A GPU based Parallel Hierarchical Fuzzy ART Clustering

Sejun Kim
Department of Computer Engineering, Missouri University of Science & Technology
skgcf@mst.edu

***Abstract*— Hierarchical clustering is an important and powerful but computationally extensive operation. This motivates the exploration of highly parallel approaches such as is available in Graphics Processing Units, as well as low-complexity algorithms such as Adaptive Resonance Theory (ART). Although ART has been implemented on GPU processors, this is the first hierarchical ART GPU implementation we are aware of. Each ART layer is distributed in the GPU multiprocessors and is trained simultaneously. The experimental results show that for deep trees, the GPU performance advantage is significant.**

I. Introduction

GRAPHICS Processing Unit (GPU) programming, particularly using the NVIDIA CUDA(Compute Unified Device Architecture) has been of interest in computational intelligence, particularly for population based algorithms [6]-[8]. But it would be of great value to use GPU programming to apply the known advantages in hierarchical clustering [1],[2].

Fuzzy Adaptive Resonance Theory (ART) is attractive for hierarchical clustering because of speed, scalability and amenity to parallel implementation. [17]. However, hierarchical fuzzy ART based on GPU engines has not been previously reported. One main constraint in CUDA is the inflexibility of memory inside the kernel meaning that the generation of dynamic arrays are limited only in the host(CPU) side. Typical tree structure algorithms implement pointers for both node creation and reference [9], which is inefficent to do in CUDA programming. The other is that each ART units is trained as data are fed sequentially. GPU implementation which focused in the behavior of a single ART unit was achieved in [10],[18],[19], but hierarchical fuzzy ART needed a different approach. The architecture is inspired from the structure of pipelining [11]. As shown in Fig. 1, even though ART networks are trained sequentially, the parallalization was successfully accomplished.

This paper describes the method of adapting multi-layer tree structure composed of FA units into CUDA platforms. The experiment results are also presented to imply the performance boost on various data sets and parameters compared with those on conventional CPUs. Section II briefly explains FA followed by an overview of CUDA in Section III. Section IV mainly focuses on the proposed algorithm and the experimental data and results appears in Section V. Finally, conclusions and further research tasks are drawn in Section VI.

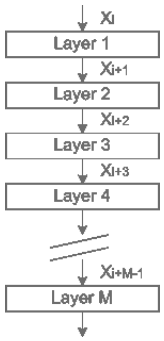


Fig. 1. Data feeding example of CUDA based hierarchical fuzzy ART. The first layer which is also the root node starts with each sample and once the training is finished, the root ART unit passes it to a child node corresponding to which category has won. Each layer loads the proper ART unit for the training for different samples as the winning category varies.

II. Fuzzy Adaptive Resonance Theory and The Hierarchical Fuzzy ART Network

Adaptive Resonance Theory (ART) is an unsupervised learning method which vanquishes the "stability-plasticity dilemma". ART is capable of learning arbitrary data in a both stable and self-organizing manner [4]. ART1 deals with binary data, whereas Fuzzy ART deals with arbitrary data. Henceforth, we will be referring to Fuzzy ART.

Before the training, the data passes through a preprocess step, scaling them to fit in the range of [0,1]. The weight vectors $\mathbf{w}_j$ are initilazed to be all 1. Let $\mathbf{x}$ be an input sample. In category choice, the competition in F_2 is calculated, defined as

$$T_i = \frac{|\mathbf{x} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{1}$$

where $\wedge$ is the fuzzy AND operator defined by

$$(\mathbf{x} \wedge \mathbf{y})_i = min(x_i, y_i), \tag{2}$$

and $\alpha > 0$ is the choice parameter. By the winner-take-all competition,

$$T_J = max\{T_j | \forall j\}. \tag{3}$$

The winning neuron J becomes activated and is fed back to layer F_1 for the vigilance test. If

