ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(19), DOI: 10.17485/ijst/2016/v9i19/93835, May 2016* ISSN (Online) : 0974-5645

A Survey on Parallelization of Neural Network using MPI and Open MP

P. Chanthini* and K. Shyamala

PG & Research Department of Computer Science, Dr. Ambedkar Government Arts College (Autonomous), University of Madras, Chennai-600 039, Tamil Nadu, India;
chanthini19@gmail.com, shyamalakannan2000@gmail.com

Abstract

Background: Neural networks are relatively crude computerized model based on the neural system of the human brain. The Complex problems may require sophisticated processing techniques to achieve practical speed. In human brain millions of neurons form a massively parallel information system. **Method:** A neural network is a parallel and distributed process. The parallel execution of the neural network is achieved up to the level of the training process. Parallelization approaches may work well on hardware implementations, a software package (SPANN), special purpose hardware and multicore CPUs through MPI. **Findings:** In order to achieve parallelism and speed up the training process, each neuron and full neural network is duplicated to multiple threads. In modern Microprocessor number of cores is rapidly increasing. So high-performance computing is a great challenge for the application developers. **Improvements:** Our future work directs to distribute neurons to multiple threads for parallel execution and duplicate the full neural network for parallel training.

Keywords: MPI, Multi-threaded and Multi-core System, OpenMP, Parallelization

1. Introduction

Artificial Neural Networks (ANNs) are an extremely powerful computational and non-linear information processing device[1]. It has the capability to solve all kind of problems like Classification, Pattern Recognition, and functional approximation etc. Neural networks as universal approximators[2], because it is a powerful tool to emulate terribly complicated non-linear dependencies. The goal of ANN is to utilize technology and construct machines that will work like the brain of humans[3]. Massive parallelism makes the Neural Networks very efficient. This may make machines more powerful and intelligent, relieve human's tedious tasks, and may even improve upon human performance.

The authors observed that training of neural network is very difficult task[4] for complicated and multi-dimensional problems with a huge number of patterns. The training process can take along period of time even months to achieve the desired accuracy such that the author has decided to speed up the training process of neural networks, especially for very large training data sets. The datasets selected for parallel training are Character recognition, Surface Interpolation, Ozone extrapolation and O_2 A-band simulation shown in Table 1.

Table 1. Simulation datasets used for training[4]

Dataset				Neural network		
Name	**No. of patterns**	**No. of inputs**	**No. of outputs**	**Training algorithms**	**No. of layers**	**No. of neurons**
Character Recognition	3823	64	10	Backprop&RProp	5	177
Surface Interpolation	7840	2	1	Backprop&QuickProp	4	53
Ozone Extrapolation	120072	2	1	Backprop&RProp	4	53
O2 A-band Simulation	1998855	8	62	Backprop&QuickProp	5	126

* *Author for correspondence*

The author focused their attention on multithreaded and multi-core CPUs to implement the parallel training of neural network. The proposed work is to design the network and duplicate the network over the multiple threads and train parallel. In Figure 1, they have shown the comparison between sequential and parallel training architecture.

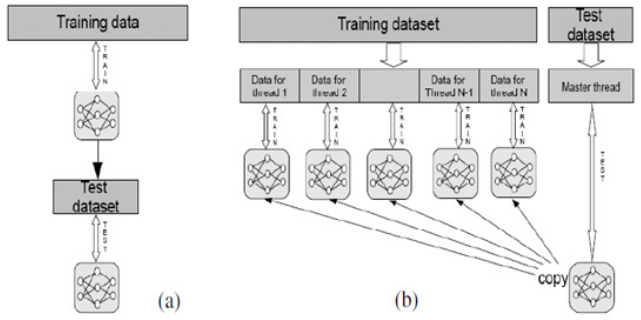

Figure 1. Portraiture of training and testing procedure of neural network for (a) sequential and (b) parallelized backpropagation implementation[4].

The author summaries about multilayer perception neural networks and the backpropagation learning algorithm[5,6] and also described the classical algorithm of batch training.

Backpropagation Learning algorithm starts with the initialization of weights, the number of iterations and fixed desired error value. The next step is the beginning of iteration part. Then errors in the output neurons are calculated and back propagate the errors to hidden neurons. The delta weights are calculated and weights are accumulated to stop the iteration. Later, weights are updated in the network, Means Square Network Error (MSE) is computed and the required conditions have been checked i.e., if the calculated error is greater than the desired error, and then repeat the iteration process until the network converges.

