate possible areas for optimization. Some areas of interest included the amount of granularity of the work done and the amount of parallelization used in processing the work. The most important area uncovered dealt with data size and transfer of said data.

Studying the possibility of executing the MRI processing program under Windows, a Windows implementation of the MPI standard had to be setup. Performance comparisons were made between the Linux Beowulf cluster and the cluster under Windows. The BAX program had to undergo further modifications in order for it to port correctly under the new platform.

It had been intended to remotely execute the BAX program. The current infrastructure in place on the original system in Japan to perform remote job execution was through the Globus toolkit. A server was to be setup here to be remotely called by a client. However, at that point in time, the Myrinet cluster had not yet arrived. Hence, it was decided to setup both client and server locally. Problems arose due to lack of resources, specifically a whole new set of computers free to be used. In order to continue work, both server and client were setup on the same cluster. It was then planned to perform approximations of performance. In order to come up with such an approximation, it was necessary to approximate the time required to send data between a client and a server. As a test case it was done between Japan and the Philippines, using the benchmarks as basis for the delay incurred by the distance between the two countries. We also attempted to transfer data between our local cluster and the Agila cluster for reference on local transfers.

Due to problems beyond our control, the Myrinet cluster arrived much later than expected and since we did not have complete access to the Myrinet cluster, we could not setup a remote execution environment under Globus as we would have desired. Although we were able to perform some benchmark tests.

Research on the costs of maintaining both types of clusters plus internet connections was conducted.

Possible areas of development were considered, such as either removing parallelization altogether, porting the MRI processing code to other parallel computing platforms, in hopes of achieving better and more cost-effective parallelization, and some analysis of network communications.

8.RESULTS

We have been able to successfully run the BAX program on the Agila High Performance Cluster. According to our performance analysis, there is significant speedup with using only 3 nodes. Beyond that, it degenerates.

Agila High Performance Cluster

BAX

Procs:	CS:	RA:	SM:	CC:	Tot:
2 :	0.1181	4.0993	0.4916	0.3509	754.645
3:	0.0583	2.7694	0.6289	0.7980	591.250
4:	0.0380	2.5655	0.7692	1.0031	606.427
5:	0.0287	2.4653	1.0203	1.0520	651.203
6:	0.0362	2.6803	1.0083	1.1796	662.348

Local Linux Beowulf Cluster

BAX

Procs:	CS:	RA:	SM:	CC:	Tot:
2 :	0.0239	1.6153	0.4916	0.2421	311.511
3:	0.0120	1.3812	0.3713	0.4211	299.380
4:	0.0080	1.9322	0.6709	0.5271	333.091

5:	0.0063	2.1135	0.7272	1.7123	517.395
6:	0.0059	2.1233	0.7964	1.6532	476.683
7:	0.0041	2.0921	0.9463	1.7857	529.413

According to our results, there is a relative speedup only when the number of processes are equal the number of nodes. In effect, this is achieved only when it uses all three available nodes with one process each. Using anything above three processes makes it steadily degenerate into slower and slower processing times. It accomplishes relative speedup when using a number of processes equal to a multiple of the available nodes but compared to using a number of processes which exact the number of nodes, the later is still faster. This is roughly the same result as in the Agila cluster.

Apparently, above the three process mark, all the tasks, except Statistics Computation slows down for BAX.

It had been determined that there is very little CPU cycles used by each process, therefore, it has been suggested that the program be modified such that it uses coarser granularity. This means that a larger amount of data is to be sent to each node for processing in order to better utilize the cluster parallelization. However, in the current algorithm, each node is sent an equal amount of voxels taken from a whole of 64x64x30 volume. Each node then realigns its data all at once with some inter-process communication for cross-referencing of data. Hence, it appears that there is no immediately feasible means of improving the efficiency by increasing the amount of voxels per node due to logical limitations and possible dependencies between data. An added limitation is introduced by real time processing since the data to be processed relies on data which might not have arrived yet. It has been considered that it might be possible to restructure the processing such that each node is given an entire volume, to allow for coarser granularity and eliminate data dependency issues since each is a self-contained whole. However, this is not possible in a real time setting precisely because each volume is received one at a time.

