Hardware Computing for Brain Network Analysis

Yu WANG*, Yong HE†, Yi SHAN*, Tianji WU*, Di WU* and Huazhong YANG*
*Department of Electronic Engineering
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University
Email: yu-wang@tsinghua.edu.cn, {shany08,wutj06,wud07}@mails.tsinghua.edu.cn
† State Key Laboratory of Cognitive Neuroscience and Learning
Beijing Normal University
Email: yong.he@bnu.edu.cn

***Abstract*—[1] As the scale of computer clusters and supercomputers is getting larger, the problem of power consumption and heat dissipation has become the biggest obstacle for the ever growing need for computation. Designing platforms for specific applications using the reconfigurable logic such as Field Programmable Gate Arrays (FPGAs) or highly parallel processors such as Graphic Processing Units (GPUs) will dramatically increase power efficiency. This is the concept of domain specific computing. Combining the advantages of different platforms to build a heterogeneous computing platform is the trend of domain specific computing.**

On the other hand, the research on brain networks plays a vital role in understanding the connectivity patterns of the human brain and disease-related alterations. Recent studies have suggested a noninvasive way of modeling and analyzing the human cortical networks with MRI by graph theory based approaches. However, both the construction and analysis of brain networks require tremendous computation. Currently, only hundreds of nodes can be analyzed due to lack of computing power. By increasing the number of nodes, the resolution of cortical networks will be greatly enhanced, thus hopefully helps the early diagnosis of brain diseases such as Alzheimer's disease. A well-designed computing platform is the key to this problem. In this work, we inject the power of heterogeneous hardware computing into the brain network research, to help the research on the connectivity patterns of both normal and diseased brains. Besides, one important outcome is an accelerated BLAS and Graph algorithms package, which will provide insights into domain specific computing to boarder audience in both biomedical and computer science domains.

I. Introduction

In parallel with the quick development of science and technology, people's demand for higher computation power is insatiably expanding. In the field of computational biology, researchers from IBM recently announced their work about brain activity simulation of 1 billion spiking neurons and 10 trillion synapses [1]. Another example is the brain network research presented in this work, which also demands tremendous computation power.

To satisfy the demand, scientists and engineers build supercomputers with hundreds of thousands of cores. However, when the scale of supercomputers is enlarged, the problem of power consumption and heat pollution is sharpening. For instance, "Dawn" the BlueGene/P supercomputer utilized for IBM's neuron simulation consumes 1.4MW to mimic a neural network at roughly the same size as a cats brain. However its speed is 100 times slower than the real brain [2], [1].

To satisfy such computation-hungry applications effectively, people build specific computing platforms for specific application domains. These platforms may have several kinds of computation cores, including general purpose CPU, general purpose GPU, and FPGAs. The table I shows the comparison of CPUs, GPUs and FPGAs. Each technology has its advantages and drawbacks. By combining the advantages of each technology while avoiding drawbacks, we can build heterogeneous hardware computing (HHC) platforms.

TABLE I
Comparison of CPU, GPU and FPGA

	Peak 32/64bit GFlops	Power(W)	Design Effort
CPU(Core i7)	70/70	130	Easy
GPU(RV870)	560/2800	150	Middle
FPGA(Vertex-5)	36/140	5	Hard

The HHC has several advantages. Firstly, by utilizing the reconfigurable hardware such as FPGA or highly parallel processors such as GPGPUs, it can greatly improve the performance space ratio and power efficiency. Secondly, instead of spending heavy costs to solve each application separately, it provides a framework or platform that addresses the needs for a specific application domain. Hence, the designing efforts for a particular application in the domain are greatly alleviated, with little performance or efficiency lost.

These approaches are crucial in human brain network research, which also requires huge amount of computation. Recent studies [3], [4], [5] show that people can construct a network model of the brains noninvasively by multi-modal brain imaging techniques (e.g. structural, functional and diffusion MRI). And these network models can be used to identify brain diseases, such as the Alzheimer's disease [6].

