Parallel Algorithmic Optimization and Achievement for LDPC Encoding and Decoding on CUDA Platform

Liu Zhen
School of Computer Science and Engineering
Jiangsu University of Science and Technology
Zhenjiang , China
ecsilz@qq.com

Wang Yunpei, Lu Lulu
School of Computer Science and Engineering
Jiangsu University of Science and Technology
Zhenjiang , China
1214790021@qq.com; 2514228568@qq.com

(Regular Research Papers)

***Abstract*—With the development of mobile Internet applications, the fourth generation mobile communication (4G) has been widely used, low-density parity-check (LDPC) codes have gradually become the first choice for 4G communication because of its superior performance. For the short of traditional encoding and belief propagation (BP) decoding algorithm, this paper adopts a method of parallel processing for LDPC encoding and decoding which is finally realized on the mobile terminal. On the CUDA platform, CPU scheduling and GPU parallelization are used to process a large number of repeated operations so that parallel encoding algorithm and heterogeneous parallel BP algorithm are achieved and the efficiency is significantly improved.**

Keywords—LDPC codes; encoding algorithm; BP algorithm; parallelization

I. INTRODUCTION

With the rapid development of mobile Internet applications, improving the communication system's reliability is particularly important. For the sake of safety and efficiency in the process of information transmission, it needs through the channel encoding and decoding to achieve the purpose of eliminating or reducing the error probability. The low-density parity-check (LDPC) codes are a kind of linear block codes, its performance is very close to the Shannon limit, it was put forward by Dr. Gallager in 1962, after a long development, people finally realized the superior performance of LDPC [1]. In general, there is a exponential relationship between the decoding complexity of the linear block code and code length, when code length increases to a certain degree, the increase of complexity will be uncontrollable [2]. LDPC codes use the sparse matrix for error detection and correction, which reduces the computational complexity.

In the practical application, the traditional encoding algorithm of LDPC codes have higher complexity, in this paper, using the parallel method for block processing of data sets can improve the encoding efficiency.

The decoding algorithm of LDPC codes is a kind of message passing algorithm sets which based on encoding bipartite graph structure, and in which belief propagation(BP) algorithm has the best performance and the highest complexity. In BP algorithm [4], the messages passing between variable nodes or check nodes are independent, the decoding efficiency can be improved by parallelization.

This paper uses the CUDA platform [6] to configure the GPU processing chip in the mobile terminal communication system to realize the parallelization of the traditional encoding algorithm and BP decoding algorithm, so as to reduce the complexity of the encoding and decoding algorithm, and improve the efficiency of encoding and decoding. CUDA (Compute Unified Device Architecture) platform is a new kind of parallel computing platform that was launched by NVIDIA company, which uses the GPU threads to parallel solve complex problems that consume a long time, it does not need to build a large-scale cluster, and saves project costs. Currently, the latest application of CUDA platform is mobile terminal processing chip (Tegra K1, TK1), this chip integrates a 4-core CPU and a 192-core GPU. In the encoding and decoding algorithm, the target information code word is divided into blocks and assigned to multiple GPU threads for parallel processing by the coordination of CPU. This paper is based on TK1 to implement heterogeneous parallelization of the encoding and decoding algorithm.

II. ENCODING AND PARALLEL ALGORITHM

A. Traditional Encoding Algorithm of LDPC Codes

LDPC codes are actually a kind of linear block codes, which are usually represented by a sparse check matrix H, in which the ratio of the number of non-zero elements that are contained in each row or column to its relative number of rows or columns is very small.

Definition 1: For a linear block code whose code length is n and the number of information bits is k, it can be defined by using a generator matrix G_{k*n} , and information sequence S_{1*k} is mapped to the code word $x = s * G$ through generator matrix G. Linear block code can also be equivalently described by a parity check matrix, all code words are satisfied.

$$x * H^T = 0 \qquad (1)$$

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.189

The traditional encoding algorithm of LDPC codes is generally divided into two steps, the first is to seek the parity check matrix, then is the encoding algorithm which based on check matrix.

