2012 IEEE 18th International Conference on Parallel and Distributed Systems

Parallel Processing of Massive EEG Data with MapReduce

Lizhe Wang[1,2†], Dan Chen[2], Rajiv Ranjan[3], Samee U. Khan[4], Joanna Kołodziej[5], and Jun Wang[6†]
1 Center for Earth Observation and Digital Earth, Chinese Academy of Sciences, P.R. China
2 School of Computer, China University of Geosciences, P.R. China
3 Information engineering lab, CSIRO ICT Centre, Australia
4 Department of Electrical and Computer Engineering, North Dakota State University, USA
5 Institute of Computer Science, Cracow University of Technology, Poland
6 Department of Electrical Engineering & Computer Science, University of Central Florida, USA
† Corresponding author: Lizhe.Wang@gmail.com, jwang@eecs.ucf.edu

***Abstract*—Analysis of neural signals like electroencephalogram (EEG) is one of the key technologies in detecting and diagnosing various brain disorders. As neural signals are non-stationary and non-linear in nature, it is almost impossible to understand their true physical dynamics until the recent advent of the Ensemble Empirical Mode Decomposition (EEMD) algorithm. The neural signal processing with EEMD is highly compute-intensive due to the high complexity of the EEMD algorithm. It is also data-intensive because 1) EEG signals contain massive data sets 2) EEMD has to introduce a large number of trials in processing to ensure precision. The MapReduce programming mode is a promising parallel computing paradigm for data intensive computing. To increase the efficiency and performance of the neural signal analysis, this research develops parallel EEMD neural signal processing with MapReduce. In this paper, we implement the parallel EEMD with Hadoop in a modern cyberinfrastructure. Test results and performance evaluation show that parallel EEMD can significantly improve the performance of neural signal processing.**

I. INTRODUCTION

a) Neural signal processing and EEMD: Neural signal analysis is a process of detection, diagnosis, and treatment of brain disorders and the related diseases [21]. Typical neural signals include electroencephalogram (EEG), magnetoencephalography (MEG), electrocorticographic (ECoG), magnetic resonance imaging (MRI), and functional MRI (fMRI).

Neural signals are naturally nonlinear and non-stationary. Researchers have intensively studied linear neural signal processing algorithms, such as short-Fourier transform, Wigner-Ville distribution and wavelet filtering, in spectral analysis of EEG recordings. However, it has been examined that the temporal patterns of neural signals, such as instantaneous amplitude and phase/frequency, cannot be accurately estimated with linear algorithms [20], [9].

The Ensemble Empirical Mode Decomposition (EEMD) algorithm [28], in conjunction with Hilbert-Huang Transform (HHT) [18], has been proposed as a revolutionary solution to the neural signal processing. This method can break down a complicated nonlinear and non-stationary signal into a collection of oscillatory Intrinsic Mode Functions (IMFs). The EEMD algorithm eliminates mode mixing in all cases automatically and excels in resistance to noises, thus making the EEMD algorithm much superior to its linear counterparts in processing neural signals.

The EEMD algorithm demands repetitively processing many trials of the noise-added signal in an ensemble, and the precision of the outputs depends on the number of trials, which should be large enough to neutralize the effect of added noises. Furthermore, as experimental techniques for recording neural activities have been advancing quickly, i.e., rapidly increasing number of channels (electrodes) and sampling frequencies. The density and the spatial scale of neural signals have been increasing exponentially. Therefore the neural signal analysis with EEMD manifests a problem, not only compute-intensive, but also highly data-intensive. Thus it is a natural solution to develop parallel EEMD neural signal processing with high performance computing architectures.

b) Parallel Neural Signal Processing: There are mainly two research directions for high performance neural processing:

- Effective storage and management of complex and massive experimental data in a distributed environment:
  [17] proposes a solution of management of massive neuroimaging data by integrating database management systems (DBMS) with Grid computing and cluster-based computing. The approach centers on a DBMS, which first facilitates storing and sharing fMRI data and eases the analysis of these data. [16] proposes a high-performance scheme for the EEG compression using a multichannel model. SignalML, a meta-format for description of biomedical time series storage, is discussed in [13]. [6] presents a compressed sensing framework for EEG compression.
- Parallel neural signal processing with HPC architectures and paradigms:
  Eichner et. al. [14] develop a neural simulation on multicore architectures. Chen et. al. [10] and Wilson et. al. [27] use GPGPU to parallelize the neural processing algorithms. A Beowulf cluster is employed in [29] for parallel EEG processing. Laubach et. al. [19] develop a parallel architecture of cluster of workstations for on-line

1521-9097/12 $26.00 © 2012 IEEE
DOI 10.1109/ICPADS.2012.32

analyses of neurophysiological data. Muller et. al. [22] present a client – server application for the distributed multivariate analysis of time series using standard PCs. Date et. al. [11] and Buyya et. al. [7] develop Grid computing infrastructures for distributed EEG data processing and visualization. [15] implements parallel EEG processing with the Granules, a lightweight runtime for orchestrating a large number of computations in a Cloud.