To parallelize the algorithm, the author, divide the training dataset T into equal parts T_1, T_2,……T_n, where 'n' is a number of threads.

From the backpropagation learning algorithm, the iteration part can run in parallel threads and each pattern in the Dataset T_k performs back propagation independently. The corresponding delta weights are accumulated together once all thread has finished its training. At last, algorithm is executed sequentially and the weights are updated and the process is repeated until the network is converged.

The Author has trained the four different problems with different algorithms. The results and simulations are discussed in the paper. Best efficiency and speed up are achieved for the O2 A-Band spectrum simulation problem (740% with pthreads and 743% with OpenMP) which is shown in Figure 2.

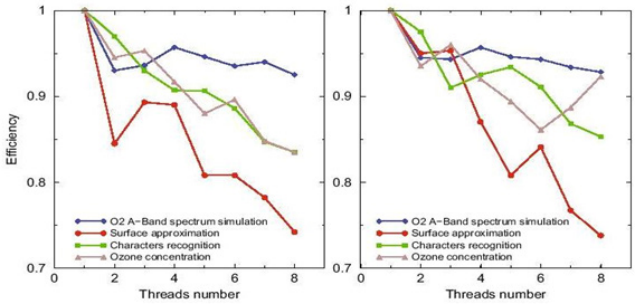

Figure 2. The performance of parallel training of neural network on an8-corecomputerby using thread[4].

Advantages

- The best expedite is achieved by creating as many server threads from available cores.
- In parallel training, OpenMp shows better performance when compare with POSIX threads.

Limitation

Time taken to switching from one thread to another is very high in case of processing more threads than available cores, such that efficiency of parallelization is considerably low.

The authors briefly discussed about the present state of automation in terms of memory and speed[7,8],along with data on some living entity. They discussed Back-Propagation algorithm for multi-layer neural network and attempt to carry out ANNs on enormously parallel computers.

The advanced ANN software, called SPANN (Scalable Parallel Artificial Neural Network)[7], which uses the backpropagation algorithm and runs on enormously parallel computers. The following techniques are used to develop the software:

- The neural network represented using an object-oriented (C++) approach.
- Parallelization is achieved through message passing Interface (MPI) Library.

The author compared the serial training of neural network along with parallel training and the results are presented[7].They have presented the performance results of SPANN on a large parallel computer. The training was

performed on the NASA SGI Columbia computer, Figure 3 shows3-processor (P0, P1 and P2) simulation, which connection involved with 12 inputs, 2 hidden layers with 9 hidden units each and 6 outputs.

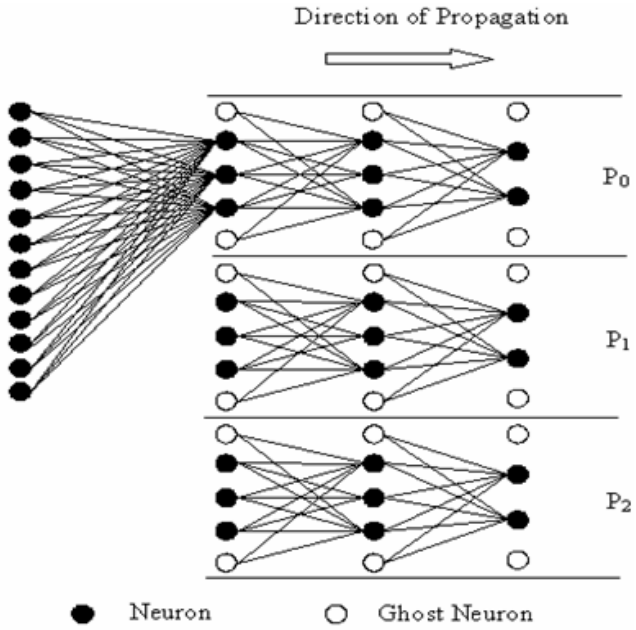

Figure 3. Schematic representation of neuron connection in a two processor system[7].

Table 2 shows the required neurons for the available processors.

Advantages

- SPANN is extensible to train the neurons and permits to use billions of weights.
- It is used on small serial computers and allows an ANN to be trained on an enormously parallel computer.
- In this approach, only value at the boundaries of the processor domains is communicated thus reducing the communication cost significantly.

Limitation

The author concludes the current system is not sufficient for massively parallel processing because it requires an enormous amount of inter-processor communication for the neurons at each level to communicate all neurons on the previous level.