Currently, the imaging techniques is advanced enough to give a high resolution voxel based image of the brain. Analyzing the brain network in voxel basis will give us a much more detailed view of the brain, which is important and necessary

[1]This work was supported by NSFC (No.60870001), 863 program of China (No. 2009AA01Z130), TNList Cross-discipline Foundation, and MSRA and AMD University program.

in the brain network research. However, both the construction and analysis of voxel based brain networks require tremendous computation power. Currently, the brain can be represented at a resolution of 20k-100k voxels, each of which can be considered as a single network node, which will result in large scale networks. Analyzing such large scale networks is quite time consuming. Moreover, this kind of analysis need to be done on a large number of subjects in order to conclude statistic results or patterns, which also adds to the time complexity.

In this paper, we, for the first time, propose a heterogenous hardware computing framework for brain network analysis, i.e. to accelerate the analysis of large scale fMRI data. Under this framework, we accelerate the construction and modularity operations of voxel based brain networks. Our brain network has around 38000 nodes.

The rest of this paper is organized as follows. Section II presents some related work in both brain network research and heterogenous hardware computing. Section III introduces our computing platform for the application domain and our implementations as well as our experimental results, including construction and modularity operations. Section IV concludes the paper and puts forward the future work.

II. Related Work

Many work has been done to construct and analyze the brain networks, and most of them focused on a coarse scale (i.e., region level). For instance, several studies have utilized a prior brain atlas to parcellate the brain into tens of brain regions and then constructed region-based brain networks[7], [8]. Other studies have used image voxels to build a partial brain network at a fine scale[9]. Both methods lose some detailed connectivity information.

In [10], brain network analysis is done on voxel basis, which is much finer than previous region based analysis. When partitioning the voxel-based brain network, some approximate algorithms (such as random walk method) were used to avoid complex eigenvectors computation of the correlation matrix. However, tremendous computation is unavoidable when conducting deeper analysis on voxel basis.

On the other hand, heterogenous and reconfigurable hardware computing platform has become a new focus for efficient computing in recent years. Much research work has been done on utilizing the power of heterogenous and reconfigurable platforms in various areas.

In [11] and [12], FPGAs and GPGPUs are used to accelerate machine learning algorithms in order to accelerate web search engines. [13] and [14] presented significant performance improvement for multiple sequence alignment (MSA) by using GPGPU. MSA is a frequently used process in molecular biology.

In [15], the use of heterogenous and reconfigurable computing platform in the application domain of medical imaging is discussed as a typical case. The author addressed several widely used algorithms in X-ray and MRI data processing, showing that FPGA and GPGPU can greatly improve the performance. The report also showed that different algorithms have different 'ideal' chips - some are more suitable for FPGAs while some are better with GPGPUs.

In our project, basic linear algebra and graph algorithms are identified as performance bottlenecks. There is some recent work addressing the issue of BLAS or graph algorithms acceleration on FPGA or GPGPU. For example, Zhuo et al used FPGA to accelerate Level 1, 2 and 3 BLAS [16], or SpMV routine [17]. In [18], the author compared and analyzed the performance and efficiency of CPU and FPGA implementations of BLAS, based on IEEE standard floating point format. In [19], a framework for linear algebra operators implementation on GPU is proposed.

For graph algorithms, in [20], the author focused on the mapping between reconfigurable circuit (such as FPGA) and graph, and solved graph algorithms on that basis. In [21], several fundamental graph algorithms are implemented on GPU, such as breadth first search, single source shortest path, and all-pairs shortest path. [22] presented some ways to implement sparse matrix and graph algorithms on CUDA; [23] presented a method to find shortest paths in graphs on GPGPU.

All these work brings great help to our project. Our research takes some steps further by focusing on the particular application domain of brain network research. We design some typical graph-based algorithms to best fit the domain's features such as the precision requirements and graph density, since these factors will affect the data structure and memory access patterns.

III. Hardware Computing Framework for Brain Network Analysis

The computation strategy for brain network analysis can be divided into two categories. One corresponds to end users in hospitals. PCs equipped with dedicated hardware (such as GPGPUs and FPGAs) are the best choice for their convenient installation. These computers can quickly analyze patients' brain network and give diagnostic suggestions based on an expert system. The other category corresponds to research institutes, where it is possible to set up various high-performance computers. Those computers with dedicated hardware may group into clusters to form a heterogeneous hardware computing platform. Its high efficiency and computation speed are especially crucial for the analysis of the brain networks of large number of subjects, on which the expert system is based.

