34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Estimating Correlation for a Real-Time Measure of Connectivity

Akhil Arunkumar, Ashish Panday, Bharat Joshi, Arun Ravindran, and Hitten P. Zaveri

Abstract — **There has recently been considerable interest in connectivity analysis of fMRI and scalp and intracranial EEG time-series. The computational requirements of the pair-wise correlation (PWC), the core time-series measure used to estimate connectivity, presents a challenge to the real-time estimation of the PWC between all pairs of multiple time-series. We describe a parallel algorithm for computing PWC in real-time for streaming data from multiple channels. The algorithm was implemented on the Intel Xeon™ and IBM Cell Broadband Engine™ platforms. We evaluated time to estimate correlation for signals recorded with different acquisition parameters as a comparison to real-time constraints. We demonstrate that the execution time of these efficient implementations meet real-time constraints in most instances.**

Keywords – Pair-Wise Correlation, EEG, Intracranial EEG, fMRI, MRI, Real-time, Intel Xeon, IBM Cell Broadband Engine

I. Introduction

The motivation for this study stems from recent developments in fMRI and EEG. First, the advent of real-time fMRI (rtfMRI) augurs the possibility of real-time diagnosis and behavioral modulation of brain activity based on observed time-series [1] [2]. The marriage of rtfMRI and connectivity measurement would hold the promise of the use of connectivity analysis in real-time for a broad range of applications. Second, as with the connectivity analysis performed with fMRI data, the correlation coefficient is a common time-series analysis measure used for connectivity of scalp EEG and intracranial EEG (icEEG) [3]. Most of the analysis performed on streaming data with compute-intensive kernels like correlation is currently performed offline. Adding to the challenge of this problem is the need to acquire the EEG, icEEG and fMRI time-series with increasingly greater temporal and spatial resolution. That is, the sensor density (number of channels of streaming data) and frequency of monitoring (F_S measured in KHz) is progressively increasing.

The pair-wise correlation (PWC or r) is a single number that describes the degree of relationship between two variables. Pearson's Product-Moment Correlation Coefficient (PPMCC) is widely used to estimate r [4]. The computational requirement for the PPMCC estimator is lower than that of other estimators [5] [6]. In this study we consider the extension of the PWC to multiple channels (MCPWC). That is, the estimation of PWC for all possible pairs that can be formed from a given set of time-series. We compare the execution time of a proposed sliding window protocol (SWP) for MCPWC to real-time constraints (RTC). These tests were performed on two platforms, Intel Xeon cluster (IXC) and IBM Cell Broadband Engine (CBE). MCPWC is characterized by a high degree of parallelism, and implementation of the kernel without careful analysis can lead to redundant computation. To parallelize MCPWC, the job division and load balancing is achieved by partitioning and streaming sampled data to multiple computing elements. In the CBE the partitioned data is streamed to compute-intensive cores called the Synergistic Processing Element (SPE) and in the IXC it is streamed to an available processing core. The factors that impact the performance of the SWP are: (1) choice of MCPWC implementation, (2) organization of data in the main memory, (3) data access strategy, (4) efficiency of job division and load balancing, and (5) hardware specifications of the computing platform. All of these factors were considered while designing the SWP kernel.

II. Methods

A. Multi-Channel Pairwise Correlation

The data collected in fMRI and scalp EEG and icEEG are a set of discrete samples and hence PPMCC can be computed using eqn (1):

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} - \sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}} \quad (1)$$

where,

n : number of elements in set X and Y

x_i : i^{th} element in the set X

y_i : i^{th} element in the set Y

MCPWC can be represented by eqn (2):

A. Arunkumar, A. Panday, B. Joshi, and A. Ravindran are with the Electrical and Computer Engineering Department, University of North Carolina at Charlotte, NC 28223 USA (phone: 704-687-8098; e-mail: {aarunkum, apanday, bsjoshi, arun.ravindran}@uncc.edu).