c) Why MapReduce and Hadoop?: Although there have been significant progresses made in parallel neural signal processing as discussed above, it is still hard to reach the objective of reliable, high-throughput processing of massive neural signals. For example, it is very likely that certain computing elements, such as hard disk or processor, would encounter failures during the neural signal processing. This is especially unacceptable in a real-time neural signal processing use case. Furthermore, the data processing ability of current fine-grain parallel processing paradigms, such as GPGPU [3] and OpenMP [4], are not satisfactory due to the limited processing capacity and storage size on a single server. On the other hand, the coarse-grain parallel processing paradigm, for example, the distributed data-intensive workflow processing lacks of the ability of "high throughput". Therefore new computing paradigms and platforms are of demand for parallel neural signal processing.

MapReduce [12] is a new data parallel processing model and Hadoop [1] is its open-source implementation on clusters (Further technical information will be discussed in IV. Compared with existing parallel processing paradigms, MapReduce and Hadoop have two advantages: 1) fault-tolerant storage and data processing by duplicating computing tasks on different compute nodes; 2) high-throughput data processing via a batch processing model and a highly efficient massive file system – HDFS. Our research in this paper thus employ the MapReduce paradigm for parallel neural signal processing.

d) Research Contribution: In our research, we develop a parallel EEMD algorithm with the MapReduce programming model and implement it with Hadoop in a modern Cyber-infrastructure – the FutureGrid testbed. To the best of our knowledge, this paper is the first attempt to parallelize the EEMD processing with the MapReduce paradigm.

e) Paper Organization: The rest of this paper is organized as follows: Section II introduces the structure of neural signal investigated in this research and Section III discusses the Hilbert-Huang transform & Ensemble Empirical Mode Decomposition (EEMD) method. The MapReduce and Hadoop are investigated in Section IV. Section V presents the design and implementation of parallel neural signal processing with MapReduce and Section VI gives and performance evaluation of our design and implementation. Finally Section VII concludes the research and points out the future work.

II. STRUCTURES OF NEURAL SIGNALS FOR PROCESSING

The EEG data sets are captured using multiple electrodes simultaneously from an epilepsy patient continuously over a period of several hours or days. As a result, the neural signals under investigation are massive and are persisted in the form of a multi-channel time series. The volume of the EEG dataset prior to analysis scales in three dimensions: the time (T), the number of channels (NC), and the sampling rate (SR). The total number of variables in the multi-channel time series is $T \times NC \times SR$.

When processing the source EEG dataset with EEMD, we apply a sliding time-window to each individual EEG time series, corresponding to one channel, to generate numerous short time series (epochs) covered by the windows. There exists an overlap of a fixed length between two consecutive epochs. We write the ratio of the overlaps length against an epoch's length as ro. Clearly, the size of data in the form of epochs "swells" to $1/(1-ro)$ of the source EEG dataset. Obviously the number of epochs that may be extracted from a source time series is proportional to T.

On the initial stage of an EEMD method, each epoch is added with white noise repeatedly to generate an ensemble of trials, where each trial represents an instance of the original epoch mixed with noise. A trial is the elementary data chunk computed by the EEMD algorithm. The number of trials (NT) in an ensemble should typically be in hundreds. As shown in Figure 1, after initialization, and immediately before being fed to the EEMD algorithm, the size of the data, in the form of trials, explodes by hundreds of times from the source EEG dataset.

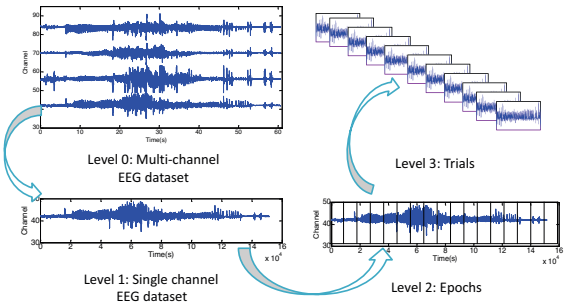

Fig. 1. The Multi-level of EEG dataset

The formation of the initialized EEG data set can be viewed as a three-level hierarchy. Each EEG data channel (time series) can be singled out from the whole multi-channel dataset at the top level. At the middle level, a number of epochs can be generated from an EEG time series by applying a sliding time window. At the bottom level, an epoch spawns off numerous trials by adding white noise to itself.

The order of processing different trials has no influence on the final results as long as those trials rooted from the same source epoch are properly bundled. There exist a temporal parallelism along the X dimension (time T, normally in hours) and a spatial parallelism along Y dimension (data channels, NC, in hundreds) and Z dimension (trials, NT, higher than 1kHz) in the initialized dataset in the form of trials. From the top level to the bottom level (see Figure 2), the grain of parallelism becomes exponentially finer. Given a fixed number of data channels, the very fine-grained parallelism in the other

two dimensions is focused.

