GPU 를 활용한 스캔라인 블록 Gibbs 샘플링 기법의 가속

*Dongmeng Zeng, **김원식, *Yong Yang, *박인규
*인하대학교, **삼성전자
{zdmicc@gmail.com, ultra16@snu.ac.kr, 425656445@qq.com, pik@inha.ac.kr}

Accelerating Scanline Block Gibbs Sampling Method using GPU

*Dongmeng Zeng, **Wonsik Kim, *Yong Yang, *In Kyu Park
*Inha University, **Samsung Electronics

Abstract

A new MCMC method for optimization is presented in this paper, which is called the scanline block Gibbs sampler. Due to its slow convergence speed, traditional Markov chain Monte Carlo (MCMC) is not widely used. In contrast to the conventional MCMC method, it is more convenient to parallelize the scanline block Gibbs sampler. Since The main part of the scanline block Gibbs sampler is to calculate message between each edge, in order to accelerate the calculation of messages passing in scanline sampler, it is parallelized in GPU. It is proved that the implementation on GPU is faster than on CPU based on the experiments on the OpenGM2 benchmark.

1. Introduction

Markov Random Field (MRF) has been widely used in various research areas. Markov Chain Monte Carlo (MCMC) has been used for the inference on the MRF model. Due to its slow convergence, traditional MCMC is not widely used in MRF optimization. Afterwards, lots of MCMC optimization methods have been proposed such as Swendsen-Wang cuts method and the population-based MCMC[4].

In this paper, a method called a scanline block Gibbs sampler is introduced and implemented on GPU. The message passing part is parallelized in order to accelerate the computational speed. It is shown that the GPU implementation would be faster than CPU from the experimental results.

2. Proposed Algorithm

The main idea of the proposed algorithm is to use the scanline block Gibbs sampler for optimization which is quite similar to the conventional approaches. However, the novelty is that message passing part is implemented on GPU rather than CPU. The pipeline of the algorithm is shown in Fig. 1. And the details of block Gibbs sampler can be referred in [2].

Firstly, the 4-neighborhood grid graph model is chosen from the OpenGM2 benchmark [5] as our graph model, then the whole graph is divided into disjoint set, in which each row becomes each block. So let us consider a set of nodes $X = \{x_1, ..., x_n\}$ in a row. Then the distribution of X can be formulated as follows:

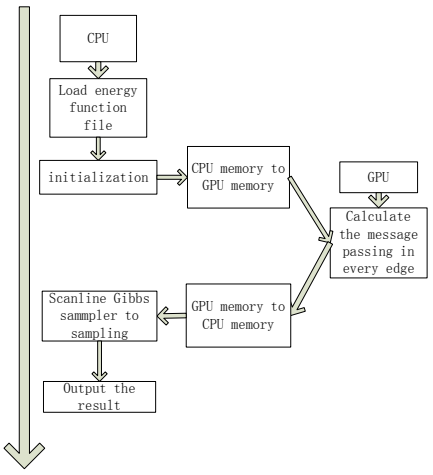

Fig. 1 The pipeline of the proposed algorithm

$$P(X) = \prod_{i=1}^{m} \Phi_i(x_i) \prod_{i=1}^{m-1} \Phi_{i,i+1}(x_i, x_{i+1}) \quad (1)$$

Eq. (2) is referred to [1]. And the message is calculated by Eq. (2).

$$m_{i \to i+1}(x_{i+1}) = \sum_{x_{i=1}}^{L} \Phi_i(x_i) \Phi_{i,i+1}(x_i, x_{i+1}) m_{i-1 \to i}(x_i) \quad (2)$$

Then the conditional distribution for a node i

conditioned on x_{i+1} can be derived by Eq. (3).

$$P(x_i \mid x_{i+1}) \propto \Phi_i(x_i)\Phi_{i,i+1}(x_i, x_{i+1})M_{i-1\to i}(x_i) \quad (3)$$

Once the joint distribution can be decomposed in the form of

$$\begin{aligned} &P(X) \\ &= P(x_1 \mid x_2)P(x_2 \mid x_3)\cdots P(x_{m-1} \mid x_m)P(x_m) \end{aligned} \quad (4)$$

Finally, sampling is available from node x_m to x_1.