The original data used the work of Brain Network analysis is acquired from functional magnetic resonance imaging (fMRI), from which a voxel-based brain network can be built. It is a network that illustrates the connections of the brain. Each node in the network represents a voxel, and each connection represents the correlation between the input signals of two voxels. After the brain network is built, graph theory algorithms can be applied to analyze the network, such as network hub detection, modularity detection, small-world analysis, etc.

In relation to the construction and analysis of Brain Networks, we design a heterogeneous computing platform by

utilizing CPU, GPU and FPGA techniques as well as our previous work of FPMR and its extension FPMR^2 and FPMR^3 [24]. The platform supports both single node and multiple nodes systems. With such a heterogeneous platform, we can considerably improve the performance of Brain Network analysis by redistributing the computing tasks to devices with the most suitable computing resources and memory structure.

In the subsections below, a detailed description of our work is presented by dividing the Brain Network analysis problem into three fundamental tasks: network construction, all-pairs shortest path and network modularity. The method of adapting these tasks to heterogeneous platforms is introduced in each subsection correspondingly, as well as some brief experimental results of the specific implementations.

The computing platform in our experiments has a quad-core Phenom II 956 CPU running at 3.4GHz, 8GB DDR3 memory, a Radeon HD 5870 graphic card with RV870 core at 850MHz and 1GB GDDR5 memory. Our GPU kernels are written in ATI Intermediate Language (IL) [25].

A. Construction

By fMRI, a series of signal with length L is acquired for each of the N_v voxels. For each pair of nodes (voxels) (v_i, v_j), we obtain the Pearson's correlation [26] between the series of the pair, i.e.

$$\hat{r}_{i,j} = \frac{\sum (v_i - \bar{v_i})(v_j - \bar{v_j})}{\sqrt{\sum (v_i - \bar{v_i})^2 \sum (v_j - \bar{v_j})^2}} \quad (1)$$

$$= \frac{\sum v_i v_j - \frac{1}{n}(\sum v_i)(\sum v_j)}{\sqrt{\left(\sum v_i^2 - \frac{1}{n}(\sum v_i)^2\right)\left(\sum v_j^2 - \frac{1}{n}(\sum v_j)^2\right)}} \quad (2)$$

where v_i denotes to the series of voxel i, $\bar{v_i}$ is the average of the series of that voxel, and all $\sum$ denotes to $\sum_{t=0}^{L-1}$, i.e. summing along the whole time series.

This operation is very suitable for mass parallel processors such as GPUs, since the computation of different pairs of voxels can be fully parallelized.

Table II shows the comparison of running time for constructing brain networks. In the first test, we generate the full correlation matrix on both CPU and GPU. The result shows that construction of brain networks on GPU runs 26x faster than that on a single core CPU. In the second test, we apply a threshold $r_{th} = 0.75$ on the correlation matrix so that only those correlations exceeding r_{th} are considered connections. Our experiment shows that when applying the threshold for binaryzation, the time of GPU accelerated implementation is 24x faster than the CPU implementation.

TABLE II
BRAIN NETWORKS CONSTRUCTION SPEED COMPARISON (IN SECONDS)

	CPU 1-core	RV870	Speedup
Full matrix	985.5	37.5	26x
Adjacency list	1021.9	42.0	24x

B. Modularity

The algorithm we choose for brain network modularity is the eigenvector-based spectral partition method of Mark Newman [27] for non-directed non-weighted networks. The key of the algorithm is to calculate the eigenvalue of a Modularity Matrix constructed from the adjacent matrix. The partitioning is based on the eigenvector of the most positive eigenvalue.

We use Power method to calculate the eigenvalue and eigenvector. The Power method can be divided to several basic matrix and vector additions and multiplications. So our parallelization is focused on these basic operations. We use the CSR (Compressed Sparse Row) format for the adjacent matrix as it can be easily implemented on the AMD GPU platform. Moreover, in each iterations of the power method, the adjacent matrix, degree vector and would not be changed. As a result, all the input data need only one time transfer.

