Parallelization of computations for lossless data compression

Sergey Novikov
Siedlce University of Natural Sciences and Humanities
Institute of Computer Science
3 Maja St. 54, 08-110 Siedlce, Poland

Abstract. The paper presents two algorithms of parallelization of computations for lossless data compression. It is proposed as a parallel algorithm for Huffman coding and a parallel algorithm for coding of a set of numbers by the family of arithmetic progressions.

Keywords. Coding, parallel algorithm, arithmetic progression, shortest covering.

1 Introduction

Currently, you can increase the efficiency of the solution of many important applied problems by applying modern multiprocessor computing systems (clusters). Computing clusters allow to solve applied problems simultaneously on multiple processors with the parallelization of computations.

There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism. However, we cannot increase the efficiency of decisions for large-scale problems significantly without developing appropriate parallel algorithms.

The aim of this paper is to report two algorithms of parallelization of computations for lossless data compression.

2 Parallel algorithm for Huffman coding

We first consider the problem of parallelization of computations with the help of the ***(m +1)***-cluster for classical Huffman coding [1] .

Huffman coding uses the specific method for choosing the representation for each symbol of a coded text resulting in a prefix code, that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols.

When parallelizing computations with the help of our parallel algorithm ***G1*** to solve the following subproblems:

1. The partition of the coded text ***T*** on ***m*** subtexts $T_1, T_2, \ldots, T_m$.

The control processor p_0 partitions the original text ***T*** on ***m*** subtexts $T_1, \ldots, T_m$ with the help of the program $A1(T; T_1, T_2, \ldots, T_m)$. Then the control processor p_0 sends subtexts $T_1, \ldots, T_m$ to processing processors $p_1, \ldots, p_m$ as input data.

2. Counting the frequency of occurrence of each character in the encoded text T_i.

Each processing processor counts the frequency of occurrence of each character in the encoded text T_i and constructs the frequency table $F(T_i)$ using the same program $A2(T_i; F(T_i))$.

Then each processing processor sends the solution (the frequency table $F(T_i)$) on the control processor.

3. Creating the Huffman tree and the table of codewords for ***T***.

The control processor adds solutions (frequency tables $F(T_1), \ldots, F(T_m);$) to create Huffman tree and the table of codewords ***KH(T)*** for ***T*** with the help of the program $A3(F(T_1), ..., F(T_m); KH(T))$. Then the control processor p_0 sends the table of codewords ***KH(T)*** to processing processors $p_1, \ldots, p_m$ as input data.

4. Compression of subtexts $T_1, T_2, \ldots, T_m$.

Once a Huffman code has been generated (we have ***KH(T)***), data may be encoded simply by replacing each symbol with its code. Each processing processor p_i encodes the subtext T_i with the help of the program $A4(T_i, KH(T); C(T_i), D(T_i))$ and sends the compressed text (block) $C(T_i)$ and the block mark $D(T_i)$ to the control processor. The block mark, which contains the block number and block size, may be required for decompression.

5. Compression of the text ***T***.

The control processor adds solutions (blocks $C(T_1), \ldots, C(T_m)$) to complete the encoding process of specified text ***T*** with the help of the program $A5(C(T_1), \ldots, C(T_m); C(T))$. The stream of prefix codes ***C(T)*** is the compressed original text ***T***.

Our parallel algorithm ***G1*** for Huffman coding implements the following computer schedule

$$S_{m+1}(G1)=((A1,p_0), (A2, p_1, \ldots, p_m), (A3,p_0), (A4, p_1, \ldots, p_m), (A5,p_0)),$$

where a record (Aj,p_i) indicates that the processor p_i performs the program ***Aj*** ; ***A1*** - program for partition of the coded text ***T*** on ***m*** subtexts; ***A2*** - program for counting the frequency of occurrence of each character in the encoded text T_i ; ***A3*** - program for creating the Huffman tree and the table of codewords for ***T***; ***A4*** - program for compression of subtexts; ***A5*** - program for compression of the text ***T***.

3 Parallel algorithm for coding of a number set by arithmetic progressions

The second our parallel algorithm ***G2*** was developed for building an optimal covering of set of numbers ***Q*** by the family of subsets of ***R(Q)***, the elements of each of which form an arithmetic progression. The solvable problem is interesting for coding theory and developers of software systems of manufacturing integrated circuits. A sequential algorithm to solve this problem is published in [2].

An arithmetic progression (AP) in the number set Q is denoted by the triple $M=(i,d,l)$, if her elements are numbers of the form $i+d*j$, where i is the first element of our progression, d – step our progression, $j \in \{0,1,\ldots,l-1\}$, $2 \le l \le |Q|$. Solving the above problem of building an optimal covering of the number set Q, we will consider only such AP, each of which is not covered no other AP in the Q.

