Parallel Optimization computing in Universal Combinatorial Coding

Tingting Gao[1,2]
[1]Key Laboratory of Database and Parallel Computing, Heilongjiang Province, Harbin 150080, China
[2]College of Computer Science and Technology, East University of Heilongjiang
Harbin, China
15695582@qq.com

Jun Lu*[1,2]
[1]Key Laboratory of Database and Parallel Computing, Heilongjiang Province, Harbin 150080, China
[2]College of Computer Science and Technology, Heilongjiang University, Heilongjiang Province
Harbin, China
Corresponding author: lujun111_lily@sina.com

Abstract— Universal combinatorial coding derives from the combination principle and its core is ordinal calculation. The solution of ordinal is a process of computing permutation and combination values. But permutation and combination computing is very time-consuming. As a result, adopting optimized methods or parallel techniques to increase ordinal calculation efficiency is very necessary. Parallel optimization algorithms of the ordinal in universal combinatorial coding are advanced based on CPU multi-core and multi-thread techniques in this paper, including prime splitting and unequal parallel algorithms. Firstly, the thesis simplifies the multiplication and division calculations of the ordinal parallel computing through prime splitting optimization. Secondly, the thesis analyzes the deficiency of equal parallel that the unbalanced computing time of each part will lead to inefficient parallel computing of the ordinal, and then proposes the unequal parallel algorithm to shorten the ordinal number calculation. The experiment results of prime splitting and unequal parallel algorithm proved that two parallel optimizations are better in improving the efficiency of ordinal computing.

Keywords- Universal combinatorial coding; Ordinal; Parallel; Optimization; prime splitting

I. INTRODUCTION

Data coding technique is a basic technique in contemporary information society. There have been some basic coding methods, such as Arithmetic coding, dictionary encoding and so on [1-2]. The universal combinatorial coding is very special. It is reversible and lossless coding. According to the combination principles, this coding method takes advantage of the relationship between the sequence space and the corresponding ordinal space. It is independent of probability and statistic features of information source. Furthermore, it has many other coding features. For example, it has attributes of arithmetic coding, dictionary encoding and tree coding simultaneously [3]. It has multiple application such as data re-compression, data encryption/decryption.

The core of universal combinatorial coding is ordinal calculation. Because universal combinatorial coding derives from the combination principle, the combination principle involved with huge calculations, and the solution of ordinal is a process of computing values of permutation and combination. As a result, adopting the method of optimized or parallel techniques to increase ordinal calculation efficiency is very necessary. According to the property of universal combinatorial coding [4], the space optimization [5-6] and the speed optimization [7-11] can be done.

The speed optimization technique is the transferring of ordinal combined algorithms into proportion algorithm, with equal parallel technique, which has promoted the efficiency of ordinal calculation. Equal parallel technology is taken advantage of multi-thread technique to calculate the corresponding ordinal of the sequence with the length of L. It needs to divide the sequence into several districts with equal length and the length of each district can be adjusted reasonably. So each district could begin with the non first element of benchmark sequence and end with the first element of benchmark sequence (there might be many sequential elements like this). Each thread calculates the number of permutation and combination of one district, and all threads execute separately in parallel, finally the values of permutation and combination of all districts are added together as a result of the ordinal number. The following section will introduce the method of sequence equal parallel algorithm, and aim at its shortage, the unequal parallel algorithm is advanced. Furthermore, the efficiency of ordinal algorithms is promoted.

II. EQUAL PARALLEL OF THE ORDINAL IN UNIVERSAL COMBINATORIAL CODING

Equal parallel method can be adopted to computing the ordinal in universal combinatorial coding. This method is that the sequence which length is L (L = 256 k, for example) is divided equally and processed parallel in multi-core computer. It uses mainly multithread technology. Rough left and right boundary can be got after the sequence is divided equally, and then the length of each region is adjusted to make each region begin with the non first element of the benchmark sequence and end with

the first element of the benchmark sequence. So the new left and right boundaries of each region are confirmed and then the combination value of each region is computed parallel. Due to parallel computing, the main thread which calculates the first region must judge circularly whether other threads is over. If the calculation is finished, the combination value of the corresponding thread will be added to the sum and the ordinal of the sequence is got.

For example, to a sequence which length L=256k, first of all, equidistance partition can be done according to the computing kernel number M of the computer. The length of each region is n=L/M. It is shown as Fig .1.

