MRI Image Segmentation Based on a GPU Shortest Path Algorithm

Jie Wang[a], Weihao Chen[b]
School of Software Technology, Dalian University of Technology, Dalian, China
[a]wangjie1003@163.com, [b]endeavour35@163.com

***Abstract*—Dijkstra algorithm can be adopted to distinguish different parts of boundaries in medical image segmentation problem, which can be a reference for further segmentation operation. However, classical Dijkstra algorithm can hardly adapt to real time image segmentation problem owing to its exponentially $O(n^2)$ computing complexity, especially for the increasing number of nodes. In this paper, we designed and implemented a parallel shortest path algorithm accelerated by GPU for medical image segmentation problem. A dynamic relax approach is presented to optimize classical Dijkstra algorithm. Therefore, the new parallel Dijkstra algorithm can be easily applied to CUDA parallel framework without concerning about GPU hardware and CUDA optimize details. Two experiments have been conducted to evaluate the algorithm performance. The results show that our new Dijkstra algorithm can get 8 speedup for 4096 points compared with the classical Dijkstra algorithm. Besides, an impressing result with two speed up for 128*128 points problems is demonstrated in parallel Dijkstra compared with parallel Moore. In conclusion, the new parallel Dijkstra algorithm can significantly improve the real-time performance of image segmentation.**

Keywords- CUDA; shortest path algorithm; medical image segmentation

I. INTRODUCTION

Based on magnetic resonance phenomenon, magnetic resonance imaging (MRI) gets electromagnetic signals from human body, which filters to obtain useful information and rebuild image for each organ of human body. Modern medical image segmentation is a technology which decomposes image into specific and different nature areas, and extracts interested target areas. Medical image segmentation [1] is an important research direction of medical image processing, the basis of measuring lesion region extraction, specific organization and implementation, 3d reconstruction result of medical image segmentation, which has attracted widespread attention.

The shortest path for segmentation is the most commonly used method in image segmentation. Dijkstra algorithm [2] is a classic algorithm in SSSP (Single Source Shortest Paths) problem, which can efficiently calculate the figure of all nodes of the shortest path to the Source point. Medical image segmentation problem based on Dijkstra algorithm can be used to distinguish different adjacent parts of the boundary in the figure, and provide the basis for further segmentation operation. These images can be specified to isolate the area of organs and tissues, so as to help professional medical workers for further analysis. However, the time complexity of Dijkstra algorithm is $O(n^2)$. In practice, computation time will increase exponentially along with the increase of the node size. Parallelizing the existing serial algorithm gradually becomes the hot spot of research, with the growth of computing power and programmability of graphics processor GPU.

The pre-segmentation image is usually abstracted as a connected graph in medical image segmentation problem, whose pixels are abstracted as nodes, edges are abstracted as the connection between adjacent pixels. Due to more nodes, how we find the shortest path between nodes to speed up the search becomes the key of this problem.

The main research of image segmentation problem at home and abroad is how to design more efficient strategy of priority queue or reduce the search space [3, 4]. Such as lv jie [5] etc. select efficient sorting algorithm to improve the speed of path searching. They optimize the process of Dijkstra algorithm by sequencing the cycle of temporary node binary heap sort of shortest path set [6], and reduce the time complexity of searching process to O (nlogn). However, the key of the above algorithms is not comprehensive, which are relatively single and should be further improving the precision of the searching. This algorithm is especially poorly applied in medical image segmentation field. So using a large number of threads in parallel execution of optimal performance based on the existing parallel shortest path algorithm can effectively improve the efficiency of the shortest path algorithm.

II. PARALLEL SHORTEST PATH ALGORITHM

Classic serial SSSP algorithm including the Dijkstra algorithm, Moore algorithm, etc. The study of the parallelization of the classic algorithms can significantly improve the efficiency of searching. The paper [7] proposes parallel Moore SSSP algorithm which based on the storage of source node to the nodes of the array D, queue elements of an array Q, parallelize the process of search and accelerate the algorithm to obtain the ideal results. There are also some other improved parallelization strategies for the optimization of the shortest path algorithm [8-11].

In medical image segmentation problem, the abstract of each node in the undirected graph leads to the small number of sides, narrow weight range and nonnegative integer. So the current image segmentation uses Dijkstra algorithm or the optimization algorithm based on Dijkstra for the calculation of the shortest path.

A. Algorithm of Overall Process

The phase of file data reading is completed in the CPU, data in a file is stored in n * n matrix way, there is also a TAB between each elements. Initializing an array of data

transformation can improve the probability of a cache hit and improve the efficiency of the I/O reading when every time the GPU reads a row.