The modularity computation costs a very long time and divides the network in to thousands of communities. The CPU version requires so much time that we cannot finish the full division of the network. In response to that, we truncate the calculation to the first 100 iterations and compare the CPU results with GPU. Table III shows the speed comparison.

TABLE III
BRAIN NETWORKS MODULARITY SPEED COMPARISON (IN SECONDS)

	CPU 1-core	RV870	Speedup
First 100 iterations	74496.97	928.73	80x
Slowest iteration	2515.55	31.68	79x
Fastest iteration	3.78	0.29	13x

C. All-pairs Shortest Path

Dijkstra, Floyd-Warshell (FW) and Breadth-First Search (BFS), are the most popular shortest path solvers used in application and research. However, due to the limited computing power of CPUs, when the input graph is huge, the running time is intolerable.

After comparing the three algorithms, we choose BFS for FPGA implementation. Although Dijkstra has the lowest time complexity among the three algorithms, its sorting and selecting operations and tight data correlation make parallel processing difficult. BFS has high parallelism and hardware mapping ability compared with Dijkstra. Its performance is better than FW's and lower than/similar to Dijkstra's with sparse graphs.

We design a universal PE to deal with a single source node shortest path problem. A PE contains the following elements: 1) A FIFO is used to play the part of a queue. As an optimization, a register array is included in the FIFO to avoid inserting nodes already in the queue, which can push down the amount of nodes expanded during the processing. 2) A RAM is included to record shortest path (distance) from the source node to each node. After the processing, data in the RAM is directly sent to output. 3) An adder and a comparator perform as the expanding unit. They calculate the new distance

from source node to a certain node, comparing the result with current recorded distance, and updating it if needed. The graph is organized in CSR format, stored in off-chip memory. A carefully designed controller is in charge of managing data streams between memory and all PEs.

We use a blocked Floyd-Warshall algorithm [28] for the calculation of the all-pairs shortest path on GPUs. The core of FW algorithm is the iterative calculation of the minimum between every element of the Cost matrix and the sum of elements on its column and row.

The idea of block FW is to divide the Cost matrix to several sub matrices with the same size, and calculate them separately. In the calculation of each sub matrices only a part of the Cost matrix is needed, in which way the parallelization of the iterations can be achieved. Besides, in the blocked algorithm, only a part of the matrix is needed to store on the local memory on GPU, which enables the possibility of the calculation of very large matrices. As a result, the performance is greatly improved by the GPU realization.

IV. Conclusion

In this work, a heterogenous computing platform is proposed to accelerate the computation of brain networks' construction and analysis. In this application domain, we first implemented two algorithms, e.g. network construction and modularity analysis, and achieved huge improvement in efficiency. It makes large-scale voxel-based brain network analysis applicable on large number of subjects.

Based on our platform, we can further implement other key algorithms for brain network research, and try to build the model of high voxel-basis resolution for brain diseases diagnose. Furthermore, we will build two levels of platforms, one for end users such as hospitals, which is capable for assisting disease diagnose, while the other for research institutes, which is capable for collecting and processing the data of a very large number of subjects, trying to gain more insights of the human brain.

References

[1] IBM, "IBM moves closer to creating computer based on insights from the brain," http://www-03.ibm.com/press/us/en/pressrelease/28842.wss, 11 2009.

[2] "IBM unveils a new brain simulator," http://spectrum.ieee.org/computing/hardware/ibm-unveils-a-new-brain-simulator.

[3] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nat Rev Neurosci*, vol. 10, pp. 186–198, 2009.

[4] Y. He, Z. Chen, G. Gong, and A. Evans, "Neuronal networks in Alzheimer's disease," *Neuroscientist*, vol. 15, no. 4, pp. 333–350, 2009.

[5] O. Sporns, G. Tononi, and R. Ktter, "The human connectome: A structural description of the human brain," *PLoS Comput Biol*, vol. 1, no. 4, p. e42, 09 2005.

[6] K. Supekar, V. Menon, D. Rubin, M. Musen, and M. D. Greicius, "Network analysis of intrinsic functional brain connectivity in Alzheimer's disease," *PLoS Comput Biol*, vol. 4, no. 6, p. e1000100, 06 2008.