Pattern-Parallel Training(PPT)[9], it is more appropriate for cluster systems. In PPT, the idea is to duplicate full ANN at every cluster node and preferred a subset of patterns are processed from the training set.

The author summarizes parallelizing neural network training in the previous work. This work focused on creating special purpose neural network hardware[10–12]. For an appliance, the author has used MPI[13] function that uses the FANN[14], open neural network training library. They have done some changes in FANN tool to export some of its internal structures.

They have selected the problem called an eight-bit parity pattern and back propagation algorithm to train eight cluster nodes with 1GB Ethernet switch. The author said, during the training process, when patterns are selected randomly from the full training set may not suffer from anomalies[9].

Finally, the author statedthat they have planned to develop fully integrating PPT into FANN library to equip complete library interface to PPT.

Advantage

PPT results in significant speedups over sequential version. The best speed-up of 10.6 was achieved by the eight node version of PPT (65 seconds vs 689).

Limitation

The author found that they do not have access to more nodes. The techniques are essential up to 8 nodes after that network start to memorize not to generalize so they are unable to test 16 and 32 node versions.

The author focuses on parallel implementation[15] by appropriate the recent and available parallel computer

Table 2. Required Training time for the serial ANN[7]

Processors	Inputs	Neurons	Neurons Per Hidden Layer	Weights	Percent Correct	Memory Used(GB)	CPU Time(Sec.)
16	37,500	1584	256	9,613,376	100%	0.08	246
64	150,000	6272	1024	153,652,480	100%	1.20	2489
500	600,000	25,000	4000	2,400,106,384	89%	19.0	6238

software and hardware. From the analysis, he says that standard Backpropagation training algorithm is not sufficient for parallel computing. The author summarizes the topics about communication load and numerical complexity, which is required for experimental analysis.

The key to parallel programming is the exchange or distribution of information between the nodes. The better solution to the problem is to use Message Passing Interface (MPI)[16,17], it is designed to communicate between homogeneous cluster systems.

Finally, the author concludes many neural network applications, shows that all clusters afford only limited speedup.

The author said a parallelization of MLP[18] with the usual sequential BP training algorithm is not extensible due to high synchronization and connection overhead among parallel processors[13]. So the author decided to update the neurons weights and thresholds at the end of each training process.

To develop the parallel algorithm, the author decided to create a Master processor, which executing, assigning functions and calculations to the workers. The computational work is divided among the processors. After the assigned patterns, the nodes are executed and synchronization with other processors is automatically provided[16].

The author used C programming language with the standard MPI functions to develop the software code. The experiments were carried out on SMP supercomputer and Computational Cluster. The two techniques used are OpenMP and MPICH2[19].

Finally, the experimental results show that different internal algorithms of MPI –Allreduce() gave better results for different scenarios of the parallelization problem on parallel systems.

Limitations

- The developed algorithm is not enough to design an optimal parallel algorithm on the computational cluster with distributed architecture.
- The author addresses the two following sufficient conditions to decrease the communication overhead and to develop the efficient parallel algorithm of neural networks,
    - Minimizing the number of MPI collective's function calls.
    - To use the advanced properties of the latest release of appropriate MPI packets related to the improved performance of collective communications.

The authors focused on High-Performance Computing (HPC) system[20] and a hybrid approach[21] that is a combo of two traditional programming modelsMPI and OpenMP. They have described the applications of the hybrid model.

They discussed the performance of Standard Benchmarks from the multi-zone NAS Parallel Benchmarks (NPB-MZ) and they are derived from pseudo-application benchmarks included in the regular NPBs[22,23]. They have considered two pseudo-applications and two real-worldapplications and determined benefits of the hybrid approach for performance and resource usage on three multi-core based parallel systems.

Finally, the authors concluded the hybrid approach for developing load balance and numerical convergence. They have described their approach to extending OpenMP with the conception of improving the performance of resultant programs on ccNUMA systems.

Advantage

- Enhanced load balancing and numerical convergence.

Limitation

There is only finite performance of OpenMP and ambiguous interaction between OpenMP and MPI processes.

The authors focused on HPC[24]:Shared memory nodes with several multi-core CPIs are communicated via a network infrastructure. The authors discussed possible reasons in order to get optimal scalability. They tried to implement the following strategies:

- Reduce Synchronization overhead
- Curtail load imbalance
- Reduce computational overhead and memory utilization.
- Curtail MPI communication overhead

The authors have included an "OpenMP" branch because, beyond the boundaries of a single cluster node, an OpenMP allows the parallelization in "distributed virtual shared memory". Figure 4 shows Taxonomy of Parallel programming prototype on hybrid platforms.