Specific encoding algorithm is as follows:

Step1. Using Gallager construction method to generate the check matrix H. First initialize n, j, k (n represents the code length, j, k denote the row weight and the column weight of check matrix) these three variables' values according to the rules of (n, j, k) to construct LDPC codes. According to the line, the check matrix is divided into submatrixs with the same size, and the number of submatrixs is j, each column has one "1" and each line has k "1" in every submatrix; Constructing the first submatrix, and the rest of submatrixs can be obtained by permutation of the ranks of the first submatrix, then through vertical alignment of submatrix to get matrix H.

Step2. According to the definition 1 to find out the generator matrix G.

Step3. Finding out code word information X. According to the formula $x = s * G$ (s represents uncoded information) to get code word information X.

Traditional encoding needs obtain generator matrix G according to the parity check matrix H, then using the generator matrix G to encode the information code word s, its computational complexity is $O(n^2)$. Because of the independence between the uncoded information, it can be divided into blocks, and the parallel processing method can be used to improve the encoding efficiency.

B. Parallelization of Encoding Algorithm on LDPC Codes

The uncoded information code word can be divided into blocks and assigned to multiple GPU simultaneously by using the features of parallel processing data in order to achieve the purpose of improving encoding efficiency. Specific parallel process of encoding algorithm is as follows:

Step1. Using Gallager construction method to generate the check matrix H.

Step2. According to the check matrix H to find out the generator matrix G.

Step3. According to the generator matrix to split uncoded information M. Setting the generator matrix G is a q*s order matrix, the M is splitted into unit row vector with q columns $(m_0,m_1,\ldots,m_{q-1})$. And copy these vectors from the host memory to the GPU global memory.

Step4. Calling q GPU threads, each thread is calculated independently $c_k=m_k*G$ (k=0,1,2,……,q-1), thus getting corresponding code word information.

Step5. The results will be merged to obtain the code word $C=(c_0,c_1,\ldots,c_{q-1})$, and copy it from GPU memory to host memory.

III. DECODING ALGORITHM AND OPTIMIZATION

A. BP Decoding Algorithm of LDPC Codes

Definition 2: For the check matrix H, each row represents a check node and corresponds to a check equation; each column represents a variable node and corresponds to a code variable. Therefore, the relationship between rows and columns of check matrix represents the relationship between the code variables and check equations, which is usually described more intuitively with Tanner graph. Check matrix H in Fig. 1 corresponds to Tanner graph that is shown in Fig. 2.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 1 check matrix H

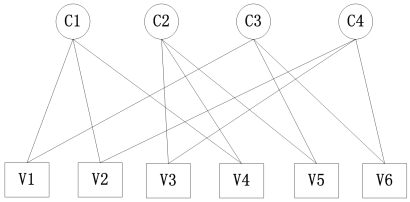


Fig. 2 check matrix H corresponding to Tanner graph

BP decoding algorithm is a kind of message passing algorithms, and has a wide range of applications in communication, artificial intelligence and other fields. In the BP algorithm, the information which transmits among the nodes is the probability of a specific value taken by nodes. For example, the probability of a variable node V that takes a value is passed to the check node C, or the check node C takes a certain probability to pass to the variable node V. BP algorithm is an iterative decoding algorithm, in each iteration, it calculates the probability of all variable nodes or check nodes. In the process of the ith iteration, the data information transmitted from variable nodes to check nodes is derived from the data values transmitted from check nodes to variable nodes in the i-1th iteration, and the current check code is not included in the collection of check nodes. BP algorithm is described below, and schematic graph is shown in Fig. 3.

Step1. Setting x_j as the information sent by the variable node j, y_j is its observation. Variable node calculates priori probability of x_j and sends it to its associated check node, that uses $q_{ji}(1)$ and $q_{ji}(0)$ to represent priori probability of 1 or 0.