H. P. Zaveri is with the Department of Neurology, Yale University, New Haven, CT 06520, USA (e-mail: hitten.zaveri@yale.edu)

$$r_{(i,j)} = \frac{n\sum_{k=1}^{n} x_{(i,k)}x_{(j,k)} - \sum_{k=1}^{n} x_{(i,k)} \sum_{k=1}^{n} x_{(j,k)}}{\sqrt{n\sum_{k=1}^{n} x_{(i,k)}^2 - \left(\sum_{k=1}^{n} x_{(i,k)}\right)^2} \sqrt{n\sum_{k=1}^{n} x_{(j,k)}^2 - \left(\sum_{k=1}^{n} x_{(j,k)}\right)^2}} \quad (2)$$

where, $r_{(i,j)}$ is the PWC for time-series i and j, and $0 < i < j \leq m$; m being the number of channels

B. Sliding Window Protocol for MCPWC

The SWP computes MCPWC of samples collected during a predefined analysis window. The number of samples (w) per channel within the analysis window is defined by the window length (T_W) measured in seconds and the sampling frequency F_S. The number of samples, w, is:

$$w = T_W * F_S * 1000 \quad (3)$$

On completion of the MCPWC computation, the analysis window is advanced by a number of samples which is specified by the slide factor (SF). The slide factor is a function of F_S and the desired output resolution $(1/T_R)$. Where T_R is the time interval, measured in seconds, after which the MCPWC output is refreshed. SF can be calculated by (4):

$$SF = T_R * F_S * 1000 \quad (4)$$

The first MCPWC output is available after the initial window has been built. Following that, the output is updated every SF samples. This process of advancing the window and computing MCPWC continues until the user stops the process or the data stream is exhausted.

It is assumed that $w \geq SF$ and that at least one sample is collected before the output is updated ($SF \geq 1$). The optimization efforts for real-time implementation were directed towards not overwhelming system resources such as data collection buffers. To meet the real-time constraint, SWP outputs have to be computed before the next data window is available. Thus, T_R becomes the RTC for the algorithm.

C. Optimization Methods for the Intel Architecture

As indicated in the Introduction, load balancing is achieved by dividing the problem into sub-problems and distributing these sub-problems between the processing cores. For the SWP implementation on the Intel architecture, this can be achieved efficiently by dividing the problems along the time domain (subscript k in (2)) rather than along the channel domain (subscript i in (2)). By doing so, the high penalty bearing repetitive accesses to data elements can be avoided. This concept is illustrated in Fig. 1.

We note the time required to complete the computation and update the output is independent of w when F_S is constant. This is because in the implementation of SWP,

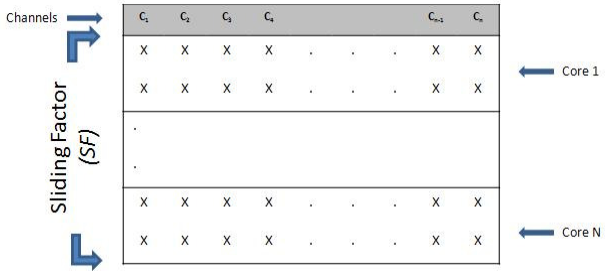

Figure 1. An illustration of PWC load balancing. The samples which have been advanced are split into sub-problems along the time axis and each sub-problem is assigned to a processing core. For example, the first sub-problem is assigned to Core 1 and the N^{th} sub-problem is assigned to Core N.

the calculation of the following three values $S1 = \sum x_i$, $S2 = \sum x_i^2$ and $S3 = \sum x_i y_i$ are performed in blocks of length SF. These blocks are combined to form the initial analysis window and compute the initial MCPWC estimate. SWP avoids redundant computations by using the values of $S1$, $S2$, and $S3$ computed for the previous window and updating them with the corresponding values for the new block of samples collected when the window is advanced, after subtracting the values of $S1$, $S2$, and $S3$ for the first block of samples of the previous window. That is, the newly computed values of $S1$, $S2$, and $S3$ are used to update the MCPWC estimate. Therefore, importantly, once the initial window has been processed, the process to update the MCPCW output requires only the new samples collected when the window is advanced and not the samples for the entire analysis window.