Kernel function is the parallel part that implements on the GPU, whose primary mission is updating data of nodes, searching nodes and storing functions of the shortest path. Data will be stored by n * n matrix and return to the host side after performing the kernel, the allocated storage space on the GPU will be released at the same time.

B. The Key Variable Initialized

It is necessary to initialize the data and deliver them to the GPU before parallel computing. Data structure directly affects the work process, memory access patterns and the system performance. GPU does not support allocating device memory dynamically and using some dynamic data structures. So the algorithm uses the linear array structure to store the nodes information. Table I lists the types and functions of some key variables used in algorithm implementation process.

TABLE I. KEY VARIABLES AND FUNCTIONS

Name	Type	Function
matrixSize	constant	Number of nodes in figure decide the array size, which defined by the user
BLOCK_NUM THREAD_NUM	constant	User specify the number and size of the blocks and threads
cost	array	Store the adjacency matrix
alldist	array	Store the result, and say the adjacency matrix of the shortest path

Due to directly access the host memory of GPU is impossible, you need to allocate memory on the GPU equipment, and copy the data from GPU memory to CPU memory. We can define the structure to unify the management of the GPU memory.

C. The Kernel Function

The overall design thought of this program is avoiding multiple calls of the kernel function which causes additional threads for I/O. So we should design the kernel function to avoid the correlation between processes and assign disposal tasks to each kernel. After that all the kernel functions will perform at the same time and make one-off copies at the end of the completion of kernels, thus such ability can minimize the cost of fetching.

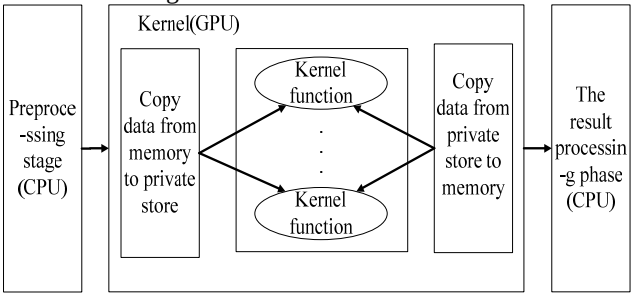

Figure 1. Parallel implementation process function

The concrete implementation of this kernel is closely related to the entire program parallelization strategies. Figure 1 shows the program of parallel implementation process, which includes the figure data inputs, adjacency matrix data for an array of data transformation, data transpose matrix and data from CPU to the GPU global memory copy operation, etc. After completion of copies of data, all the kernel functions will copy data from the global memory to the corresponding thread private store (including block within the shared memory and registers), then all threads will execute concurrently.

After the execution, kernel function will finish the results that copy data from private memory of the threads to the GPU memory. Finally, the program will run over copy function at CPU side to copy the results of GPU memory back to CPU and write data to the output file.

Kernel function algorithm for pseudo code is shown below:

Algorithm 1: The main body of the Kernel

```
1: Initialize( dist,preNode,decide);
2: Relax(dist,preNode,decide);
3: for i=0 to matrixSize do
4:       alldist[source*matrixSize+i]=dist[i];
5: source+=BLOCK_NUM*THREAD_NUM;
```

Algorithm 2: Initialize(dist, preNode, decide**)**

```
1: for i = 0 to matrixSize do
2:     dist[i] = cost[source+i*matrixSize];
3:     preNode[i] = source;
4:     decided[i] = 0;
5:decided[source] = 1;
```

Algorithm 3: Relax (dist, preNode, decide**)**

```
1: for i=0 to matrixSize do
2:         for j=0 to matrixSize do
3:             find the value of the smallest element；
4:             Nd=low;
5:         for w=0 to matrixSize do
6:             if(decided[w]==0&&dist[w]>(dist[Nd]+
7:                     cost[Nd+w*matrixSize]))
8:             dist[w] = dist[Nd] + cost[Nd+w*matrixSize];
9:             preNode[w] = Nd;
10:            end if
```

Through continuous updating and relaxing, algorithm can find out the shortest distance to the first element in one dimensional array, the shortest path value is assigned to all dist matrix after the execution of elements of matrix Size. Finally, all the results will be written back to the text file in the host by the write file function.

