The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Neural network application using GPGPU

Y. Tsuchida and M. Yoshioka
Osaka Pref. Univ., Sakai, Osaka 599-8531 Japan
tsuchida@sig.cs.osakafu-u.ac.jp

Abstract

In this paper, we have proposed the speed up method of neural network's running especially in learning time using GPU resource. Implementation of the General-Purpose computing on GPU- GPGPU became easier by the integrated development environment, CUDA distributed by NVIDIA. GPU has dozens or a hundred arithmetic circuit, whose allocations are controlled by CUDA. We propose the neural network implementation using GPGPU. Local minimum, one of the limitation of neural network, is conquered by using some networks with different initial weight coefficient in parallel. On the other hand, the neural network structure was discussed to adapt to parallel processing. One of this implementation is applied to the remote sensing. As a result, this implementation is 20 times faster than CPU.

keywords GPGPU ,CUDA, Nural Network

1 Introduction

Recently, graphic boards have higher performance with development of 3DCG and movie processing than CPU, and widely used with progress of computer entertainment. In this paper, we have proposed that the neural network's running time become speed up especially in learning time using GPU resource. Implementation of the General-Purpose computing on GPU-GPGPU became easier by the integrated development environment, CUDA distributed by NVIDIA. GPU has dozens or a hundred arithmetic circuit, whose allocations are controlled by CUDA. We propose the neural network implementation using GPGPU. Local minimum, one of the limitation of neural network, is conquered by using some networks with different initial weight coefficient in parallel. The performance depends on the allocation of threads, because the CUDA has hierarchical framework to treat many threads. Second, CUDA has many types of memories too, especially, how to treat the shared memory, one of the on-chip memory, influences performances. Therefore the neural network structure was discussed to adapt to parallel processing.

2 CUDA

CUDA, Compute Unified Device Architecture, is a software development kit distributed free by NVIDIA. By employing a supported graphics board, the parallel computing architecture is used easily. In computing using CUDA, the threads are stored in hierarchy streucture. The grid exist as highest hierarchy in GPU, in the grid there are 65,535 x 65,532 blocks, and the threads are managed by three dimensions in the block. Some types of memories are implemented in CUDA, and, among them, the shared memory exists in each blocks. This memory is implemented as on-chip memory, therefore the access to the memory is faster than any other memories. However this memory is used in threads only. This memory is "shared" among threads existing in a block.

3 The parallelization for the whole neural network

At first, one neural network is implemented as one computational thread of parallel computing, and these threads have each different initial weight coefficients.

In the CUDA, it must be configured which hierarchy the threads are assigned to. Therefore, when N networks are executed, the number of thread in block is expressed by m and the number of blocks in grid is N/m, which m and N are integer. The relationship between m and processing time are evaluated firstly.

Table 1 shows the specification of GeForce GTX480 which is used in this research. Test network has 2 neurons in input layer, 3 neurons in hidden layer, and the 1 neuron in output layer. There are 4 lerning pattarns, and the learning is repeated by 2200 steps at each pattern. This routine is employed as one thread, and the time of processing for N threads are measured

Table 1: GeForce GTX480

Number of cores	480
Global Memory	1.5GB
Constant Memory	64KB
Shared Memory	48KB/block
Clock rate	1.40GHz
Maximum threads	1024/block

by changing N and m. Figure refmap shows the measuring result, where $m = N$ represents all threads are assigned in one block. Because of the limited number of arithmetic units, the process time is proportional to N upward $N = 40$ in $m = 1$, and because of the time of memory access, the performance is down in $m = N$ too.

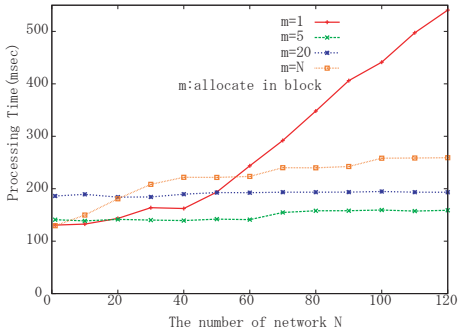

Figure 1: Processing time for whole network thread

One of the goal of the parallelization, the calculation time became flat independent of N, is achieved rid of $m = 1$. From this result, the time is depend on m, however the influence is small.

The problem of m is influenced to the shared memory on implementation, because this memory size uses m times, and this is allocated fixed size in each block. Because shared memory is less than grobal memory, m must be decided by the scale of the network.

4 Parallelization of the inner structure of network

In previous section, one entire network is implemented as one thread, therefore we propose that the network structure is broken up and scattered on GPU. Figure 2 shows the model of the neural network. In the network, some neurons exist in a layer, these neurons can be evaluated in same time.