Much of the degeneration of the speed of the processing was found to owe mainly not so much to the CPU load but to the data transfer delay. The amount of data being transferred along the wires is what was actually delaying the process. There appears to be an optimum amount of processes spawnable for parallelization. Using less than optimum implies that the capabilities of the cluster are not fully utilized and using too many would then mean there would be too much data flowing through the wires, and a very minute amount of computational data to be split up in proportion to the data communication delay it generates.

After multiple tests, it has been determined that the same resulting effect applies for the Windows cluster. There was some speedup on the use of three processes, one on each node, with the best relative performance occurring when the number of processes are equal to a multiple of the available nodes. The difference is that the Windows system was generally slower. It was slightly slower on the 3 process benchmark. This remained the optimum number of processes to be run. On all other benchmarks using a different number of processes, it was greatly slower.

Local Windows Cluster

BAX

Procs:	CS:	RA:	SM:	CC:	Tot:
2 :	0.0376	1.8933	0.2708	0.1675	363.109
3:	0.0118	1.4612	0.3635	0.3551	310.297
4:	0.0123	2.2469	0.9221	0.81607	589.333
5:	0.0106	3.3807	1.2698	1.2953	889.594
6:	0.0079	1.8641	0.6120	0.8446	438.125
7:	0.0073	3.4853	1.3964	1.4070	838.016

Test results on the local Globus setup yielded predictable results. It was only slightly slower, owing to overhead, but the pattern followed that of the results of the Linux Ethernet Beowulf cluster benchmarks. However, due to the time required to shuttle data across, it added on average 3~4 seconds to the total time to complete the entire job.

On a Myrinet cluster, there was a progressive speedup as more and more processes were used until they matched the number of available nodes. It refused to spawn more processes than there were available nodes. The fastest time resulted from using all the available nodes.

Myrinet Cluster

Procs:	CS:	RA:	SM:	CC:	Tot:
2:	0.2073	1.2230	0.0787	0.0128	224.578
3:	0.0111	0.5518	0.0529	0.0446	130.785
4:	0.0147	0.6372	0.0631	0.0341	143.863
5:	0.0073	0.3614	0.05187	0.0452	104.792
6:	0.0064	0.4109	0.0762	0.0609	116.541
8:	0.0047	0.3161	0.0674	0.0549	101.313
10:	0.0036	0.2859	0.0594	0.0642	101.067
14:	0.0022	0.2357	0.0518	0.0553	92.345
16:	0.0020	0.2022	0.0441	0.0490	88.203

After comparing the benchmarks with another sample program, a PI calculation given 2140000000 iterations, we yielded the a different set of results. One issue is that since the programs ran too fast no matter the adjustment, it might be susceptible to delays on the network. However, the pattern of the results indicated that the more nodes used, the faster it went.

PI for Agila

Procs:	Time
1:	48.257
2:	32.214
3:	24.487
4:	24.574
5:	19.665
6:	16.333

PI for Local Cluster

Procs:	Time
2:	16.321
3:	11.021
4:	8.625
5:	7.157
6:	6.497
7:	5.554

PI for Myrinet Cluster

Procs:	Time
2:	17.215
3:	11.541
4:	9.337
5:	7.043
6:	6.017
7:	5.343
8:	5.001
9:	4.761
10:	3.889
11:	3.623
12:	3.223
13:	3.110
14:	2.991
15:	3.005
16:	2.745

According to our measurements, the average speed of data transfer between two nearby systems in the Ateneo, network-wise, namely the Agila cluster and our local cluster, was 906.14 KB/s. This is achievable due to the fact that these two systems share one network. On the other hand, transferring data from Japan to the local Myrinet cluster took 92.7 KB/s using the school's E1 connection and an E1 connection in the other country. There were roughly 26 hops between said networks. Given the data size and bandwidth speed, it can be calculated that it should take 5 seconds to send the data (482880B / 92.7KB) while it takes 20 seconds to return (1966080B.92.7KB).