III. APPLICATION AND COMPARISON

A. Test Platform and Data Sets

This paper designs the parallel algorithm for the shortest path to accelerate the test environment to run on Windows 7

system, choose NVIDIA GeForce GTX460 GPU. GTX460 computing capacity of 2.1, memory 1 G, bit width 256 - bit memory frequency of 1848.00 MHz, a total of 336 CUDA Cores, 1.40 GHz GPU frequency. The latest GTX460 Fermi architecture allows the maximum number of active threads in a block of 1024, so using the maximum 512 threads in each block for the sake of backward compatibility.

As the process of MRI image abstraction for connected graph is more complex, especially the design of the cost function, which involves professional medical knowledge, this paper tests the data according to document [12] simulation. The input data using adjacency matrix which store in text file, we need to input data that capture into different size.

B. *The Comparision of Serial and Parallel Algorithm*

Serial algorithms and GPU parallel algorithms speedup comparison is shown in figure 2. Because the kernel function of parallel algorithms costs a lot of fetch operations, which increase the additional time consumption of I/O operations. Meanwhile the process of copying the result back to the memory is more time-consuming. Given the cost of using the GPU parallel acceleration is much higher than the pure CPU, so it's not suitable to use GPU parallel calculation of the shortest path when the amount of nodes are small.

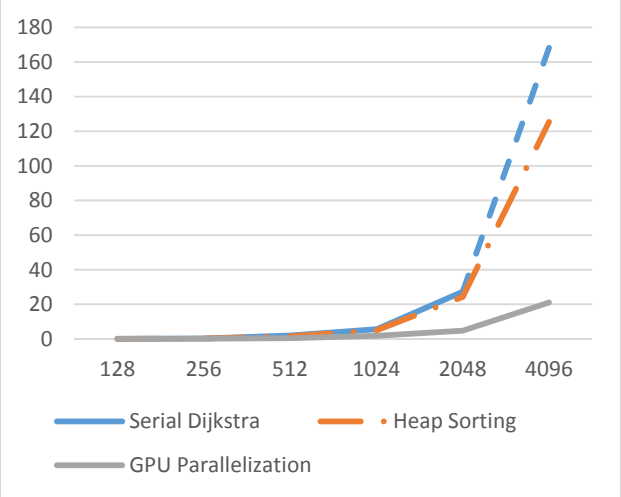

Figure 2. Serial algorithms and GPU parallel algorithms consuming comparison

The performance of parallel algorithm based on GPU is well when junction points reach 256, while the number of nodes is around 1000, the calculation efficiency of the parallel algorithm based on GPU shows nearly three times higher than that of serial algorithm. As the junction points gradually increase, the acceleration of parallel algorithms obtains an increase gradually for 7 ~ 8, and the gap is widening.

C. *The Comparision of Moore and the New Algorithm*

Parallel shortest path algorithm based on GPU has excellent speedup compared with other parallel algorithm. Moore parallelization and Dijkstra parallelization speedup comparison is shown in figure 3. GPU acceleration parallel shortest path algorithm is nearly 1.5 times higher than average GPU Moore algorithm when nodes are under 20000. As nodes continue to increase, the speed of the acceleration of GPU Moore algorithm will increase faster than the proposed algorithm, they will increase gradually close, even GPU Moore algorithm increase slightly more than the algorithm proposed in this paper.

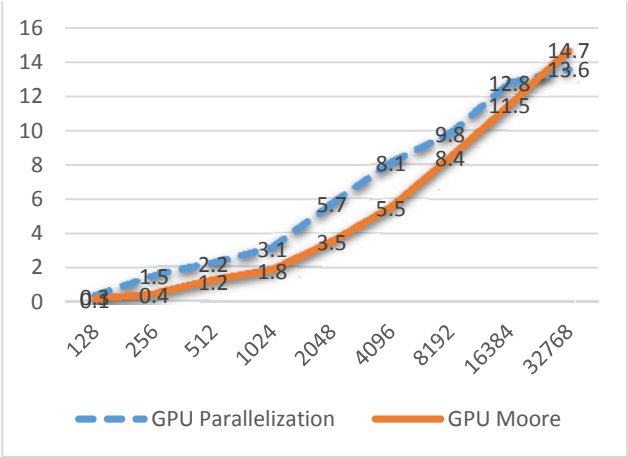

Figure 3. Moore parallelization and Dijkstra parallelization speedup comparison

It can be seen from the analysis of implementation of the process that the algorithm which the paper puts forward can reduce the kernel function calls. Whose computing performance is excellent than GPU Moore algorithm of hierarchical parallel strategy, so the speed is better in the test. When the junction is more than 20000 points, fewer kernel calls has resulted in excessive internal implementation process. Because of operating too much node data, the kernel functions need extra time to access the shared memory, read data you need to own private storage, and increase the running time of the kernel function. However, GPU Moore algorithm of hierarchical kernel calls can ensure the efficient operation of the kernel function in each level task, which can make up the cost of overheads caused by multiple calls to the kernel function.