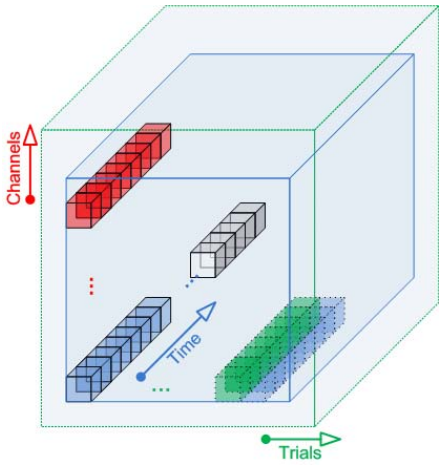

Fig. 2. The Multi-dimension of EEG dataset

III. ENSEMBLE EMPIRICAL MODE DECOMPOSITION

A. Overview of EEMD

The EEMD algorithm is an improvement of EMD. Hilbert-Huang Transform (HHT) introduces the concept of Intrinsic Mode Function (IMF) and Empirical-Mode Decomposition (EMD). As shown in Figure 3, for a non-stationary signal $x(t)$ the EEMD method can decompose the signal into a series of Intrinsic Mode Functions (IMF): imf^i, $i = 1, 2, \ldots, I$, I is the number of IMFs [18].

The signal $x(t)$ can be express as follows:

$$x(t) = \sum_{i=1}^{p} imf^i(t) + r^p(t) \tag{1}$$

where, $r^p(t)$ is the p^{th} residual signal after p IMFs have been decomposed from $x(t)$.

Applying the Hilbert transform and the concept of Shannon entropy, the Hilbert-Huang Spectral Entropy (HHSE) can be constructed, which denotes a more accurate and nearly continuous distribution of the signals energy. IMFs and HHSE precisely characterize the true physical nature of the signal $x(t)$ [18], [28].

B. EEMD Algorithm

Given the size of a time window and the overlap of two consecutive windows, the whole procedure for processing an epoch of signal in a time window, $x(t)$, is portrayed as the following steps. Steps 1 – 7 decompose a signal to obtain its true IMFs then Step 8 calculates its HHSE :

1) Calculate the number of IMFs in $x(t)$:

$$I = log_2\{length[x(t)]\} - 1 \tag{2}$$

Set the amplitude of white noise to be added to $x(t)$;
Set the number of trials in the ensemble: K. k is the variable that represents the k^{th} trial, initialize $k = 1$.

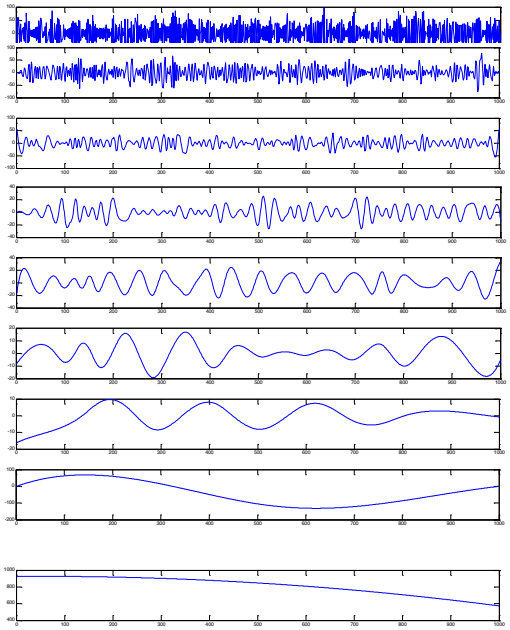

Fig. 3. Decomposition of an EEG epoch into averaged Intrinsic Mode Functions

2) In the k^{th} trial, set

$$x_k(t) = x(t) + n_k(t) \tag{3}$$

where $n_k(t)$ denotes the write noise;
i is the variable that represents the i^{th} IMF decomposition. Initialize $i = 1$, set the residual signal,

$$r_k^i(t) = x_k(t) \tag{4}$$

3) When extracting the i^{th} (in the k^{th} trial) IMF, set

$$h_{j-1}(t) = r_k^i(t) \tag{5}$$

Initialize $j = 1$ and calculate local maxima $h_{max}(t)$ and minima $h_{min}(t)$ of $h_{j-1}(t)$.

4) Interpolate $h_{max}(t)$ and $h_{min}(t)$ using Cubic Spline Interpolation lines, thus to extract the upper and lower envelopes of $h_{j-1}(t)$;
Eventually $m_{j-1}(t)$, the mean of the upper and lower envelopes, can be calculated;

$$h_j(t) = h_{j-1}(t) - m_{j-1}(t) \tag{6}$$

5) Check whether $h_j(t)$ is an IMF. When the termination requirement is satisfied, then goto Step 3 with $j = j+1$; Otherwise, the i^{th} IMF in the k^{th} trial is obtained,

$$imf_k^i(t) = h_j(t) \tag{7}$$

Set

$$r_k^{i+1}(t) = r_k^i(t) - imf_k^i(t) \tag{8}$$

as the new residual signal for sifting the $(i+1)^{th}$ possible IMF;

6) If $r_k^{i+1}(t)$ still has at least 2 extrema, then step 3 will be returned with $i = i + 1$; Otherwise, the decomposition

procedure in the k^{th} trial stops. Thus the noise-added signal $x_k(t)$ can be decomposed in the following form:

$$x_k(t) = \sum_{i=1}^{I} imf_k^i(t) + r_I(t) \tag{9}$$

where $r_I(t)$ is the residual signal of the $x_k(t)$ after I IMFs are decomposed.