The set of AP $R(Q)$ is called irreducible covering of the Q, if every element of the Q is a member of at least one of the AP from the family $R(Q)$, and removing from $R(Q)$ of at least one AP ceases to be a covering of Q. An irreducible covering $R^*(Q)$ is called optimal if it contains the minimum number of AP compared to other irreducible coverings of the set Q.

The solution of this problem on a multiprocessor cluster can be organized as follows.

1. The partition of the number set Q on m subsets $Q_1, Q_2, \ldots, Q_m$.

The control processor p_0 with the help of the program $A1(Q; Q_1, \ldots, Q_m)$ orders (if necessary) the elements of the Q, where $|Q| = n$, and divides them into m subsets Q_k, where $k \in \{1, \ldots, m\}$, $|Q_k| =]n/m[$ for $1 \le k \le m-1$ and $|Q_m| = n - (m-1)*]n/m[$. Then the control processor p_0 sends subsets $Q_1, \ldots, Q_m$ to processing processors $p_1,\ldots,p_m$ as input data.

2. Construction of optimal coverings of sets Q_k.

Each processing processor p_i constructs an irreducible covering of the set Q_k, where $k \in \{1, \ldots, m\}$ and $|Q_k|=n_k$, by the family of subsets $R(Q_k)$, the elements each of which form an AP through the same program $A2(Q_k; R'(Q_k))$. The program first generates all possible AP for Q_k with step $a_j-a_i=d$ for each $1 \le i \le n_k - 1$, $j = i+1$. These AP form the set $R'(Q_k)$, which is a covering of the set Q_k, but may contain excessive progressions.

To eliminate the redundant elements of $R'(Q_k)$, processing processor p_i with the help of the program $A3(Q_k, R'(Q_k); M_k)$ builds the Boolean matrix M_k, the columns of which correspond to elements of the set Q_k, and rows - to elements of $R'(Q_k)$. Our program in the process of filling out the elements of the Boolean matrix M_k writes 1 at the intersection of a i-row and a j-column, if AP appropriate i-line covers the element of the set Q_k corresponding the j-column, and writes 0, if AP appropriate i-line no covers the element of the set Q_k corresponding the j-column.

Then each processing processor p_i builds an optimal covering the columns M_k by lines with the help of the program $A4(M_k; R(Q_k))$. As the program $A4(M_k; R(Q_k))$ we can use our program **POKRMB** [3].

Then each processor p_i sends to the control processor p_0 the optimal covering $R(Q_k)$ of the set Q_k.

3. Preparing the data for summations progressions.

The control processor p_0 prepares the data to be combined AP of neighboring sets $R(Q_j)$ and $R(Q_{j+1})$ with the help of the program $A5(R(Q_1), \ldots, R(Q_k); (R(Q_j), R(Q_{j+1})))$. Initially (at first iteration) are such pairs $(R(Q_1), R(Q_2))$ for the processor p_1, $(R(Q_2), R(Q_3))$ for the processor $p_2,\ldots, (R(Q_{m-1}), R(Q_m))$ for the processor p_{m-1}. In the next iteration of the need to prepare the data for $m-2$ processors. Prepared for the p_1 the pair $(R(Q_1 \cup Q_2), R(Q_2 \cup Q_3))$, for the p_2 the pair $(R(Q_2 \cup Q_3), R(Q_3 \cup Q_4))$ and for the p_{m-2} the pair $(R(Q_{m-2} \cup Q_{m-1}), R(Q_{m-1} \cup Q_m))$. At the last $(m-1)$-th iteration is required to prepare data for only one processor p_1.

4. Summation and expansion of the AP from neighboring sets $R(Q_j)$ and $R(Q_{j+1})$.

For solving an applied problem with the help of a parallel algorithm the most important tasks are a task of decomposition of initial data for processing processors and a task of summation of solutions obtained by processing processors.

Both of these tasks can be solved trivially for parallel Huffman coding (algorithm ***G1***).

The task of summation of solutions obtained by processing processors in our algorithm ***G2*** solved as follows.