We can draw the following conclusions from the above experiment that when the node size continues to increase, the GPU Moore algorithm for acceleration ratio will increase at a speed of more than the design algorithm until a threshold in theory. Thus we will choose GPU Moore algorithm for our acceleration when the junction points are more than 20000. The general image for MRI image segmentation problem is 128* 128 pixels which contains 16384 nodes when converted into matrix, so a design of parallel shortest path algorithm of this paper can significantly improve the computational efficiency.

IV. CONCLUSION AND FUTURE WORK

We design and realize a kind of GPU acceleration parallel shortest path algorithm in this paper, which use CUDA framework implementations of NVIDIA and

compatible with NVIDIA whole series of graphics. User needs specify the number of block and thread calls according to the nodes, memory for allocation. The computational algorithm is able to complete work calls automatically. This algorithm use CUDA to block the details of the hardware to simplify the general difficulty of GPU. At the same time, parallel shortest path algorithm based on GPU has a speed ratio about 8 times compares with the traditional serial algorithm, which can improve the efficiency of image segmentation significantly.

There are also some shortcomings in the parallel shortest path algorithm which the paper has discussed. The acceleration effect of this parallel algorithm is worst than other parallel algorithms in dealing with the data node size 20000 and above. Because of the limitation of hardware conditions in theory, parallel shortest path algorithm based on GPU acceleration will not always increase, but will close to a threshold of the optimal speed ratio that the algorithm can achieve and stabilize. Although test speedup of the parallel shortest path algorithm based on GPU in MRI image segmentation problem is good, it does not achieve the optimal speed ratio because of the limitation of problem size. In order to apply the algorithm on more practical problems, we must find the threshold value corresponding to the problem size and improve the related optimization. This will be the focus of future work.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support from National Natural Science Foundation of China (61472100) and the central university Fundamental Research (DUT14QY32).

REFERENCES

[1] Zhang Dong, Zeng Wenhua. An Improved MRI Brain Segmentation Algorithm Based on AntPart [C]. //2nd International Conference on Computer and Automation Engineering. Singapore, Singapore: IEEE, 2010:533-535

[2] E. W. DIJKSTRA. A Note on Two Problems in Connexion with Graphs [M]. Numerische Mathematik ,1959: 269--271

[3] Wang Shaohua, Zhong Ershun, Zhang Xiaohu, et al. The shortest path algorithm analysis technology to accelerate the search space. Geospatial Information [J], 2013, 11(06) : 62-65

[4] Song Qing and Wang Xiaofan. Review the shortest path algorithm to accelerate technology research [J]. Journal of University of Electronic Science and Technology, 2012, 41(2) : 176-184

[5] Lv Jie, Xiong Chunrong. Interactive medical image segmentation algorithm simulation [J]. Computer Simulation, 2010,27(12):262-266

[6] Dang Jianwu, Du Xiaogang, etc. Pairing heap interactive medical image segmentation algorithm [J]. Computer Science, 2009, 36(11): 290-292

[7] Guo Zhaozhong, Wang Wei, Zhou Gang, et al. Single- source shortest path algorithm design and implementation of GPU[J]. Computer Engineering, 2012,38(2) : 42-44

[8] Hector Ortega-Arranz, Yuri Torres. A New GPU-based Approach to the Shortest Path Problem [C]. //Proceedings of HPCS'2013. Helsinki, Finland: IEEE, 2013:505-511

[9] Pawan Harish, P. J. Narayanan. Accelerating Large Graph Algorithms on the GPU Using CUDA[C]. // Proceedings of HiPC'2007. Goa, India: Springer, 2007: 197-208

[10] Micikevicius P. General Parallel Computation on Commodity Graphics Hardware: Case Study with the All-Pairs Shortest Paths Problem[C] // Proceedings of PDPTA'04. Las Vegas, NV, United states: CSREA Press: 2004: 1359-1365

[11] U. Meyer, P. Sanders. Δ-stepping: a parallelizable shortest path algorithm [J]. Journal of Algorithms, 2003. 49(1): 114-152

[12] Moore E F. The Shortest Path Through a Maze[C]// Proceedings of the International Symposium on Theory of Switching. Cambridge, UK: Harvard University Press, 1959: 285-292