D. Optimization Methods for the CBE

In case of the CBE implementation of SWP, the data is divided along the channel domain (subscript i in (2)). For example, if the number of distinct time-series is 66 and six SPEs are available, then 11 time-series are assigned to each SPE. However, in order to estimate all possible pairs, inter-SPE communication is required. We have previously reported an efficient communication scheme to do this using the "Ring Algorithm" along with a system-specific optimized computation scheme [5] [6]. This communication scheme is very efficient. It contributes to less than 1% of the time profile of the algorithm. The computations of $S1$, $S2$ and $S3$ are performed as described in Section II C.

E. Evaluation Systems

The experimental setup consisted of the following:

a. Intel Xeon cluster system with a dual quad core Intel Xeon X5365 processor cluster; each core consisted of 32 KB L1 instruction cache, 32 KB L1 data cache, and an 8 MB shared L2 cache. The compiler set up consisted of GCC 4.1.2 with O3, Loop Unrolling, and Fast Math optimizations.

b. Sony Playstation3™ with CBE consisting of six compute-intensive cores (SPEs), 256 KB Local Store on each SPE, on-chip 4-way Bidirectional Interconnect Bus and one control-intensive core (PowerPC Processing Element, PPE)

F. Experimental Validation Protocol

We evaluated the SWP kernel with simulated time-series for different parameter values: (1) electrode grids of varying sizes ranging from current standards such as 8 x 8 to 15 x 15, 30 x 30, and 40 x 40, (2) window duration, T_W, was selected to be 0.5, 1, and 2 seconds, (3) sampling frequency, F_S was selected to be 0.25, 0.5, 1, 2, 5, and 10 kHz, (4) the output resolution, T_R, was selected to be 2, 1, 0.5, 0.1, and 0.05 seconds. The values of *w* and *SF* tested are shown in Table 1 and Table 2, respectively. The SWP implementation was evaluated all the grid sizes and for each combination of *w* and *SF* other than cases where *SF* > *w*. Additional tests were performed for fMRI data sets of dimensions up to 64x64x32. The data samples considered for evaluation were simulated floating point values.

III. RESULTS

The results of our tests are presented in Fig. 2 in terms of the percentage of the real-time constraint (PRTC) used for the SWP computation. PRTC is calculated using (5):

$$\text{PRTC} = (\text{Execution Time} / \text{RTC}) * 100 \qquad (5)$$

Table 1. Values of *w* for varying values of T_W and F_S

w		***T_W* (in seconds)**		
		0.5	***1***	***2***
***F_S* (KHz)**	***0.25***	125	250	500
	0.5	250	500	1000
	1	500	1000	2000
	2	1000	2000	4000
	5	2500	5000	1000
	10	5000	10000	20000

Table 2. Values of *SF* for varying values of T_R and F_S

SF		***T_R* (in seconds)**				
		2	***1***	***0.5***	***0.1***	***0.05***
***F_S* (KHz)**	***0.25***	500	250	125	25	12.5
	0.5	1000	500	250	50	25
	1	2000	1000	500	100	50
	2	4000	2000	1000	200	100
	5	10000	5000	2500	500	250
	10	20000	10000	5000	1000	500

A. Results for the IBM Cell Broadband Engine

Figs. 2(a), 2(b), and 2(c) display computation time as PRTC for the CBE for different numbers of channels, *w*, F_S and T_R. The time taken by the implementation scheme increases with an increase in F_S. This is expected because *SF* and thus the computation required for *S1*, *S2* and *S3* increases with the increase in F_S. As expected, the implementation is independent of *w*. The computation is completed within a fraction of the RTC (< 20 %). Real-time evaluation of the SWP was possible for fMRI data sets of dimensions up to 64x64x32 for F_S = 1, 0.5, and 0.33 Hz and T_R = 1, 2, or 3 seconds.

B. Results for the Intel Xeon Cluster

The results obtained for the IXC are displayed in Figs. 2(d), 2(e), and 2(f).

The PRTC is expected to be consistent over varying window lengths. This is because the slide factor, *SF*, governs the amount of computation and communication the application performs thereby affecting kernel performance. This is illustrated in Figs. 2 (d), 2 (e), and 2 (f).

For a given F_S, as output resolution increases (that is as T_R decreases), the slide factor *SF* decreases. We notice that even with decreasing *SF*, PRTC generally increases. This is because the performance gains achieved by parallelization decrease with decrease in *SF* and thereby reduce the effective performance gain achieved.