7) If $k < K$, go to Step 2 with $k = k + 1$; Otherwise, the present trials have been completed; The i^{th} IMF of the average of the results in all the trials is:

$$\overline{imf^i(t)} = \frac{\sum_{k=1}^{K} imf_k^i(t)}{K} \tag{10}$$

where $i = 1, 2, \ldots, I$.

8) Based on $imf_i(t)$, $i = 1, 2, \ldots, I$, we can eventually obtain the HHSE of $x(t)$ as follows:

- Apply the Hilbert transform on IMF, we then have:

$$Z(t) = imf(t) + iH[imf(t)] \tag{11}$$

$$= a(t)e^{i\int \omega(t)dt} \tag{12}$$

where,

$$a(t) = \sqrt{imf^2(t) + H^2[imf(t)]} \tag{13}$$

$$\omega(t) = \frac{d}{dt}\Big(atg\big(\frac{H[imf(t)]}{imf(t)}\big)\Big) \tag{14}$$

$$h(\omega) = \int H(\omega, t)dt \tag{15}$$

- Then we can calculate the HHSE as follows:

$$HHSE = \frac{H}{log(I)} \tag{16}$$

where, I is the number of frequencies,

$$H = -\sum_{j} \hat{h}(f) log(\hat{h}(f)) \tag{17}$$

$$\hat{h}(f) = \frac{h(f)}{\sum h(f)} \tag{18}$$

From the presentation of above algorithms, we can find that:

- The EEMD calculation is very data intensive due to a number of additional trials introduced in the epoch level signal processing.
- The data set processed by EEMD is massive as the real-time neural signal data contains sources from multiple channels, epochs and trials.
- The EEMD algorithm has multiple levels of parallelism in the epoch level and in the trial level.

Therefore in this research we present our work of the massive neural signal processing with MapReduce on Hadoop, which allows to distribute the massive EEG data on various nodes and analyze it in parallel with a batch mode. The next section we will introduce the technical details of MapReduce and Hadoop.

IV. MAPREDUCE PROGRAMMING MODEL

The MapReduce [12] programming model is inspired by two main functions commonly used in functional programming: Map and Reduce. The Map function processes key/value pairs to generate a set of intermediate key/value pairs and the Reduce function merges all the same intermediate values. Many real-world applications are expressed using this model. One framework that implements MapReduce is Hadoop [1], which allows applications to run on large clusters built from commodity hardware. The Hadoop framework transparently provides both reliability and data transfer.

Our work of parallel neural signal processing is implemented in the Hadoop common framework. Therefore this section discusses the background of the Hadoop common framework, which includes two parts: the Hadoop MapReduce framework and the Hadoop Distributed File System (HDFS).

V. DESIGN AND IMPLEMENTATION OF PARALLEL EED PROCESSING WITH MAPREDUCE

A. Design of Parallel EEG Processing with MapReduce

The parallelism of an EEMD processing with the EEMD algorithm can be characterized in at least two levels (Figure 1):

- Epoch level
  The EEMD procedure for an epoch of time series is treated as a whole at this level. The data in an epoch are input to the same EEMD procedure individually, and the outputs from any instance of EEMD procedure will not be consumed by another. The degree of parallelism is the number of epochs, which increases linearly with the size of the EEG dataset.
- Trial level
  A trial (a noise-added epoch) is treated as whole at this level. Given a number of trials per EEMD instance, the decomposition of each trial is always performed independently from the others. The IMFs of an original epoch are only inferred after computing the ensemble of trials. The degree of parallelism is the number of trials per ensemble times the number of epochs. A task at trial level only handles a short time series.

Therefore we design the parallel EEG processing with a hierarchical MapReduce as follows (Figure 4):

- Each epoch is processed by one MapReduce job executed by a Hadoop cluster (shown in Figure 4 (1)).
- For one MapReduce job, we have K trials processed by one Map tasks: k^{th} mapper, $k = 1, \ldots, K$. The k^{th} mapper decomposes all $imf_k^i(t)$ for the k^{th} trials, $i = 1, \ldots, I$. Then we have one reducer , which calculates $imf^i(t)$ for all K trials and HHSE (shown in Figure 4 (2)).

