Efficient Pearson Correlation and All-Pairs-Shortest-Path on GPUs for Large-scale Brain Network Analysis

A supplement to our ICCAD 2011 paper

Ling Ren, Mo Xu

Nov 2011

Abstract

In this report, we present the GPU implementations of Pearson correlation and the Blocked Floyd-Warshall algorithm, the two key modules in voxel-based brain network analysis, which are not discussed in detail in our ICCAD'11 paper. Pearson correlation is actually matrix multiplication. For Blocked Floyd-Warshall algorithm, we propose two optimizations based on previous studies. Our methods performs well with very large graphs, and hence is applicable in voxel-based brain network analysis.

1 Introduction

The Human Connectome is a comprehensive structural description of the connectivity of the human brain [1]. It has attracted great research attention. A promising method is to model the human brain as a network based on BOLD (Blood Oxygen Level Dependent) signals acquired from fMRI (functional Magnetic Resonance Imaging) and then to employ graph theory algorithms to analyze the network [2]. This method is known as Brain Network Analysis (BNA).

Voxel-based BNA can take advantage of the improving resolution of the imaging technologies and provide insights into the human brain. The major challenge for voxel-based BNA is the great computational complexity as a result of higher resolution. A voxel-scale brain network usually contains 20K to 100K nodes. Many graph algorithms become intolerably time-consuming when the network gets such large. In our ICCAD paper, we have proposed a heterogeneous (CPU/GPU) accelerator platform for voxel-based BNA. In this report, we present more details about the GPU implementations of Pearson'correlation and Blocked Floyd-Warshall algorithm.

2 Pearson Correlation

The Pearson's correlation [3] between node (v_i, v_j) is defined as follows:

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

where v_i denotes the BOLD series of voxel i, $\bar{v_i}$ is the average of the series of that voxel, L is the length of time serials, and all $\sum$ sums along the whole time series $\sum_1^L$.

Actually, Pearson Correlation is matrix multiplication. To show this, we first normalize the BOLD signal

$$\overrightarrow{u} = \frac{\overrightarrow{v} - \bar{v}}{|\overrightarrow{v} - \bar{v}|} \tag{2}$$

and then denote $\mathbf{U} = (\overrightarrow{u}_\mathbf{1}, \overrightarrow{u}_\mathbf{2} \cdots, \overrightarrow{u}_\mathbf{N})$. The output correlation matrix is

$$\mathbf{R} = \mathbf{U^T U} \tag{3}$$

Matrix multiplication is very efficient on GPU, e.g 1400 Gflop/s on AMD Radeon 5870 GPU [4]. This topic has been thoroughly studied, and we do not elaborate here. Here we discuss the scalability of correlation calculation. Since the size of brain networks are usually very large, the correlation matrix $\mathbf{R}$ often exceeds the GPU memory. So we use the blocked matrix multiplication and divide the matrix into blocks with a preset block size (e.g 2048×2048):

$$\mathbf{U} = [\mathbf{U_1}, \mathbf{U_2} \cdots, \mathbf{U_m}] \tag{4}$$

so that

$$\mathbf{R} = \mathbf{U^T U} = \begin{bmatrix} \mathbf{U_1^T U_1} & \mathbf{U_1^T U_2} & \cdots & \mathbf{U_1^T U_m} \\ \mathbf{U_2^T U_1} & \mathbf{U_2^T U_2} & \cdots & \mathbf{U_2^T U_m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{U_m^T U_1} & \mathbf{U_m^T U_2} & \cdots & \mathbf{U_m^T U_m} \end{bmatrix} \tag{5}$$

Considering the symmetry of the correlation matrix, only the upper half of the matrix is calculated. One block is calculated each time. While this block is transferred back to the host memory, the GPU calculates another block. This implementation can easily scale to networks with higher resolution, without increasing the demand for graphic memory.

3 Blocked Floyd-Warshall Algorithm

In this section, we first briefly review the Block Floyd-Warshall algorithm and its GPU implementation in previous studies, and then introduce our further optimization.