[7] Y. He, J. Wang, L. Wang, Z. J. Chen, C. Yan, H. Yang, H. Tang, C. Zhu, Q. Gong, Y. Zang, and A. C. Evans, "Uncovering intrinsic modular organization of spontaneous brain activity in humans," *PLoS ONE*, vol. 4, no. 4, p. e5226, 04 2009.

[8] J. Wang, L. Wang, Y. Zang, H. Yang, H. Tang, Q. Gong, Z. Chen, C. Zhu, and Y. He, "Parcellation-dependent small-world brain functional networks: a resting-state fMRI study," *Human Brain Mapping*, vol. 30, no. 5, pp. 1511–1523, 2009.

[9] D. A. Fair, A. L. Cohen, J. D. Power, N. U. F. Dosenbach, J. A. Church, F. M. Miezin, B. L. Schlaggar, and S. E. Petersen, "Functional brain networks develop from a local to distributed organization," *PLoS Comput Biol*, vol. 5, no. 5, p. e1000381, 05 2009.

[10] M. Valencia, M. A. Pastor, M. A. Fernández-Seara, J. Artieda, J. Martinerie, and M. Chavez, "Complex modular structure of large-scale brain networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 2, p. 023119, 2009.

[11] N. Xu, X. Cai, R. Gao, L. Zhang, and F. H. Hsu, "FPGA-based accelerator design for RankBoost in web search engines," in *International Conference on Field-Programmable Technology*, 2007, pp. 33–40.

[12] B. Wang, T. Wu, F. Yan, R. Li, N. Xu, and Y. Wang, "RankBoost acceleration on both NVIDIA CUDA and ATI stream platforms," *Parallel and Distributed Systems, International Conference on*, pp. 284–291, 2009.

[13] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig, "GPU-ClustalW: Using graphics hardware to accelerate multiple sequence alignment," in *HiPC*, 2006, pp. 363–374.

[14] Y. Liu, B. Schmidt, and D. L. Maskell, "MSA-CUDA: Multiple sequence alignment on graphics processing units with CUDA," *Application-Specific Systems, Architectures and Processors, IEEE International Conference on*, vol. 0, pp. 121–128, 2009.

[15] J. Cong, V. Sarkar, G. Reinman, and A. Bui, "Customizable domain-specific computing," UCLA Computer Science Department Technical Report, Tech. Rep. TR# 100018, 4 2010.

[16] L. Zhuo and V. K. Prasanna, "High performance linear algebra operations on reconfigurable systems," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2005, p. 2.

[17] ——, "Sparse matrix-vector multiplication on FPGAs," in *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. New York, NY, USA: ACM, 2005, pp. 63–74.

[18] K. D. Underwood and K. S. Hemmert, "Closing the gap: CPU and FPGA trends in sustainable floating-point BLAS performance," *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, vol. 0, pp. 219–228, 2004.

[19] J. Krüger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*. New York, NY, USA: ACM, 2005, p. 234.

[20] L. Huelsbergen, "A representation for dynamic graphs in reconfigurable hardware and its application to fundamental graph algorithms," in *FPGA '00: Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2000, pp. 105–115.

[21] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA," *High Performance Computing*, vol. 4873, pp. 197–208, 2007.

[22] M. Garland, "Sparse matrix computations on manycore GPU's," in *DAC '08: Proceedings of the 45th annual Design Automation Conference*. New York, NY, USA: ACM, 2008, pp. 2–6.

[23] G. J. Katz and J. T. Kider, Jr, "All-pairs shortest-paths for large graphs on the GPU," in *GH '08: Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 47–55.

[24] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "FPMR: MapReduce framework on FPGA," in *FPGA '10: Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2010, pp. 93–102.

[25] *ATI Intermediate Language (IL) Specification*, Advanced Micro Devices, Inc., Dec 2009.

[26] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[27] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E*, vol. 74, no. 3, p. 036104, Sep 2006.

[28] G. Venkataraman, S. Sahni, and S. Mukhopadhyaya, "A blocked all-pairs shortest-paths algorithm," *J. Exp. Algorithmics*, vol. 8, p. 2.2, 2003.