B. A Cyberinfrastructure for Parallel EEG Processing with Hadoop

A cyberinfrastructure [24], [26] is developed to handle parallel EEG processing with Hadoop. It incorporates a middleware for Cloud computing [8], [23], [25] – a dynamic

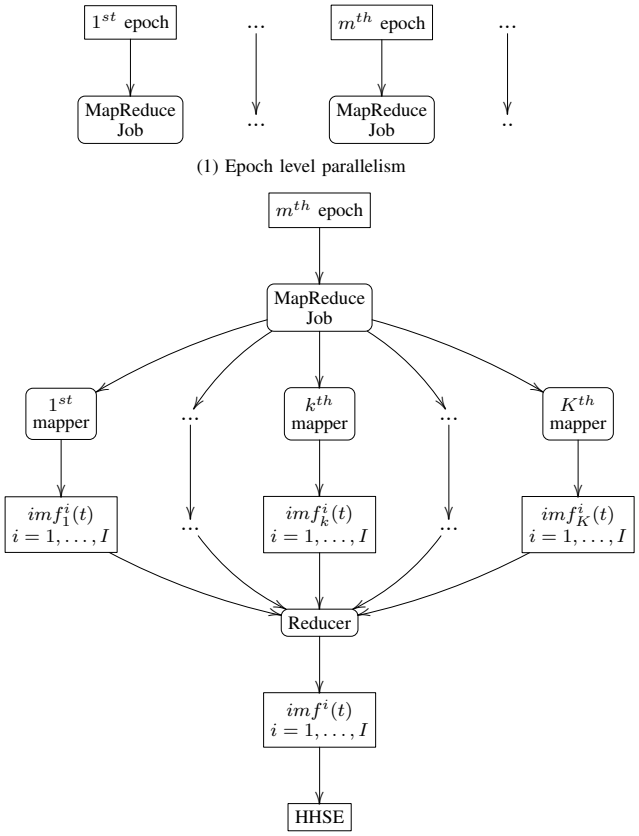

Fig. 4. Parallel EEG data processing with Hadoop

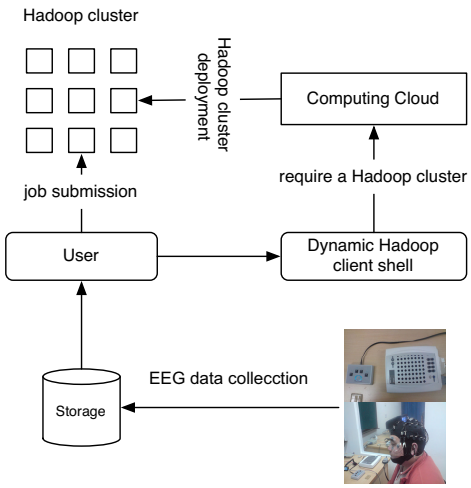

Fig. 5. Cyberinfrastructure for parallel EEG processing

Hadoop cluster client shell, a Hadoop cluster, parallel EEG calculation, and sensor networks which provide the real-time EEG signal retrieval. Figure 5 overviews the process for the parallel EEG processing in the cyberinfrastructure:

1) EEG data are retrieved by a user and stored in user's storage device.
2) The user requires a dynamic Hadoop cluster from a computing cloud test bed (e.g., the "India" cluster in our context) based on the input EEG data set.
3) A Hadoop cluster is dynamically deployed and returned to the user. A shell script is developed for dynamically deployment of a Hadoop cluster with the following steps:
   a) based on user's requirement, it demands a number of cluster nodes from the cluster job manager, such as Torque [5] in our implementation;
   b) distribute Hadoop packages to the master node and slave nodes in the Hadoop cluster;
   c) configure Hadoop configuration files with the available node IP addresses achieved in step 3(a).
   d) start Hadoop master/slave daemons on the Hadoop cluster;
   e) return the Hadoop the IP address and the port number of the Hadoop master node to the user.
4) The user then uploads the EEG data to the HDFS and submits parallel EEG processing application to the Hadoop cluster for the job execution.

VI. TESTS AND PERFORMANCE EVALUATION

A. Test organization

We test the parallel EEG processing application in Hadoop clusters for performance evaluation. The Hadoop clusters are dynamically allocated in a HPC cluster – "India", hosted by an academic Cloud computing test infrastructure – the FugureGrid test bed [2].

TABLE I
TEST BED

Resource	India
Site	Indiana University
CPU number	256
Performance (Teraflop)	11
Total RAM (GB)	3072
Secondary storage (TB)	335

We set up an 8-node Hadoop cluster from "India" as described in V-B. On the Hadoop cluster, we executed the parallel EEG processing in the following conditions:

- various input EEG data set with different size: 4KB, 8KB, . . . , 40KB;
- various MapReduce task configurations: 1, 2 and 3 mapper tasks per node.

B. Test results

The EEG data for test were recorded from scalp surface (F3, F4, C3, C4, O1 and O2 by the International 10-20 System)

using six electrodes. A segment of 6-channel EEG recordings (length of 80s) covering the period of an absence seizure (onset at 63s) is shown in Figure 6.