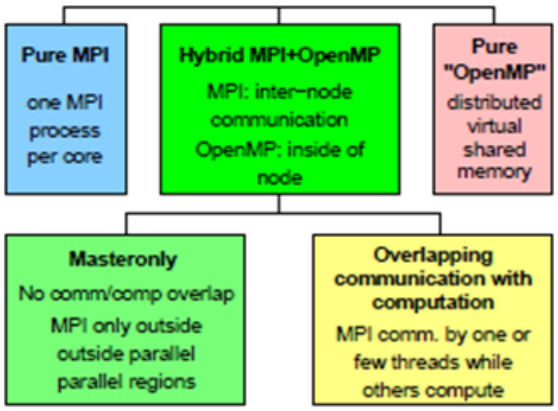

Figure 4. Taxonomy of Parallel programming architecture on hybrid platforms[24].

They summarized about pure MPI and pure OpenMP on clusters[25,26]. They highlight the main issue with getting good performance on hybrid architecture. They have been discussed on Mismatch problems (i.e.) the unsuitability of current hybrid hardware for running highly parallel workload and MPI communication issues during the mapping process.

Limitation

The mismatch problem requires special care, not only with the hybrid programming but also under pure MPI. This model cannot optimize among inter-node, inter-socket and intra-socket communication bandwidths and latencies.

Table 3 shows the summary of parallelization of Neural Network.

S.no	Authors Name	Problem discussed and Technique Used	Advantages	Limitations
1	A.Dobnikar et al.[4]	Parallel Batch Training and Sequential and parallel backpropagation learning algorithm using Pthreads and OpenMP.	The best expedite is achieved by creating as many server threads from available cores and parallelization of training algorithm with OpenMP shows the best result.	Time taken to switching from one thread to another is very high in case of using more threads than cores, such that efficiency of parallelization is considerably low.
2	Lyle N. Long et al.[7]	Identifying Character Set and SPANN [Object oriented (C++) + MPI Library].	SPANN is extensible to train the neurons and permits to use billions of weights. It is used on small serial computers and allows an ANN to be trained on an enormously parallel computer thus reducing the communication cost significantly.	The author concludes the current system is not sufficient for massively parallel processing because it requires an enormous amount of inter-processor communication for the neurons at each level to communicate all neurons on the previous level.
3	G.Dahl et al.[9]	Eight-bit parity problem, Pattern-Parallel Training (PPT) and FANN Tool + MPI Library.	PPT shows best speed-up of 10.6 was achieved by the eight node version (65 seconds vs 689).	The techniques are essential up to 8 nodes after that network start to memorize not to generalize so they are unable to test 16 and 32 node versions.
4	Udo Seiffert, [15]	Survey on Massively Parallel Computer Hardware and Multi-layer Perceptron with Back propagation parallel computer Hardware.	Message Passing Interface (MPI) is a good compromise for parallel computer hardware.	Clusters provide an only limited speedup.
5	V. Turchenko et al.[18]	Parallel Batch Pattern Training and Parallel Programming using OpenMP & MPICH2.	MPI–Allreduce() is one of the internal algorithms gave better results for different scenarios of the parallelization problem.	The developed algorithm is not enough to achieve an optimal parallel process on the computational cluster with distributed architecture.

6	H.Jin et al.[20]	Multi-zone NAS Parallel Benchmark and NAS Parallel Benchmarks (NPB-MZ), MPI and OpenMP.	Extended OpenMP programs on ccNUMA improved load balancing and numerical convergence.	They achieved the only limited performance of OpenMP because there is no well-defined interaction between OpenMP threads and MPI.
7	R. Rabenseifner et at. [24]	Hierarchical hardware design for parallel programming and MPI Communication and OpenMP on cluster systems.	In mapping process, the author addresses the communication issues in MPI.	The mismatch problem needs special care, not only with the hybrid programming but also under pure MPI. This model cannot optimize among inter-node, inter-socket and intra-socket communication bandwidths and latencies.

2. Conclusion and Future Work

We have presented a brief introduction on parallelization of Artificial Neural Network in the cluster and shared memory systems. The techniques MPI and OpenMP have been applied to a number of applications for ANN parallel training, but the parallel execution of neurons at each level is highlighted very less. The software code generation and implementation for parallelization can be considered as a future direction of research.