In Eq. (2), the CUDA model is utilized to compute the message between each edge. The more details about the CUDA programming could be found in [3]. Due to the maximum threads $T_{\max}$ (=512 on the GTX 580) in a block, so the two dimensional parallelism is been used in Fig. 2. Firstly, assume that the resolution of the image is width*height. Then the number of threads in a block is set to $16*16$. So M (rows) and N (columns) blocks can be defined as $M = width/16$, $N = height/16$. According to the aforementioned, each thread can be corresponding to each pixel. And let assume that the current pixel coordinate is (x, y). According to Eq. (5), the thread can be mapped to pixel position.

$$\begin{cases} x = blockIdx.x * 16 + threadIdx.x \\ y = blockIdx.y * 16 + threadIdx.y \end{cases} \quad (5)$$

where *blockIdx* and *threadIdx* are the built-in variables in CUDA.

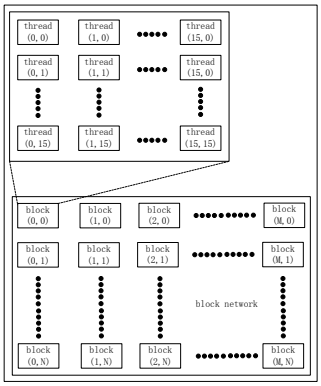

Fig. 2 The thread and block assignment

Before we execute the kernel function, we used the flowing test as our determine condition:

$$if\ (x < width\ \&\&\ y < height) \quad (6)$$

3. Experimental Results

In order to achieve maximum a posterior (MAP) solution, the simulated annealing (SA) scheme is applied together with the proposed method. For the initialization, the winner-takes-scanline (WTS) is chosen as our initialization algorithm, which is an extension of the winner-takes-all (WTA) from a single node to a row of nodes.

The decaying factor c is assigned to 0.995 and a is to 0.5. All the experiments are executed on the 2.67GHz CPU AND 4 GB RAM and GPU is NVIDIA GeForce GTX 580. The penguin-small in color-segment is chosen as our energy function. For more information about the color segment, please refer to [5]. From Fig. 3, we can see both of results in CPU and GPU can find the global optimization and converge. And the speed is also compared in table 1, and the result in GPU is obviously faster than the result in CPU.

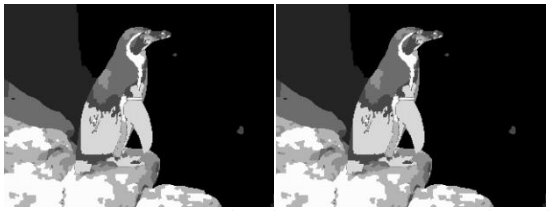

Fig. 3 CPU result (left) and GPU result (right)

Device	Time(s)
CPU	1.104
GPU	0.896

Table1. Execution time comparison (for a single iteration)

4. Conclusion

In this paper, a kind of MCMC method combined with GPU method was proposed. From the experimental results, it is proved that the scanline block Gibbs sampler implemented in GPU is faster than CPU. And the energy function which has a large number of labels will be chosen as our future work.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2012R1A1A2009495).

References

[1] R. Szeliski and R. Zabih, "A comparative study of energy minimization methods for Markov random fields with smoothness-based priors," IEEE TPAMI, vol. 30, no. 6, June 2008.

[2] J. P. Hobert and C. J. Geyer, "Geometric periodicity of Gibbs and block Gibbs samplers for a hierarchical random effects model," Journal of Multivariate Analysis, vol. 67, no. 2, 1998.

[3] NVIDIA CUDA Programming Guide Version 3.0.

[4] W. Kim, J. Park, and K. M. Lee, "Stereo matching using population-based MCMC." IJCV, vol.83, no.2, June 2009.

[5] J. H. Kappes and B. Andres, "A comparative study of modern inference techniques for discrete energy minimization problems," Proc. of CVPR, June 2003.