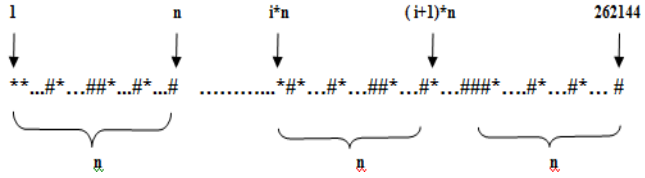

Figure 1. The initial segmentation

Next, the length of each region is reasonably adjusted that each region begins with the non first element of the benchmark sequence and ends with the first element of the benchmark sequence (there may be multiple consecutive such elements, until the last one). So the new left and right boundaries of the region are determined. It is shown in Fig .2 after adjustment.

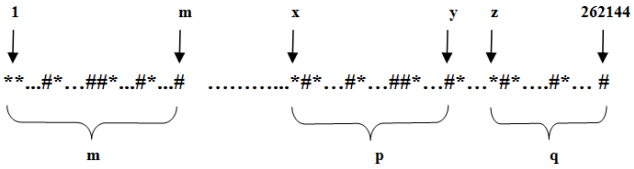

Figure 2. Regional adjustment

In Fig .2, „*" indicates non-first element of benchmark sequence, „#" indicates the first element of benchmark sequence. After the old points are adjusted, the new region begin with „*" and end with „#". Each adjusted region is divided according to the first element of benchmark sequence, so the combination value of each region can be computed parallel. For multi-core computer, each thread calculates the combination value of one region parallel. All combination values will be added at last.

The length of the tested data is 8k, 16k, 32k, 64k, 128k and 256k respectively. It can be shown in Fig .3. Horizontal axis is length(unit is k), vertical axis is time(unit is minute).

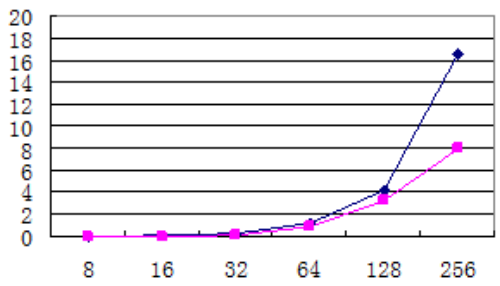

Figure 3. Ordinal compared time between the serial and the parallel

As Fig .3 shows, ordinal computing efficiency of equal diversion parallel method is greater than serial operation efficiency. In fact, it can be further optimized ordinal parallel computing process. This thesis mainly optimizes the process of ordinal parallel computing from two aspects: one is the prime splitting in ordinal parallel computing, the second is the unequal parallel processing to the sequence.

III. PRIME SPLITTING IN ORDINAL PARALLEL COMPUTING

In the process of calculating corresponding ordinal, lots of multiplication /division operations of large numbers are involved alternately and a lot of time is consumed. Prime decomposition technique can be introduced. In the process of ordinal calculation, prime splitting method can be done to the operation data. It makes the large data transform ed from multiplication and division to addition and subtraction, thus the computational complexity is reduced greatly.

A. Prime splitting

The main principle of prime splitting is that multipliers and divisors in the ordinal calculation process are decomposed (decomposed results have been all put into file in advance. When the ordinal is computed, the result can be obtained by looking up the table whose data is from file). Multipliers or divisors are translated into numerator or denominator and then translated into the product of prime power. At last, exponents with the same prime can be offset and the same result can be obtained after simplification. This method can reduce a lot of multiplication and division operation consumed time and the operation efficiency is improved.

According to the principle of prime splitting, the operation data will need to be split. In this experiment, sequence length is 256k (262144) bytes, which the biggest splitting data does not exceed 262144. The splitting results of all data (2 ~ 262144) is put into a file. When the ordinal is computed, the file contents are read into memory array.

According to the minimum product of prime, each split data can be split into six different prime products. That is to say, for maximum data 262144, the smallest product of six different primes is the closest to this value: 2*3*5*7*11*13<262144. So it can create a two dimensional array FenJieSection_Long[262145][6]. Each array element has two components, one is the prime, the other is the corresponding exponent of the prime. It can quickly find the split result of the data by looking for an array subscript. For example, 300 can be divided into 22*31*52, that is to say, the value of FenJieSection_Long[300][0] is {2,2}, the value of FenJieSection_Long[300][1] is {3,1}, the value of FenJieSection_Long[300][2] is {5,2}, the rest value is still the initial value {0,0}.

B. Prime splitting optimization test

Equal parallel computing with prime split and equal parallel computing with non-prime split are compared in this section. In order to better comparison, it takes average measure for many times. For example, sequence length is 256k and the results can be shown in Fig .1. Operation time unit is minute. At last, the time is the time of the thread that consumes the longest time.