3. References

1. Deepa SN, Devi BA. A survey on artificial intelligence approaches for medical image classification. Indian Journal of Science and Technology. 2011 Nov; 4(11):1583–95. DOI: 10.17485/ijst/2011/v4i11/30291.
2. Pinkus A. Approximation theory of the MLP model in neural networks. Acta Numerica.1999; 8:143–95.
3. Waqas A, Gilal AR, Bhatti Z, Mahessar AW. Investigating ANNs and Applications. Indian Journal of Automation and Artificial Intelligence. 2013 Feb; 1(2):65–9.
4. Dobnikar A, Lotric U, Ster. Parallel training of artificial neural networks using multithreaded and multicore CPUs adaptive and natural computing algorithms of the series. Lecture Notes in Computer Science; 2011, 6593. p. 70–9.
5. Gallant. Perceptron-based learning algorithms. IEEE Transactions on Neural Networks. 1990 Jun; 1(2):179–91.
6. Rummelhart D, Hinton G, Williams R. Learning internal representations by error propagation. Rumelhart DE, McClelland JL, editors, Parallel Distributed Processing, MIT Press, Cambridge; 1986. p. 318–62.
7. Long LN, Gupta A. Scalable massively parallel artificial neural networks. Journal of Aerospace Computing, Information, and Communication. 2008 Jan; 5(1):68–84.
8. Moravec H. Robot mere machine to transcendent mind, Oxford University Press; 1998 Nov.
9. Dahl G, Mcavinney A, Newhall T. Parallelizating neural network training for cluster systems. Proceedings of the IASTED [Internet]. 2008. Available from: cs.swarthmore.edu.
10. Faerber P, Asanovi K. Parallel neural network training on multi-spert. IEEE 3rd International conference on Algorithms and Architectures for Parallel Processing; 1997.
11. Mache N, Levi P. Parallel neural network training and cross validation on a cray t3e system and application to splice site prediction in human DNA; 1995.
12. Suri NNRR, Decodhare D, Nagabhushan P. Parallel levenberg-marquardt-based neural network training on linux clusters-a case study. The Third Indian Conference on Computer Vision, Graphics, and Image processing; 2002.
13. Lusk E. Programming with MPI on clusters. 3rd IEEE International Conference on Cluster Computing (CLUSTER'01); 2001 Oct.
14. Nissen S. Implementation of a Fast Artificial Neural Network library (FANN); 2003.
15. Seiffert U. Artificial neural networks on massively parallel computer hardware. Neurocomputing. 2004 Mar; 57:135–50.
16. Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. MPI- The Complete Reference, Cambridge, Ma: The MIT Press; 1996.
17. The MPI Forum On-line [Internet]. [Cited 2016 Jan 05]. Available from: http://www.mpi-forum.org.
18. Turchenko V, Grandinetti L, Bosilca G, Dongarra JJ. Improvement of parallelization efficiency of batch pattern BP training algorithm using Open MPI. Procidia Computer Science, ICCS; 2010. p. 525–33.
19. The MPI Tutorial [Internet]. [Cited 2016 Jan 05]. Available from: https://computing.llnl.gov/tutorials/mpi/.
20. Jin H, Jespersen D, Mehrotra P, Biswas R, Huang L, Chapman B. High performance computing using MPI and OpenMP on multi-core parallel systems. Parallel Computing. 2011; 37:562–75.
21. Gharehchopogh FS, Alumini E, Maleki I. A new approach for inter process communication with hybrid of message passing mechanism and event based software architec-

ture. Indian Journal of Science and Technology. 2014 Jun; 7(6):839–47. DOI: 10.17485/ijst/2014/v7i6/46612.
22. MPI: A message-passing interface standard, Version 2.2, Message Passing Interface Forum [Internet]. 2009. [Cited 2016 Jan 03]. Available from: http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf.
23. Bailey D, Harris T, Saphir W, Van der Wijngaart R, Woo A, Yarrow M. The NAS parallel Benchmarks 2.0, Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA; 1995.
24. Rabenseifner R, Hager G, Just G. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. Parallel, Distributed and Network-based Processing [Internet]. 2009. Available from: ieeexplorer.ieee.org.
25. Van der Wijgnaart RF, Jin H. The NAS parallel benchmarks, multi-zone versions. Technical Report NAS-03-010, NASA Ames Research Center, Moffett Field, CA; 2003.
26. Amza C, Cox AL, Dwarkadas S, Keleher P, Lu H, Rajamony R, Yu W, Zwaenepoel W. TreadMarks, shared memory computing on networks of workstations. IEEE Computer. 1996; 29(2):18–28.