Step2. Check node i calculates the posterior probability of x_j according to (2) and (3) and sends it to the associated variable node, that uses $r_{ij}(1)$ and $r_{ij}(0)$ to represent posterior probability of 1 or 0.

$$r_{ij}(0) = \frac{1}{2}[1 + \prod_{\substack{H(i,k)=1 \\ k\in(0,N) \\ k\neq j}} (1 - 2q_{ki}(1)] \qquad (2)$$

$$r_{ij}(1) = 1 - r_{ij}(0) \qquad (3)$$

Step3. According to (4) and (5) to update the probability of variable node j, $q_{ji}(1)$ and $q_{ji}(0)$. Here K_{ji} is a constant to make $q_{ji}(1)+q_{ji}(0)=1$.

$$q_{ji}(0)=K_{ji}(1-P_j)\prod_{\substack{H(l,j)=1\\ l\in(0,M)\\ l\neq i}}(r_{lj}(0)) \qquad (4)$$

$$q_{ji}(1)=K_{ji}P_j\prod_{\substack{H(l,j)=1\\ l\in(0,M)\\ l\neq i}}(r_{lj}(1)) \qquad (5)$$

Step4. The estimated value $\hat{x}_j$ of variable node j can be obtained according to (6).

$$\hat{x}_j=\begin{cases}1, Q_j(1)>Q_j(0)\\ 0, Q_j(1)\le Q_j(0)\end{cases} \qquad (6)$$

The definition of $Q_j(b)$ is as follows, K_j is a constant to make $Q_j(1)+Q_j(0)=1$.

$$Q_j(0)=K_j(1-P_j)\prod_{\substack{H(i,j)=1\\ i\in(0,M)}}r_{ij}(0) \qquad (7)$$

$$Q_j(1)=K_jP_j\prod_{\substack{H(i,j)=1\\ i\in(0,M)}}r_{ij}(1) \qquad (8)$$

Step5. Finally, the estimated values of all the variable nodes are recorded as $\hat{x}=\{\hat{x}_1,\hat{x}_2,\ldots,\hat{x}_N\}$, taking $\hat{x}_j$ into (1), if it is established, then the algorithm will terminate. Otherwise executing Step2 until the maximum number of iterations is reached.

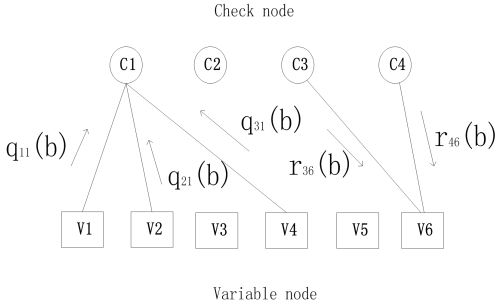


Fig. 3 message passing schematic

From the figure above, the posterior probability of check node C1 is only related to the prior probability of V1, V2, V3 these three variable nodes, and has nothing to do with the rest of the probability of check nodes, therefore, the multiple check nodes are independent of each other, and the probability can be calculated by parallel computing; Similarly, the probability of variable node V6 is only related to the probability of C3,C4 these two check nodes, so the multiple variable nodes are independent of each other, and the probability can be calculated by parallel computing. Algorithm in each iteration is to update all check nodes' probabilities by using variable nodes' probabilities first, then update all variable nodes' probabilities by using check nodes' probabilities. Obviously, iteration does not affect the parallel execution of algorithm.

B. Parallelization of BP Decoding Algorithm

In order to achieve the purpose of parallelization, each variable node or check node is assigned to a GPU thread through CPU scheduling, each thread performs the BP decoding algorithm in parallel, and independently determines whether the decoding was successful or not. Parallel algorithm is as follows, schematic is shown in Fig. 4.

Step1. Copying the data which GPU requires from the host memory to the GPU global memory, so that all threads can share data. These data include observations and code word information of variable nodes and check nodes.

Step2. Initializing the priori probability of variable node $q_{ji}(b)$ by using GPU in parallel. Parallel implementation is shown in Fig. 4 (b). Each variable node is assigned a thread, the priori probabilities of all the variable nodes can be calculated in parallel according to the observations and code word information.