$$\rho \leq \frac{|\mathbf{x} \wedge \mathbf{w}_J|}{|\mathbf{x}|}, \tag{4}$$

resonance occurs. Then in layer F_2, the input $\mathbf{x}$ is categorized to J and the network is trained by the following learning rule,

$$\mathbf{w}_J(new) = \beta(\mathbf{x} \wedge \mathbf{w}_J(old)) + (1-\beta)\mathbf{w}_J(old), \quad (5)$$

where β ($0 \leq \beta \leq 1$) is the learning rate. If neuron J does not meet the match criterion, it will be reset and excluded during the presentation of the input within the vigilance test. The hierarchical fuzzy ART network is composed of the

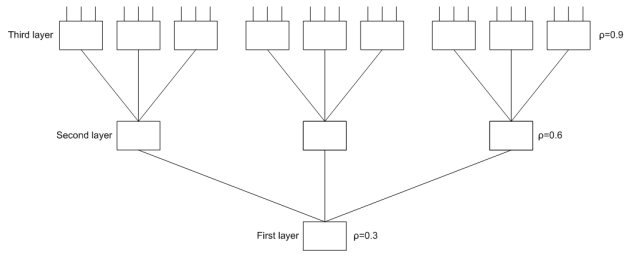


Fig. 2. A hierarchy of ART units. The input pattern is registered at the bottom and is sequentially fed only to those ART units in the hierarchy of "winning" F_2 units from the parent node. (Figure adapted from [20]).

FA[12]. The hierarchy of ART units illustrated in Fig. 2 is done in order to split the clusters more finely by increasing the vigilance. An example of a modular multi-layer network architecture composed of ART networks (HART, for "Hierarchical ART") is in [3].

III. General Purpose Graphics Processing Unit (GPGPU) With CUDA

The desire of displaying a 3D world on computers in real-time greatly increased the computational ability of graphics processors. Fig. 3 illustrates design difference between CPUs and GPUs [13]. A kernel which is the set of operations defined in GPU processors can be programmed and executed simultaneously in different threads. A single NVIDIA Fermi GPU theoretically is capable of containing up to 67,107,840 threads.

Fig. 3. GPUs devotes more transistors to data processing than CPUs.

But several constraints in GPGPU exist. Direct memory access between the host(CPU) and the device(GPU) is not possible. To handle certain data in other sides, data transfer is required either from CPU to GPU or vice versa. Becuase the transfer rate is relatively slow, minimizing data transition is the critical concern. The lack of dynamic pointer and array generation inside the kernel limits the GPU as well.

IV. Parallel Hierarchical Fuzzy ART in CUDA

To achieve the parallalization of the Parallel Hierarchical Fuzzy ART (PHF-ART), the layers, as shown in Fig. 1, were distributed among the GPU threads. Each layer is not an individual module but behaves as a controller to call up required FA on every diverse states. Layer 1 is exclusively assigned to the root FA node. Every time an input passes through a layer, the working FA module in the layer emanates the adapted category back to the layer. Then it assigns the child FA node and broadcasts the node ID and the input ID to the adjacent lower layer while receiving the new assignment from the upper layer, which can be regarded as pipelining. Algorithm 1 is the pseudocode of the kernel in the program.

Algorithm 1 Layer Behavior

```
if L_i assignment exists then
    call FA module
    call input
    do FA training
    set L_{i+1}:FA_J,input
end if
if layer is root then
    idData++
else
    wait assignment
end if
```

Defining the tree structure in CUDA platform was also a critical problem as well as the parallelization. After the initializtion step, the first data will be registered in root FA. Once the training is completed, the layer will attempt to find the ID of the corresponding child FA module which is not set yet. In generic CPU programming, generating a child node can be easily done by allocating a new pointer and cross referring between the parent and child node or by vector template coding. As these methods are impossible in the kernel, a semi-dynamic pointer method is applied. Compared with dynamic arrays, semi-dynamic arrays have a fixed maximum size set and an tracking integer is defined to record the used amount.