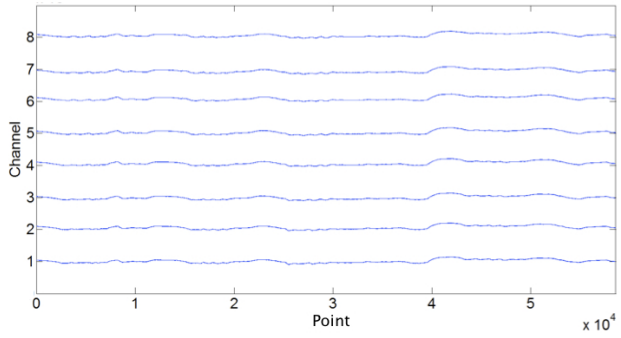

Fig. 6. A segment of 6-channel EEG recordings

The parameter settings for the test are Epoch = 1000, Overlap = 750 The amplitude of white noise for an EEMD trial $= 0.1\times$ (standard deviation of an epoch), and 100 trials per ensemble. The HilbertHuang spectrum of the first channel is illustrated in Figure 7 – 9: Figure 7 shows the EEG segments, Figure 8 and Figure 9 present the corresponding Hilbert – Huang spectrum and the marginal spectrum respectively.

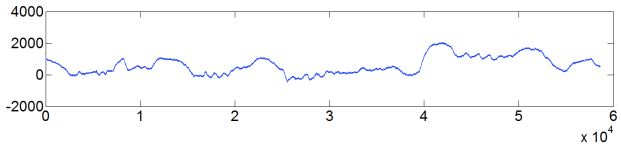

Fig. 7. EEG epochs of the second channel

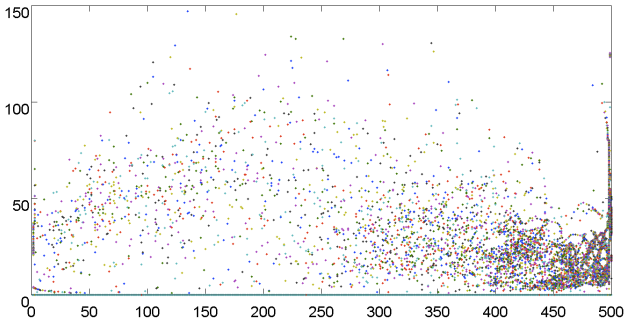

Fig. 8. The corresponding Hilbert-Huang spectrum

Then we calculate the HHSE within the band of 2-30Hz. Figure 10 and Figure 11 display the time courses of HHSE of all six channels of the EEG recordings.

Lines represent the mean values of entropy for 6 channels and bars represent the standard deviation

C. Performance evaluation

Firstly we examined the overhead to setup a Hadoop cluster dynamically and import data to HDFS. As shown in Table II, the setup of a Hadoop cluster with 8 nodes is less than 5 seconds. This is acceptable as once the cluster is setup it can be used for multiple executions until it is shutdown. The import of test EEG data to HDFS is less 1 second. This overhead is also acceptable and it is invoked every time a user executes a MapReduce task in the cluster.

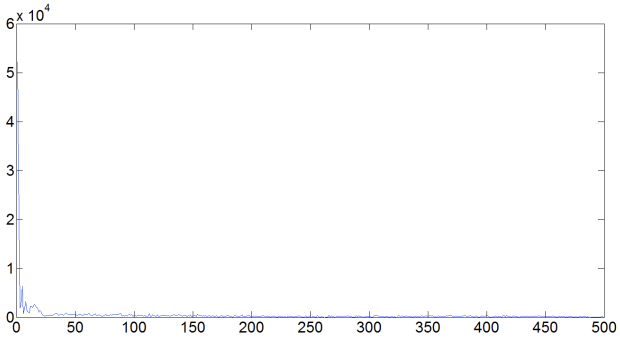

Fig. 9. The marginal spectrum

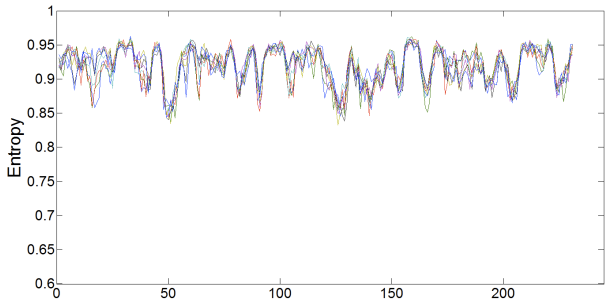

Fig. 10. Time courses of Hilbert-Huang spectral entropy of 6 channels EEG recordings

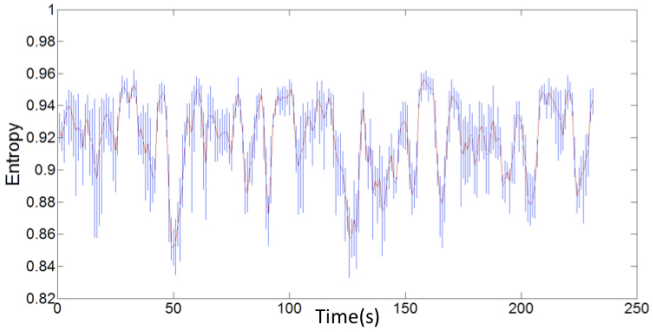

Fig. 11. Lines represent the mean values of entropy for 6 channels and bars represent the standard deviation