Step3. The estimated value of code word $\hat{\mathbf{x}}$ is brought into the (1). If it meets (1), then it will execute Step6, otherwise it will execute Step4.

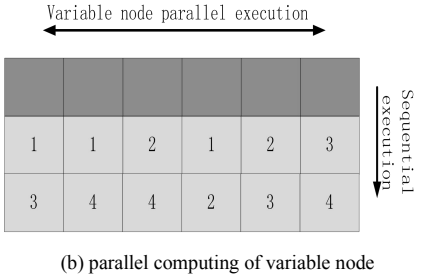
Step4. The probability of variable node $q_{ji}(b)$ is brought into (2) and (3), following the process of Fig. 4 (c) to calculate the probability of check node $r_{ij}(b)$ with parallel computing. Using (4) and (5) to update the probability of variable node $q_{ji}(b)$.

Step5. Judging whether it has reached the maximum number of iterations or not, if not, then according to (6), (7), (8) to calculate the estimated value of code $\hat{\mathbf{x}}$, and execute Step3. Otherwise the algorithm will finish.

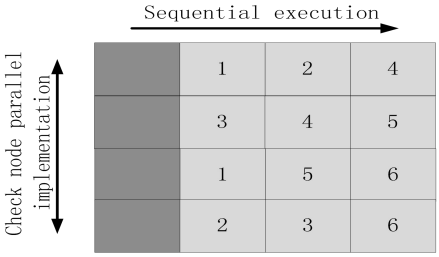
Step6. Outputting code word. Copying the code word from the GPU global memory to the host memory.

1	1	0	1	0	0
0	0	1	1	1	0
1	0	0	0	1	1
0	1	1	0	0	1

(a) check matrix



(b) parallel computing of variable node



(c) parallel computing of check node
Fig. 4 schematic of parallel BP algorithm

IV. Analysis of Performance

The computer which is used in this experiment is equipped with a mobile terminal processing chip (Tegra K1). Experiment 1 tests the impact of data blocks on the time consuming of encoding algorithm, the data are divided into blocks, and its total code word length is 153600, the comparison of time consuming results with traditional encoding algorithm and parallel encoding algorithm is shown in Fig. 5.

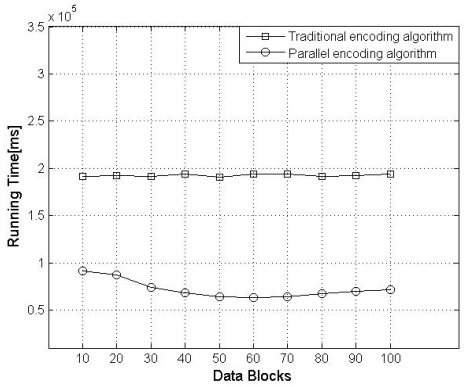


Fig. 5 time consuming of traditional encoding algorithm and parallel encoding algorithm

In experiment 1, the time consuming of parallel encoding algorithm is significantly lower than that of traditional encoding algorithm, and with the increase of data blocks number, the time consuming of parallel decoding algorithm decreases first and then increases. This is because when the blocks reach a certain number, the time consuming of data blocks and integration has exceeded the time saving by parallelization.

Experiment 2 tests the impacts of code length, parallel threads on the time consuming of BP decoding algorithm, code word blocks are used for 10, 50, and 100 data (the code word total length is 153600, 768000, 1536000) to test serial and parallel multithreadeds. The comparison of time consuming with serial BP decoding algorithm and parallel BP decoding algorithm with 16 threads at the same code word block is shown in Fig. 6. The effect of the number of threads on parallel BP decoding algorithm is shown in Fig. 7.