For moderate grid sizes such as 8 x 8 and 15 x 15, the application meets the RTC for all combinations of F_S and T_R. With the increase in grid size to 30 x 30, the application performs within the RTC for lower values of F_S. As F_S increases to very high values such as 5 kHz and 10 kHz, the RTC is satisfied up to moderate output resolution and is not met for very high output resolutions such as T_R = 0.05 seconds. The execution of SWP for very large grid sizes such as 40 x 40 is within the RTC for low F_S and low output resolutions. At higher values of F_S or high output resolutions, the application is not able to meet the RTC with IXC.

IV. DISCUSSION

The PWC and MCPWC contain a high degree of parallelism that can be meaningfully exploited to achieve real-time estimation of these measures. We have performed a large number of tests over a range of acquisition parameters which justify the use of SWP to estimate MCPWC in real-time using IXC and CBE architectures. We have highlighted instances when SWP could and could not meet the RTC. In most instances the kernel execution time is within the RTC. In these instances the processor is not required during the remainder of the RTC, that is, till the time to initiate the next set of computations. Thus, for (100-PRTC)*RTC/100 seconds the processor can be used for tasks such as data acquisition, signal pre-processing, computation of graph theoretical measures from the MCPWC estimates, and visualization of the output. We intend to optimize the algorithm further for MRI and rtfMRI with larger dimensions. The kernel's memory handling and communication strategies can be fine-tuned to further improve the performance.