However, the inputs of the neurons are previous layer's results, therefore the momentary pause is

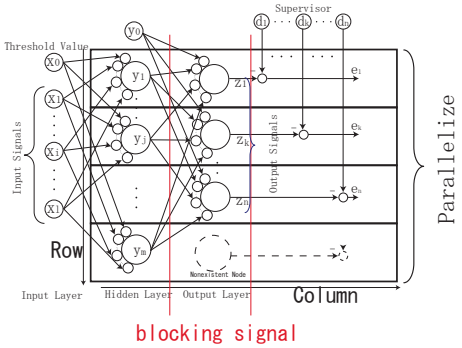

Figure 2: The neural network model

needed until the output from the previous layer become complete. The block signals are employed in front of each layers, which show 'stop' until the all neurons have been calculated completely. It is necessary to calulate one network, The number of threads which is necessary to calculate one network is maximum numbers of neurons per layers, n_{max} . Note that the processing waits for the other when it comes in the layer where the neuron is nonexistent.

The phase of foreword propagation is independent in calculating other patterns too, so p patterns loop can be calculated in parallel. In the phase of the back propagation, more flags are needed to calculate in parallel, these are hoisted to protect the one neuron from other propagation.

The sizes of grids and blocks are expressed by three dimensions, where z-direction of grid is 1. In the experiment, the grid allocate $(N/m, p, 1)$ blocks, and the block allocate $(m, n_{max}, 1)$, and the processing time was measured. The sample network is same as previous section, therefore the $p = 4, n_{max} = 3$. Figure 3 shows the result.

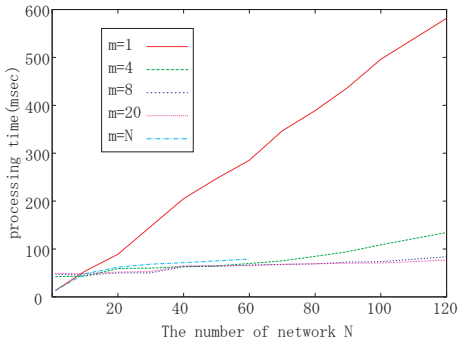

Figure 3: Processing time for scatterd network

Table 2: CPU(host machine specification)

CPU	AMD Phenom 9600
Frequency	2.31GHz
Memory	3.8GByte
Operating System	Windows XP sp2

As a result,the time is proportional to N in early stage when $m = 1$. The tendency of the total plot resembles a previous section, then the influence of an calculation cost by m on hardware is small. The result was speed-up only 2.4 times in comparison with the foregoing chapter. So, the further improvement of the block signal and flags is necessary.

5 Comparison with the CPU

This suggestion was compared with a CPU. Table 2 shows the specification of a comparison CPU.

The experiment is evaluated in processing time by changing the number of hidden layer and pattern.

Test network of this section has 3 neurons in input layer and the 3 neuron in output layer. The learning is repeated by 2200 steps at each pattern. The number of network $N = 40$ and the network in block $m = 4$ is assigned. The number of hidden layer is selected from $3, 6, 9$ and the pattern is chosen from $5, 10, 15, 20$.

Table 3 shows the processing time of the GPU, and table 4 show the ratio of CPU to GPU. The columns is the number of pattern, and the row is the number of hidden layer.

Table 3: Processing time for patterns and hidden layers by GPU

The number of hidden layer	The number of pattern 5	10	15	20
3	65.34	107.92	161.19	246.22
6	69.47	110.72	159.07	249.67
9	111.75	218.38	326.04	436.61
12	112.53	214.67	318.54	436.96

Compared with the CPU, 22.4 times in average is faster. When the number of hidden layer is 3 and 6 , the tiem of the same number of pattern is equivalent. And when the number of hidden layer is 9 and 12, the time is equivalent too.

This result shows that the usage of the block influences the performance.

Table 4: The ratio of CPU to GPU (CPU/GPU)

The number of hidden layer	The number of pattern 5	10	15	20
3	15.33	17.53	17.16	14.79
6	21.57	26.19	27.73	23.39
9	18.06	18.08	18.01	18.14
12	22.37	23.11	23.20	22.43

6 Conclusion

We proposed the speed-up method of neural network learning by using GPGPU. The GPGPU has the hierarchical structure to store the threads, the performance is influenced by assigning hierarchy. At first, the learning time of some networks in parallel was measured to show the effect of the hierarchy.

The neurons of each layer in network are processed in parallel, and patterns are processed in parallel too. This method shows the 2.4 times faster than the first method, and compared with CPU, the processing time is 22.4 times faster.

This method will be applied for home use. The method is applied with the graphics or motion pattern recognition easily, because GPU is the hardware for graphic processing. This goal of this method application is usign for the speech recognition.

In the other hand, this method is used in remote sensing field. The goal in the future is that the new learning algorithm adopted on GPGPU is created.