TABLE. I COMPARISON BETWEEN SPLITTING AND NON- SPLITTING

Thread number	non-prime splitting (minute)	prime splitting (minute)
thread0	7.93	6.59
thread1	6.41	5.43
thread2	5.31	4.36
thread3	4.01	3.07
globe time	7.93	6.59

Table.1 shows that equal parallel computing efficiency with prime split optimization is obviously better than the equal parallel efficiency without prime split optimization.

In order to get better test results, test and comparison is done to the different length data between prime decomposing and non prime decomposing in equal parallel computing. Test data length respectively is 8k, 16k, 32k, 64k, 128k and 256k. As shown in Fig .4, horizontal axis is length(unit is k), vertical axis is time(unit is minute).

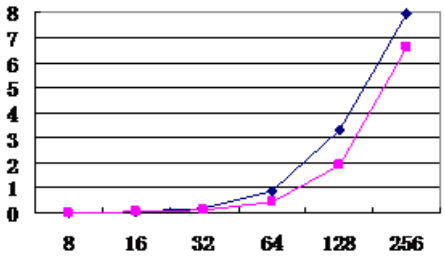

Figure 4. comparison between split and non-split to different length data

Fig .4 shows that equal parallel computing with split is faster than equal parallel computing without split. So the prime split method in equal parallel computing is efficiency.

Fig .5 shows speedup ratio before and after split in equal parallel computing and test data length respectively is 8k, 16k, 32k, 64k, 128k and 256k.

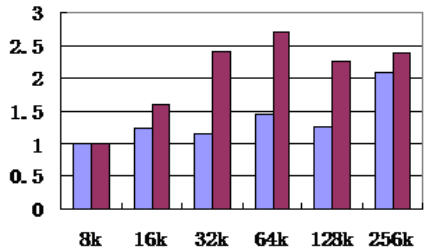

Figure 5. Speedup of split computing and non split computing

IV. UNEQUAL DIVISION PARALLEL COMPUTING

The shortage of equal division parallel computing is imbalance computing time in each regions. It can be known from universal combinatorial coding principle, the previous region consumes the longer computing time and the hinder region consumes the shorter computing time.

The total computing time depends on the first thread that consumes the longest computing time. It makes the negative waiting among the threads and the whole computing efficiency is low. For instance the sequence which length is 256k, if the sequence is divided into 4 regions equally, the permutation and combination calculating time of the first region is more one time than the last region"s.

A. *Principle of unequal division parallel*

In order to get balanced calculation time of each thread and reduce waiting time, it needs to adopt unequal division parallel computing. For example, the sequence which length is L can be divided unequally by the way of length increasing. In the follow test, the sequence is divided into 4 regions which length is 2/16, 3/16, 4/16 and 7/16.

It only needs to confirm right boundary because the left is confirmed in the first region. The right boundary is L/4 in equal division and it is at 2/16 in unequal division. The value is L/(4*4)*2. For the other regions, it needs to confirm left boundary. In equal division, the rough left boundary is L/4* region number. In unequal division, rough left boundary calculation is more complicated. Firstly, the rough left boundary value of the second region is L/(4*4)*2. The rough left value of each region is the right boundary of the previous one. According to the sequence length in unequal division, the left boundary value of 2, 3, 4 regions is respectively 2/16, 5/16, 9/16. The right boundary values of the second and the third region can be calculated, but the final region computation is special which right boundary value is L-1.

For instance the sequence which length is 256k, firstly, rough division can be done. As shown in Fig .6.

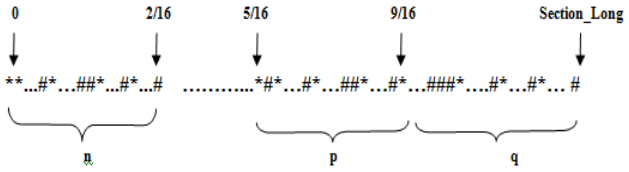

Figure 6. The initial division

Then the length of each region is adjusted reasonably. It makes each region begin with the non-first element of benchmark sequence and end with the first element of benchmark sequence (There may be many consecutive such elements until to the last one.). Thus, the new left and right boundarys are confirmed. It can be shown in Fig .7.

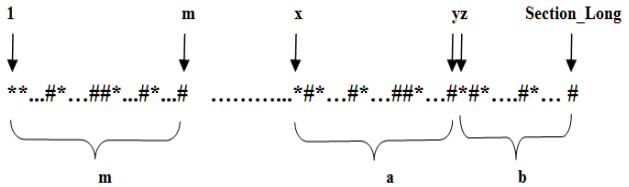

Figure 7. Regional adjustment

In Fig .6 and Fig .7, * is non-first element of benchmark sequence, # is the first element of benchmark sequence. Each adjusted region is divided according to the first element of benchmark sequence, so the combination value of each region can be computed parallel. For multi-core computer, each thread calculates the combination value of one region parallel. At last, the ordinal can be obtained by adding all the combination values.