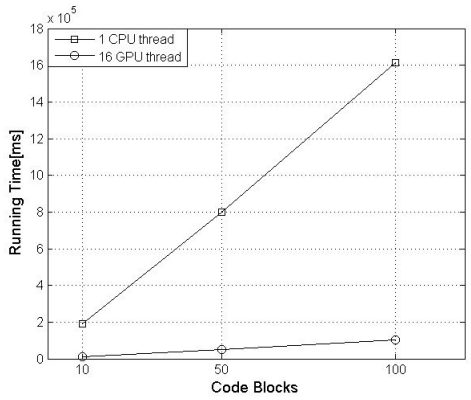


Fig. 6. the comparison of time consuming

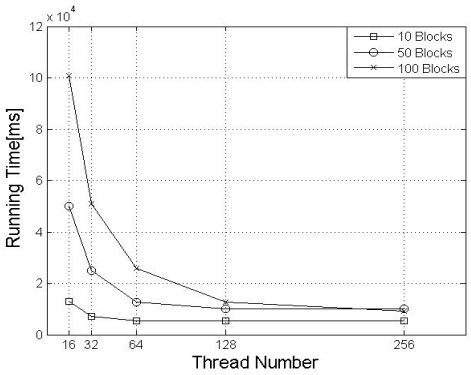


Fig. 7. the relationship between the number of threads and the time consuming of parallel BP decoding algorithm

In experiment 2, the performance of parallel BP decoding algorithm is better than that of serial BP decoding algorithm, and with the increase of threads, the algorithm takes less and less time. But after increasing to a certain number of threads, the improved efficiency is not obvious, this is because the data are close to the maximum degree of parallelization, at this time, allocating more threads will not reduce the time consuming of algorithm. When there are a large amount of data, the parallel degree of algorithm can be further improved by opening more threads. For example, when the number of threads increase from 64 to 128, the performance of parallel BP algorithm has almost no promotion when the code length is 10 code word blocks, the performance can improve about 20% when the code length is 50 code word blocks, and the performance can improve about 50% when the code length is 100 code word blocks.

Conclusion

This paper uses LDPC codes which have the characteristic of relatively low computing complexity in the linear block code to put forward the idea of parallel processing which is based on the traditional encoding algorithm and BP decoding algorithm. Data are divided into blocks and assigned to multiple threads in parallel processing, so as to achieve the purpose of completing parallel encoding algorithm and improving efficiency of information transmission. Parallel BP

decoding algorithm assigns a thread for each check node or variable node, each thread calculates independently and judges whether the decoding is successful or not, so decoding efficiency can be improved. Using CUDA platform, the parallel processing is simplified, and the effectiveness of this paper's method is verified by experiments.

REFERENCES

[1] Yu Yongsheng, Lu Peizhong. On Decoding Implementation of Regular LDPC codes Based on CUDA［J］.Computer Applications and Software, 2010,18(09): 230-232+266.

[2] Yu Yongsheng. Achievement and Research of Parallel Decoding of LDPC codes Based on CUDA Platform[D]. Shanghai : Fudan University, 2009.

[3] Cui Junlong. Study on the Construction of LDPC codes and Decoding Algorithm[D]. Xi'an：Xidian University, 2012.

[4] Huang Haiyi. The Improvement on Decoding Algorithm of Low-Density Parity-Check (LDPC) Codes[D]. Wuhan : South China University of Technology, 2013.

[5] Wang Xiaodan. Performance Analysis and Algorithmic Implementation of LDPC Encoding[D]. Harbin：Harbin Engineering University, 2007.

[6] Joo-Yul Park, Ki-Seok Chung. Parallel LDPC decoding using CUDA and OpenMP［J］. EURASIP Journal on Wireless Communications and Networking, 2011,16(07): 210-214.

[7] Moritz Beermann, Enrique Monzó, Laurent Schmalen. GPU Accelerated Belief Propagation Decoding of Non-Binary LDPC codess with Parallel and Sequential Scheduling［J］. Journal of Signal Processing Systems, 2015,33(02): 394-399.

[8] Zhenbang Wang, Zhenyong Wang. Cross-layer design of LT codes and LDPC codess for satellite multimedia broadcast/multicast services［J］. Chinese Journal of Aeronautics, 2013,22(03): 265-268.

[9] Beomkyu Shin, Hosung Park. Quasi-cyclic LDPC codess using overlapping matrices and their layered decoders［J］. International Journal of Electronics and Communications, 2013,28(09): 411-414.