TABLE II
OVERHEAD OF MAPREDUCE EXECUTION ENVIRONMENT SETUP

Step	Overhead
Setup a Hadoop cluster (8 nodes)	$<$ 5 seconds
Import EEG test data to HDFS	$<$ 1 second

Secondly we examined the task execution time of parallel EEG data processing with MapReduce. Table III and Figure 12 show the test results. We have two fundings form the test results:

- The job execution time is in direct proportion to the input EEG data size. This funding verifies the scalability of MapReduce parallel processing paradigm, which is declared as one of the Hadoop's merits.
- The optimal value of mapper number per node is 2. This value is determined by the application and compute resource (e.g. memory, processor) per node. As the overall resource per node is limited and constant, therefore too many or two small mapper number may decrease the performance of parallel processing.

TABLE III
JOB EXECUTION TIME OF PARALLEL EEG PROCESSING WITH HADOOP

EEG data size (KB)	mapper(s)/node		
	1	2	3
4	139.417	86.062	103.038
8	243.758	136.484	184.257
12	347.691	198.313	245.429
16	474.293	242.628	314.672
20	567.431	305.858	384.134
24	671.96	362.254	449.223
28	786.107	417.992	527.438
32	882.656	477.141	600.601
36	1005.863	543.588	663.078
40	1113.151	584.594	733.973

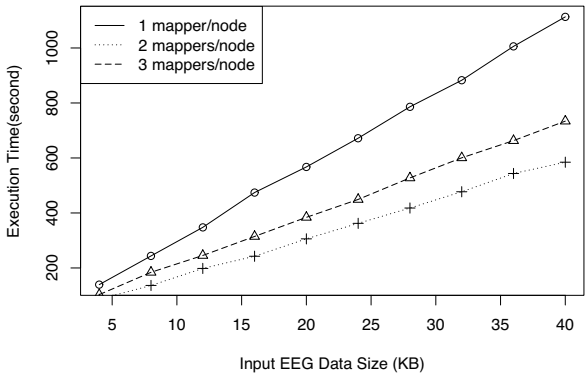

Fig. 12. Execution time of parallel EEG processing with Hadoop

VII. CONCLUSION AND FUTURE WORK

Recently Ensemble Empirical Mode Decomposition (EEMD) has become a revolutionary solution to neural signal processing. As presented in the paper, neural signal processing with EEMD is both compute-intensive and data-intensive. The MapReduce computing paradigm and its Hadoop implementation emerge as a widely-accepted programming model and solution for data intensive computing. This research has proposed a parallel neural signal processing with EEMD with the MapReduce paradigm. We develop multiple parallelism in the EEMD neural signal processing: the epoch-level parallelism and the trial-level parallelism. We implement the parallel EEMD processing with an advanced cyberinfrastructure – a dynamic Hadoop cluster on the FutureGrid test bed. Test results and performance evaluation justify our design and implementation. In our future work, we continue developing the EEMD on large Hadoop clusters to exploit multiple level parallelisms. Also we will implement highly efficient runtime support for uploading large EEG data set to the HDFS.

ACKNOWLEDGEMENT

Dr. Lizhe Wang's work is supported by "One Hundred Talents Programme" of Chinese Academy of Sciences.

Dr. Dan Chen's work is supported in part by National Science Fund for Distinguished Young Scholars (grant No. 61025019), the National Natural Science Foundation of China (grants No. 90820016, 60804036), the Hundred University Talent of Creative Research Excellence Programme (Hebei, China), the Programme of High-Resolution Earth Observing System (China), and the Fundamental Research Funds for the Central Universities (CUGL100608, CUG, Wuhan).

Dr. Jun Wang's work is supported in part by the US National Science Foundation Grant CCF-0811413, CNS-1115665, and National Science Foundation Early Career Award 0953946.

Dr. Samee U. Khan's work is partly supported by the Young International Scientist Fellowship of the Chinese Academy of Sciences, (Grant No. 2011Y2GA01).

The test bed used in the research is supported by the FutureGrid project supported in part by the National Science Foundation under Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Testbed."