After some research, it has been found that the T1 connection being used by the Ateneo for its internet connection costs around \$3,500 per month. Maintenance of the Myrinet cluster itself would cost the school somewhere around Php35,000 per month. In the test case country, Japan, an E1 connection costs only \$150.

If, for example a client from Japan were to use a cluster here remotely through Globus under optimum settings each:

Infrastructure: Send Process Return Total

Ethernet :	5s	6s	20s	31s
Myrinet:	5s	1s	20s	26s

It would take the Myrinet cluster roughly 5s less time to get the entire process done with. However, this only applies for one-to-one relationships between client and server.

9.CONCLUSION AND FUTURE WORK

This paper presents the results of experimentation and analysis of the feasibility with regards to real time processing fMRI using remote grid computing. The process is desired in order to facilitate research into brain processes.

Parallelization yields speedup for both Ethernet and Myrinet clusters. However, the rate of speedup for Ethernet clusters is significantly less compared to Myrinet clusters. The entire process itself consistently executes faster on the Myrinet than on the Ethernet even when forced to use only two nodes each. Using the test comparison program PI, it is proven that parallelization increases speed significantly either way.

In the case of BAX however, there seems to be problems. More importantly, there appears to be a limit on the parallelization of the code. Beyond a certain extent, it becomes slower regardless of the availability of nodes. The magic number appears to be three in this case. Upon closer inspection of the program, the only task that speeds up is Statistics Computation. Within, there is only simple calculations to deal with. In the case of Realignment and Smoothing, within the functions are other MPI Broadcast and Barrier calls. This means that data is being exchanged with other nodes while the process is being done. This inter-process communication is apparently a necessity due to data dependencies within. Cluster Communication increases for both clusters due to the need to transfer to data to more recipients. However, this is not as significant in the Myrinet cluster, owing again to faster infrastructure.

The BAX program is what is commonly known as a *fine grained* program. The job is relatively easily processed to the point that what contributes largely to slowdown is the transfer of data between processes or machines. Ideally, should the number of processes slightly exceed the number of nodes, there should still be some speedup equal to or slightly smaller than that of having exactly the same number of processes as nodes. In this case however, there is immediate slowdown on the Ethernet cluster. There is no benefit to using more processes than there are nodes. The best way to compensate for this is on an Ethernet machine is to increase the granularity of the program by making it such that it gives our more data to process. This way, more work is done on the CPU level, ensuring that maximum benefit is derived out of the propagation delay across the wires. The Myrinet cluster does not suffer from this problem as noticeably since the Myrinet infrastructure is characteristically fast when it comes to propagation delay. This is why there is actually more noticeable speedup when more and more nodes are used, since each additional node contributes to faster processing, with minimal propagation delay, until the maximum number of nodes is reached.

Upon comparison between the Linux system and the Windows system, the Windows performs generally slower. However, if the correct number of nodes is used, i.e. 3 in this case, the difference turns out significantly smaller. The only problem is that Globus, which is the remote execution toolkit of choice, does not work under Windows. Setting up the cluster is much easier to accomplish under this platform, however. If an efficient remote execution mechanism can be found as a replacement, then setting up a grid infrastructure under Windows can be a plausible option. The flexibility afforded by the capability to use either OS platform for cluster computing can potentially be of great value.