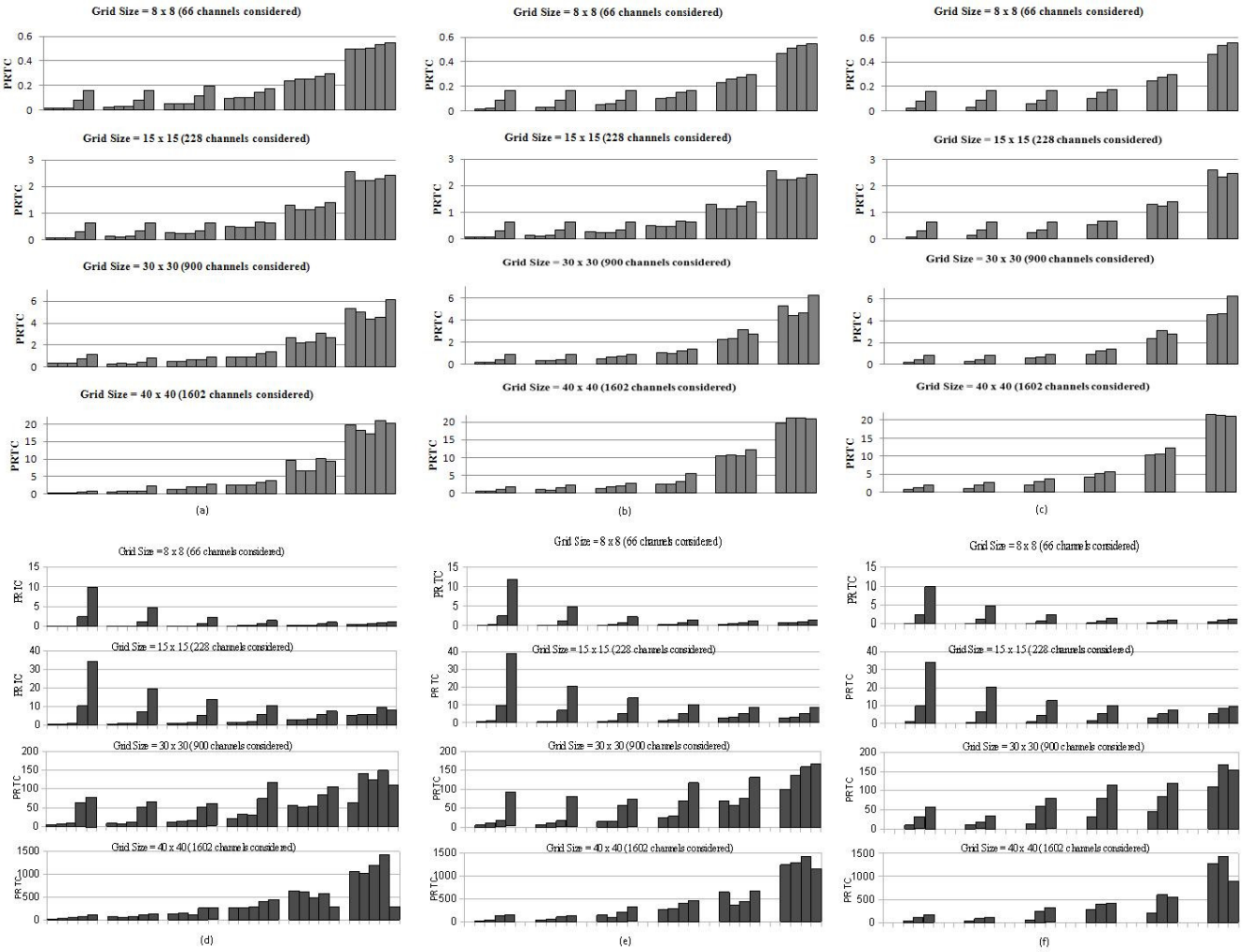

Figure 2. Performance of MCPWC kernel in terms of the percentage of the real-time constraint (PRTC). (a), (b), (c) Performance of MCPWC kernel on IBM Cell Broadband Engine. (d), (e), (f) Performance of MCPWC kernel on Intel Xeon Cluster. (a), (d) Window duration, $T_W = 2$ sec. (b), (e) $T_W = 1$ sec. (c), (f) $T_w = 0.5$ sec. Each plot is grouped into six sub-plots. These sub-plots illustrate performance for $F_S = 0.25$, 0.5, 1, 2, 5, and 10 kHz (from left to right). Each sub-plot contains up to 5 bars representing execution times for $T_R = 2$, 1, 0.5, 0.1, 0.05 sec (from left to right). $T_R = 2$ sec was not considered in (b), (c), (e), and (f), because, for these values of T_W, F_S and T_R, $SF > w$ (see Tables 1 and 2). Similarly, $SF > w$ when $T_R = 1$ sec in (c) and (f). To calculate PRTC for each particular T_R, its corresponding RTC was considered (see Section II A). The real-time constraint was satisfied for all instances (PRTC < 100) for the CBE and in many instances for the IXC.

V. CONCLUSION

We have demonstrated the ability of a parallel sliding window protocol algorithm to meet the real-time requirements for estimating the correlation of multichannel fMRI and scalp and intracranial EEG signals. The algorithm was implemented on two platforms, the Intel Xeon cluster and the IBM Cell Broadband Engine. While superior performance was achieved on the IBM Cell Broadband Engine, we demonstrate that it is possible to estimate correlation in real-time, for selected acquisition parameters, on the more familiar and widely-used Intel Xeon cluster.

REFERENCES

[1] N. Weiskopf, R. Sitaram, O. Josephs, R. Veit, F. Scharnowski, R. Goebel, N. Birbaumer, R. Deichmann, K. Mathiak, "Real-time functional magnetic resonance imaging: methods and applications." Proceedings of the International School on Magnetic Resonance and Brain Function, 2007, pp. 989-1003.

[2] R.C. deCharms, "Applications of real-time fMRI." Nature Reviews Neuroscience, vol. 9, no. 9, pp. 720-729, September 2008.

[3] H. P. Zaveri, S. M. Pincus, I. I. Goncharova, R. B. Duckrow, D. D. Spencer, S. S. Spencer, "Localization-related epilepsy exhibits significant connectivity away from the seizure onset area." NeuroReport, vol. 20, no. 9, pp. 91-95, 2009.

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

[5] A. Panday, B. Joshi, A. Ravindran, J. Byun, H. P. Zaveri, "Study of data locality for real-time biomedical signal processing of streaming data on Cell Broadband Engine" Proceedings of IEEE SoutheastCon 2010 (SoutheastCon), 2010, pp. 123 – 126.

[6] A. Panday, B. Joshi, A. Ravindran, A. Mukherjee, H. P. Zaveri, "Real-time processing of biomedical streaming data on Cell Broadband Engine", **Submitted for publication, 2012.**