REFERENCES

[1] Apache hadoop project. Web Page. http://hadoop.apache.org/.
[2] Futuregrid project. Web Page. http://www.futuregrid.org/.
[3] Gpgpu. Website. http://gpgpu.org/.
[4] Openmp. Website. http://openmp.org/.
[5] Torque resource manager. Website.
[6] Selin Aviyente. Compressed sensing framework for eeg compression. In *Proceedings of the 2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 181–184, Washington, DC, USA, 2007. IEEE Computer Society.
[7] Rajkumar Buyya, Susumu Date, Yuko Mizuno-Matsumoto, Srikumar Venugopal, and David Abramson. Neuroscience instrumentation and distributed analysis of brain activity data: a case for escience on global grids. *Concurrency and Computation: Practice and Experience*, 17(15):1783–1798, 2005.
[8] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw., Pract. Exper.*, 41(1):23–50, 2011.
[9] Dan Chen, Duan Li, Muzhou Xiong, Hong Bao, and Xiaoli Li. GPGPU-aided ensemble empirical-mode decomposition for EEG analysis during anesthesia. *IEEE Trans. Info. Tech. Biomed.*, 14:1417–1427, November 2010.

[10] Dan Chen, Lizhe Wang, Gaoxiang Ouyang, and Xiaoli Li. Massively parallel neural signal processing on a many-core platform. *Computing in Science and Engineering*, 2011.

[11] Susumu Date, Shinji Shimo jo, Mizuno-Matsumoto Yuko, Bu Sung Lee, Wentong Cai, and Lizhe Wang. Distributed processing and visualization of meg data. In *Proceedings of International Conference on Scientic and Engineering Computation (IC-SEC'02)*, pages 850 – 855, Singapore, Dec. 2002.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[13] Piotr J. Durka and Dobies Ircha. Signalml: metaformat for description of biomedical time series. *Comput. Methods Prog. Biomed.*, 76:253–259, December 2004.

[14] Hubert Eichner, Tobias Klug, and Alexander Borst. Neural simulations on multi-core architectures. *Frontiers in neuroinformatics*, 3(July), 2009.

[15] Kathleen Ericson, Shrideep Pallickara, and Charles W. Anderson. Analyzing electroencephalograms using cloud computing techniques. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 185 – 192, Dec. 2010.

[16] D. Gopikrishna and Anamitra Makur. A high performance scheme for eeg compression using a multichannel model. In *Proceedings of the 9th International Conference on High Performance Computing*, HiPC '02, pages 443–451, London, UK, UK, 2002. Springer-Verlag.

[17] Uri Hasson, Jeremy I Skipper, Michael J Wilde, Howard C Nusbaum, and Steven L Small. Improving the analysis, storage and sharing of neuroimaging data using relational databases and distributed computing. *NeuroImage*, 39(2):693–706, 2008.

[18] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, March 1998.

[19] M. Laubach, Y. Arieh, A. Luczak, J. Oh, and Y. Xu. A cluster of workstations for on-line analyses of neurophysiological data. In *Bioengineering Conference, 2003 IEEE 29th Annual, Proceedings of*, pages 17 – 18, march 2003.

[20] Xiaoli Li, Duan Li, Zhenhu Liang, Logan J. Voss, and Jamie W. Sleigh. Analysis of depth of anesthesia with hilbert-huang spectral entropy. *Clinical Neurophysiology*, 119(11):2465 – 2475, 2008.

[21] Dennis McFarland, A. Lefkowicz, and Jonathan Wolpaw. Design and operation of an eeg-based brain-computer interface with digital signal processing technology. *Behavior Research Methods*, 29:337 – 345, 1997.

[22] Andy Muller, Hannes Osterhage, Robert Sowa, Ralph G. Andrzejak, Florian Mormann, and Klaus Lehnertz. A distributed computing system for multivariate time series analyses of multichannel neurophysiological data. *Journal of Neuroscience Methods*, 152:190 – 201, 2006.

[23] Rajiv Ranjan, Liang Zhao, Xiaomin Wu, Anna Liu, Andres Quiroz, and Manish Parashar. Peer-to-peer cloud provisioning: Service discovery and load-balancing. In Nick Antonopoulos and Lee Gillam, editors, *Cloud Computing*, volume 0 of *Computer Communications and Networks*, pages 195–217. Springer London, 2010.

[24] Lizhe Wang and Cheng Fu. Research advances in modern cyberinfrastructure. *New Generation Comput.*, 28(2):111–112, 2010.

[25] Lizhe Wang, Marcel Kunze, Jie Tao, and Gregor von Laszewski. Towards building a cloud for scientific applications. *Advances in Engineering Software*, 42(9):714–722, 2011.

[26] Lizhe Wang, Gregor von Laszewski, Andrew J. Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New Generation Comput.*, 28(2):137–146, 2010.

[27] Adam J Wilson and Justin C Williams. Massively parallel signal processing using the graphics processing unit for real-time braincomputer interface feature extraction. *Frontiers in neuroengineering*, 2:11, 2009.

[28] Zhaohua Wu and Norden E. Huang. Ensemble Empirical Mode Decomposition: a Noise-Assisted Data Analysis Method. *Advances in Adaptive Data Analysis*, 1(1):1–41, 2009.

[29] Yufeng Yao, Jinyi Chang, and Kaijian Xia. A case of parallel eeg data processing upon a beowulf cluster. In *Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*, ICPADS '09, pages 799–803, Washington, DC, USA, 2009. IEEE Computer Society.