The memory size of the graphic card used for the experiment is 1.6 GB. The contents occupying the VRAM within the program are the data sample vectors, layer states and other very small entries such as the kernel itself, registers and local variables in each kernel. A million samples of 4 dimensional float vector take up only 32 MB, implying that the rest of the memory can be declared for the FA modules. The number of maximum FA modules depends on the dimension of the sample vector as well as the preset number of maximum category allowed. Typically in the experiment, 1.5 million FA modules could be pre-declared.

Even though semi dynamic array is applied, a parallel feature known as race condition [14] hinders the tracking of the maximum size. Assuming a certain situation when all of the layers needs to generate a new child FA module,

Algorithm 2 Child ID finder

```
for i = ∀layer do
  if new child needed then
    idChild←tracker
    tracker++
  end if
end for
```

the threads will attempt to assign a child node in the same place as they are running in parallel. Thus, concurrent or sequential coding is required in order to correctly assign a child node and to keep the tracker in control. To reduce the non-parallelism, the throughput of the child id finder which runs right after the FA trainer is limited to as much as possible, which pseudocode is described in Algorithm 2. Once the child node ID is all setup, the layer behavior kernel reruns to finish the task. With the child ID finder, the entire program procedure is depicted in Algorithm 3.

Algorithm 3 Parallel Hierarchical Fuzzy ART

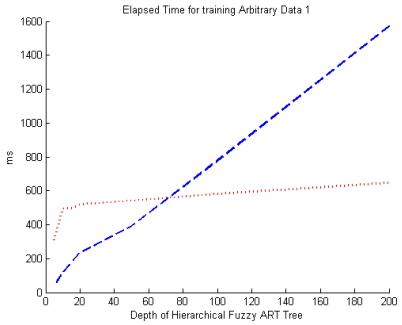
```
init setting
memcpy(host→device)
for i = 1 to nDATA + nLayer − 1 do
  FA_Trainer()
  childIDFinder()
  setNextAssignment()
end for
memcpy(device→host)
```

V. Experimental Results

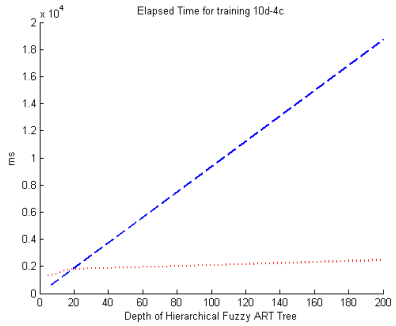
TABLE I
DESCRIPTION OF THE USED DATA

Data Set	Attributes	Number of Data Points
Arbitrary 1	2	800
Arbitrary 2	2	40000
2d-10c	2	3630
2d-40c	2	2563
10d-4c	10	1482
10d-10c	10	3788
10d-40c	10	2707
Abalone	8	4177

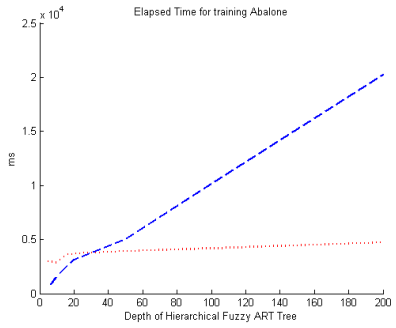
The experiments on both CPU and GPU were held on an Intel Xeon E5620 Quad Core CPU with 12 GB RAM and NVIDIA Geforce GTX 480. 2 sets of arbitrary generated data, "abalone" data from UCI Machine Learning Repository [15] and 5 sets of the synthetic data developed by Handl and Knowles [16] are used for the performance testing. The depths of the hierarchy were set in the range of 5, 10, 15, 20, 50, 100 and 200. For the simulation, only the vigilances of each layer varied linearly in the range of [0.3, 0.9]. The learning rate and the choice parameter were set as 0.8 and 0.1, respectively. The elapsed times on CPU platform and GPU platform were measured differently. The initial setup time for both platforms were excluded but the consumed time while copying data to and from the GPU was included on the GPU performance aspect. The features of the data used for the simulation are summarized in Table I.