The need for optimization lies on systems with high propagation delays such as Ethernet clusters. On high-end Myrinet systems, this is not as much an issue. The problem of optimization lies primarily on the fact that each volume is generated and received from the MRI scanner one by one, and not by batch. Therefore, there can be no attempt to coarsen the granularity of the program as a means of optimization. If an optimization can be found, then, it might therefore be possible to use a relatively cheaper Ethernet cluster, than a high end Myrinet system. This way, not only does it lower the cost required to have a grid infrastructure, but it also opens the possibility of using more readily available infrastructures to process information, resulting in theoretically closer to real time processing.

Though a Myrinet cluster represents a more expensive investment compared to an Ethernet cluster, the power and efficiency makes up for the cost. Eventually, it is planned to allow multiple MRI scanners to simultaneously link up with a single cluster. On an Ethernet cluster, the load of multiple jobs communication across each node will cause greater delays. On a Myrinet cluster, this problem will be more manageable, hence, it is recommended to use this system in such case.

With regards to actual remote grid computing, one concern that became apparent was bandwidth. Given the data transfer measurements, it can easily be seen that a relatively large amount of data needs to be sent across. For a E1 connection, it may seem manageable. However, as has been established, such a connection is expensive. A possible way to reduce bandwidth consumption is to reduce the size of the input/output files. Compressing the files is not an option for it turns out that they even grow bigger. The output files are as large as they are because of the data type used to store the information as it is written to a file. It was intentionally made large in order to preserve the precision while the calculations were in progress. It turns out that the precision does not matter as much when it is finally about to be written. Therefore, it is possible to normalize the values and store them in as small data type as necessary, and in this case, short int. Implementing this recommendation is open to further study.

Reducing the size of the output files can do a lot towards solving the bandwidth bottleneck. When multiple MRI scanners link up with a single cluster, the connection will still slow down significantly, unless an expensive line is leased. Hence, to be able to achieve faster and more cost-effective performance, it is still not enough. Although there are less bandwidth problems as compared to before, long connections, especially out-of-the-country ones are much more expensive than within the country connections and geographically close connections.

Efficient and cost-effective remote grid computing is still possible in real time only if the cluster and the MRI scanners

using it are placed in geographically or netographically close areas. It is important because close connections are faster and cheaper, as compared to farther connections, especially when it becomes international in level. If not netographically close areas, then at least set up the grid system in areas where connections are cheaper. It's not practical to have to spend on an expensive broadband line simply to achieve remote MRI processing. Multiple MRI scanners linked up to a single cluster can be serviced more cost-effectively and efficiently. The grid computing system cannot be placed simply anywhere and expect it to be efficient and cost-effective. It must be strategically placed such that the scanners and the computing facility are located netographically close.

Given that there is not much significant difference between Windows and Linux clusters, given proper conditions, it opens up the possibility of using Windows based grid-architectures to implement the MRI processing function of BAX. Further exhaustive studies have yet to be made on the feasibility or comparative efficiency between the two.

However, it must be stated that no matter what kind of architecture may be designed, it will always have to take into consideration the bandwidth problem. The amount of data to be exchanged does not change. Even if it is assumed that it is acceptable to reduce the size of the output files by changing the storage method, it will still reach a bottleneck when it comes to data communications. So, the recommendation of strategic placing must still be considered in any case.

Though it is planned to allow multiple MRI scanners to link up with a single machine so as to save costs, the full extent of the effects of such a set up is still a question open for further study.

In conclusion, It is possible to achieve strict real time processing only with a system set up such that they are netographically close to each other, using a Myrinet cluster due to bandwidth problems. The performance of an Ethernet cluster, though slower, is acceptable for near-real time purposes. The lesser cost may even make it a possible alternative for researchers who may not be able to afford a Myrinet level cluster. Across national boundaries is still an issue due to speed penalties and escalating costs when communicating on a transnational level.

10. REFERENCES

- [1] Bagarinao, Matsuo, Nakai, Real-time Functional MRI Using a PC Cluster. *Concepts in Magnetic Resonance Part B, Vol. 19B* (2003), 14-25.