B. *Test of unequal division method*

In order to verify unequal division parallel computing efficiency, test and comparison is done to the different

length data between equal division method and unequal division method. Test data length respectively is 8k, 16k, 32k, 64k, 128k and 256k. As shown in Fig .8, horizontal axis is data length (unit is k), vertical axis is time (unit is minute).

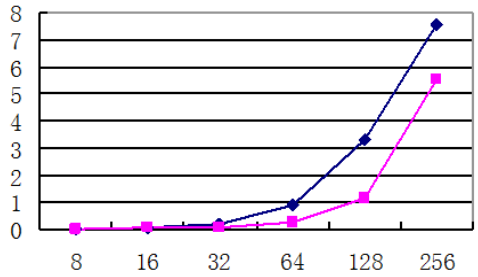

Figure. 8 parallel time comparison between equal and unequal division

Fig .8 shows that unequal parallel computing is faster than equal parallel computing. so unequal division parallel algorithm saves time and improves effective.

V. CONCLUSIONS

This thesis proposes two optimization methods about ordinal parallel computing. One is prime decomposing method. Prime decomposing makes multiplication and division offset in the process of calculation, so the computational complexity is greatly reduced. The other is unequal parallel computing that make unequal division to the sequence aiming at the shortage of the equal parallel method. Unequal division method makes each division region computed balanced and running time is similar. And then the whole running time is reduced. Experiments show that two kinds of optimization method effectively improve the ordinal parallel computing speed.

ACKNOWLEDGMENT

This research is supported by Key Laboratory of Database and Parallel Computing, Heilongjiang Province (KLDP-OF-2012-07), Natural Science Foundation of Heilongjiang Province of China (No. F201138), Scientific Research Fund of Heilongjiang Provincial Education Department of China (No. 12521395).

REFERENCES

[1] J.Rissanen, G.Langdon, “Compression of Black - White Image with Arithmetic Coding”, *IEEE Trans On Comm*. vol.29, no. 6, 1981, pp.858-867.

[2] J. Ziv , A. Lempel, “Compression of Individual Sequences via Variable Rate Coding”, *IEEE Transactions on Information Theory*. vol.24, no. 5, 1978, pp530-536.

[3] Jun Lu, Zhuo Zhang, and Juan Mo. “Research on Universal Combinatorial Coding”. The Scientific World Journal Volume 2014, Article ID 414613, 8 pages http://dx.doi.org/10.1155/2014/414613

[4] Lu Jun, Wang Tong, Liu Da-xin. “Research on Ordinal Properties in Combinatorics Coding Method”. Journal of Computers. vol. 6, Jan. 2011, pp. 51-58.

[5] Lu Jun, Liu DaXin,“Optimization of frequency table storage in Constant Grade Compression,” 2009 International Forum on Information Technology and Applications (IFITA 2009), ChengDu, P. R. China, May, 2009, pp. 72–74.

[6] Jun Lu, Tingting Gao, Juan Mo, Zhuo Zhang. “Optimization storage of Frequency table in Universal Combinatorial Coding”. International Conference on Mechatronics, Control and Electronic Engineering (MCE2014). Shenyang, China, August 27-29. 2014, Vol.101, pp 485-489.

[7] Lu Jun, Liu DaXin,“Optimization on Computing Base-data in Constant Grade Compression,” 2009 International Forum on Information Technology and Applications (IFITA 2009), ChengDu, P. R. China, May, 2009, pp. 68–71.

[8] Lu Jun, Liu Da-Xin,“Research on Parallel Technology within Section in Combinatorics Coding,” The 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, P. R. China, October,2010, pp.252-255.

[9] Zhuo Zhang, Jun Lu, Ting Ting Gao, Juan Mo, Yu Liu. Research on Max Ordinal in Universal Combinatorics Coding Based on GPU Parallel Computing. 2nd International Conference on Measurement, Information and Control (ICMIC 2013). Harbin, China, August 16-18. 2013, Vol.1, pp 457-461.

[10] Juan Mo, Jun Lu, Nan Wang, Zhuo Zhang, Yu Liu. The GPU Parallel Algorithm of Whole Ordinal in Universal Combinatorics Coding. 2nd International Conference on Measurement, Information and Control (ICMIC 2013). Harbin, China, August 16-18. 2013, Vol.1, pp 467-471.

[11] Lu Jun, Wang Tong, Li Yibing, “Study on Optimization Technology in Computing Ordinal Number,” Journal of Computers, vol. 5, Feb. 2010, pp. 210-217.