(a) Arbitrary Data 1



(b) 10d-4c



(c) Abalone

Fig. 4. The elapsed time as a function of depth of hierarchical fuzzy ART tree. The dotted line is the result acquired from the CPU while the dashed line is that from GPU.

Fig. 4 plots the elapsed time measure on each platform. When the tree depth is low, the CPU running speed is faster as the algorithm was based on layer pipelining. But as the depth grows to meet a certain value, the performance of the GPU implementation exceeds that of the CPU application. The point where the GPU exceeds the CPU varies on each data set as shown in Table II. The time comparison chart implies that the larger the dimension of the data is, the sooner the GPU surpasses the CPU. The maximum speed boost was by 1170% on 2d-10c data with 200 layers. The average performance improvement is 859.37%, 527.95%, 294.74% and 140.46% on 200, 100, 50 and 20 layers, respectively.

TABLE II

ELAPSED TIME (ms) COMPARISON

Data Set	HF-ART Depth						
	5	10	15	20	50	100	200
Arbitrary1(GPU)	312	495	498	521	541	581	647
Arbitrary1(CPU)	40	119	174	234	392	778	1572
Arbitrary2(GPU)	4245	4788	6164	6503	7206	8286	10500
Arbitrary2(GPU)	1895	5879	8792	11672	20227	39093	76597
2d-10c(GPU)	968	628	752	783	853	962	1198
2d-10c(CPU)	349	694	1044	1392	3548	6954	14025
2d-40c(GPU)	478	508	597	617	669	751	927
2d-40c(CPU)	246	493	738	987	2463	4964	9907
10d-4c(GPU)	1342	1441	1750	1807	1924	2097	2462
10d-4c(CPU)	458	921	1379	1850	4647	9340	18719
10d-10c(GPU)	2807	3059	3866	4010	4259	4688	5460
10d-10c(CPU)	1186	2354	3539	4735	11925	23926	43974
10d-40c(GPU)	1980	2213	2784	2891	3038	3323	3834
10d-40c(CPU)	836	1681	2526	3343	8406	16863	31027
Abalone(GPU)	2972	2867	3582	3710	3929	4185	4715
Abalone(CPU)	519	1586	2370	3153	5018	10165	20214

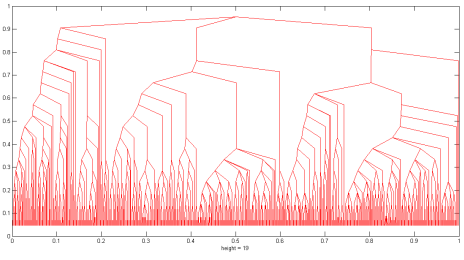


Fig. 5. Generated fuzzy ART module tree through the training.

VI. CONCLUSIONS AND FUTURE WORKS

Fig. 5 illustrates how finely the samples can be fragemented. The results also show that such deep clustering can be accomplished faster than CPU based algorithms. Even though HF-ART on GPU provides a notifiable speed improvement, still a few obstacles remain. The limited size of the graphics memory as well as its inflexibility bounds the total size of FA modules which can be generated. Furthermore high-dimensional data strains the distributed memory limits of the GPU, necessitating investigation of hybridizing this approach with data reduction such as principal component analysis in preprocessing.

To the best of our knowledge, this is the first report of hierarchical ARt clustering in GPU processors. We expect this contribution to have an impact in applications where the need for hierarchical clustering is combined with high data loads and computational demands, such as in data mining and bioinformatics.