In the Block Floyd-Warshall algorithm, the whole adjacency matrix is first converted to a $N \times N$ cost matrix C, where N is the number of the voxels. C_{ij} is the distance from voxel i to voxel j, or inf if there is no such path. Then the cost matrix is divided into r $n \times n$ sub-blocks, where $r = \lceil N/r \rceil$. The outer loop iterates over the r primary blocks (the blocks along the diagonal of the matrix). In each of the r rounds, all blocks are updated in the similar way with the basic Floyd-Warshall algorithm [5, 6], specifically

$$C_{ij} = min\left(C_{ij}, C_{ik} + C_{kj}\right) \tag{6}$$

where element (k, k) is in the primary block. In each iteration, updating a block needs 1) the block in the same column with itself and in the same row with the primary block, denoted with vertical lines in Fig. 1, and 2) the block in the same row with itself and in the same column with the primary block, denoted with horizontal lines.

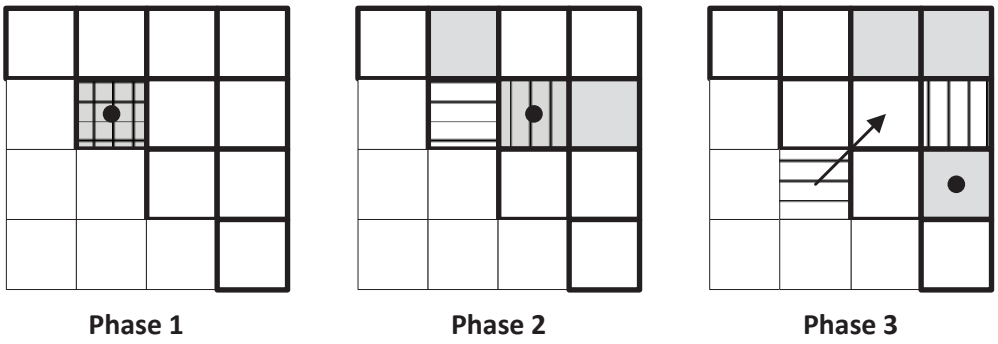

Figure 1: Illustration of Blocked FW algorithm

The basic operations in the BFW algorithm is similar to matrix multiplication [7], or referred to as generalized matrix multiplication. Following the ideas of GEMM, generalized matrix multiplication can also be implemented very efficiently on GPUs. Matsumoto implements the APSP using generalized matrix multiplication on GPU [8].

We further propose two optimizations based on [8]. First, Phase 1 in the BFW algorithm can be done using the basic FW algorithm. But in order to represent all computation with generalized matrix multiplication, another algorithm with the $O(n^3 log_2 n)$ time complexity is adopted in [8]. Actually, it is more efficient to apply the blocked FW algorithm again for Phase 1. In this way, we can take advantage of the module of generalized matrix multiplication, without bring in any extra computation. Second, the brain is often modeled as a symmetric network. In this case, only the upper half of the cost matrix has to be updated, which means only $r(r+1)/2$ blocks have to be updated in each round. But if only the upper half matrix is maintained, some source blocks do not exist Fig. 1, and we need to transpose the blocks at the symmetric location.

4 Conclusion

In this report, we present the GPU implementations of two key components in our accelerating framework for brain network analysis. Our methods are efficient and scale well with the network size.

References

[1] 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.

[2] 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.

[3] 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.

[4] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, and D. J., "From cuda to opencl: Towards a performance-portable solution for multi-platform gpu programming," *Journal of Parallel Computing*, 2011.

[5] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, 1962.

[6] S. Warshall, "A theorem on boolean matrices," *J. ACM*, vol. 9, no. 1, pp. 11–12, 1962.

[7] P. D'Alberto and A. Nicolau, "R-kleene: A high-performance divide-and-conquer algorithm for the all-pair shortest path for densely connected networks," *Algorithmica*, pp. 203–213, 2007.

[8] K. Matsumoto, N. Nakasato, and S. Sedukhin, "Blocked all-pairs shortest paths algorithm for hybrid cpu-gpu system," in *Proceedings of the 2011 IEEE International Conference on High Performance Computing andCommunications (HPCC '11)*, 2011, pp. 145–152.