REFERENCES

[1] R. Xu and D.C. Wunsch II, *Clustering.* IEEE / Wiley Press, Hoboken, NJ, 2008.

[2] D. Everitt, S. Landau and M. Leese, *Clustering analysis, 4th edition.* Arnold, London, UK, 2001.

[3] G. Bartfai, "An ART-based modular architecture for learning hierarchical clusterings," *Neurocomputing*, vol. 13, pp. 31-45, 1996.

[4] G.A. Carpenter, S. Grossberg and D.B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759-771, 1991.

[5] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, vol. 1: Foundations, ch. 5, pp. 151-193, MIT Press, Cambridge, MA, 1986.

[6] D.M. Chitty, "A data parallel approach to genetic programming using programmable graphics hardware," *GECCO '07: Proceedings of the 9th annual conference on genetic and evolutionary computation* , vol. 2, pp. 1566-1573, Jul. 1991.

[7] Z. Luo, H. Liu and X. Wu, "Artificial neural network computation on graphic process unit," *IJCNN '05. Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 622-626, Jul, 2005.

[8] J.M. Li, D.L. Wan, Z.X. Chi and X.P. Hu, "A parallel particle swarm optimization algorithm based on fine-grained model with GPU-accelerating," *Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology*, vol. 38, no. 12, pp. 2162-2166, Dec. 2006.

[9] D. Knuth, *The Art of Computing Programming: Fundamental Algorithms, 3rd Edition*, vol. 1, Addison-Wesley, 1997.

[10] M. Martnez-Zarzuela, F. Pernas, A. de Pablos, M. Rodrguez, J. Higuera, D. Giralda and D. Ortega, "Adaptative Resonance Theory Fuzzy Networks Parallel Computation Using CUDA," *Bio-Inspired Systems: Computational and Ambient Intelligence*, vol. 5517, pp. 149-156,2009.

[11] A.J. Martin, S.M. Burns, T.K. Lee, D.Borkovic, and P.J. Hazewindus, "The design of an asynchronous microprocessor," *Advanced Res. VLSI: Proc. Decennial Caltech Conf. VLSI: Proc. Decennial Caltech Conf. VLSI*, MIT Press Cambridge, MA, Mar. 1986.

[12] D.C. Wunsch II, "An Optoelectronic Learning Machine," Ph. D. Dissertation, University of Washington, Jul. 1991.

[13] NVIDIA, *NVIDIA CUDA C Programming Guide*, ver. 3.2, NVIDIA Corporation, Santa Clara, CA, Nov. 2010.

[14] R. Netzer and B. Miller, "What are race conditions? Some issues and formalizations," *ACM Letters on Programming Languages and Systems*, vol. 1, no. 1, pp. 74-88, Mar. 1992.

[15] A. Frank and A. Asuncion, *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]*, University of California, School of Information and Computer Science, Irvine, CA, 2010.

[16] J. Handl and J. Knowles, "Improving the scalability of multiobjective clustering," *Proceedings of the Congress on Evolutionary Computation 2005*, vol. 3, pp. 2372-2379, 2005.

[17] D.C. Wunsch II, "ART properties of interest in engineering applications," *Proc. IEEE/INNS International Joint Conference on Neural Networks*, Atlanta, GA, 2009.

[18] M. Gorchetchnikov, H. Ames, M. Versace, "Simulating Biologically Realistic Neural Models on Graphics Process Units," ICCNS 2008, Boston, MA, 2008.

[19] R.J. Meuth, "GPUs surpass computers at repetitive calculations," *Potentials, IEEE*, vol. 26, no. 6, pp. 12-23, Nov.-Dec. 2007.
[20] D.C. Wunsch II, T.P. Caudell, C.D. Capps, R.J. Marks II, R.A. Falk, "An optoelectronic implementation of the adaptive resonance neural network," *Neural Networks, IEEE Transactions*, vol. 4, no. 4, pp. 673-684, Jul. 1993.