Each processing processor p_i combines and extends the AP from neighboring sets of irreducible coverings with the help of the program $A6(R(Q_j), R(Q_{j+1}); R(Q_j \cup Q_{j+1}))$. In this case, each AP from the $R(Q_j)$ is compared with each progression from the $R(Q_{j+1})$. An AP $(i_t,d_t,l_t) \in R(Q_j)$ can be combined into a single progression with an AP $(i_s,d_s,l_s) \in R(Q_{j+1})$ under two conditions: 1) $i_t+d_t*l_t = i_s$; 2) $d_t=d_s$. The result is the new AP $(i_t,d_t,l_t+l_s) \in R(Q_j \cup Q_{j+1})$ at first iteration**.** The result of summation of "overlapping progressions" at subsequent iterations (if the condition $d_s=d_t$ and $i_s+d_s*l_s > i_t$) is the new AP (i_t, d_t, l_t+l^*), where $l^*=(i_t - d_t - i_s)/d_s$.

In addition to combining the two progressions from neighboring sets $R(Q_j)$ and $R(Q_{j+1})$ may expand an AP by the following rules:

1) if an AP $(i_t,d_t,l_t) \in R(Q_j)$ and a number $i_t+d_t*l_t \in Q_{j+1}$, then the result is a new extension of the AP namely the AP $(i_t,d_t,l_t+1) \in R(Q_j \cup Q_{j+1})$;
2) if an AP $(i_s,d_s,l_s) \in R(Q_{j+1})$ and a number $i_s - d_s \in Q_j$, then the result is a new extension of the AP namely the AP $(i_s-d_s, d_s, l_s+1) \in R(Q_j \cup Q_{j+1})$.

Advanced progressions can be combined later in the performance of the above conditions.

5. Paragraphs 3 and 4 are executed $m-1$ times as long as there is no possibility of combining verified progressions of the final two sets to construct covering of Q labeled by $R'(Q)$.

6. Preparation of the data to eliminate redundant elements of $R'(Q)$.

The control processor prepares data to eliminate redundant AP of $R'(Q)$ with the help of the program $A7(Q, R'(Q); N)$. The columns of a Boolean matrix N correspond to elements of the set Q, and rows - to elements of $R'(Q)$. In the process of filling out the elements of the Boolean matrix N our program writes 1 at the intersection of a i-row and a j-column, if AP appropriate i-line covers the element of the set Q corresponding the j-column, and writes 0, if AP appropriate i-line no covers the element of the set Q corresponding the j-column.

7. Construction of the optimal coating of the set Q.

The control processor analyzes the rows of the matrix N on redundancy using the program $A8(N; R(Q))$. Excess progressions among elements of the covering $R'(Q)$ can occur as a result of summation and expansion of AP from $R(Q_j)$. To solve the problem of constructing the shortest cover of a Boolean matrix N we can use our program ***POKRMB*** [3]. However (if the condition $|R'(Q)| << |Q|$), check the line for redundancy can be significantly faster than building the shortest coverage with the help of the program ***POKRMB***. Indeed, a row ai of a matrix N is redundant if it can be covered by the disjunction of remaining rows of the matrix N. Therefore, to analyze the redundancy is advisable to use the special program $A8(N; R(Q))$. Thus be constructed the optimal coating of the set Q.

Our parallel algorithm ***G2*** for coding of the number set ***Q*** by the family of AP implements the following computer schedule

$S_{m+1}(G2)=((A1,p_0), (A2, p_1, \ldots, p_m), (A3, p_1, \ldots, p_m), (A4, p_1, \ldots, p_m), (A5,p_0)_1, (A6, p_1, \ldots, p_m)_1, (A5,p_0)_2, (A6, p_1, \ldots, p_{m-1})_2, \ldots, (A5,p_0)_{m-1}, (A6, p_1)_{m-1}, (A7,p_0), (A8,p_0))$,

where a record $(Aj,p_i)_s$ indicates that the processor p_i performs the program Aj on the s-th iteration; ***A1*** - program for partition of set of numbers ***Q*** on ***m*** subsets; ***A2*** - program for construction of an irreducible covering of the subset Q_k; ***A3*** - program for construction of the Boolean matrix M_k; ***A4*** - program for construction of an optimal covering the columns of a Boolean matrix M_k by lines; ***A5*** - program for preparing of the data for summation and expansion progressions; ***A6*** - program for summation and expansion of the AP from neighboring sets $R(Q_j)$ and $R(Q_{j+1})$; ***A7*** - program for preparation of the data to eliminate redundant elements of $R'(Q)$; ***A8*** - program for construction of an optimal covering of the set ***Q***.

4 Example

Let we must build the optimal covering of the set ***Q***, where

Q={14,16,17,20,23,24,25,26,28,29,30,32,33,35,37,38,40,41,42,43,44,45,46,47, 50,53,55,56,59,60}, by the family ***R(Q)*** of subsets, elements of each of which form an arithmetic progression, using a four-cluster.

The control processor divides the set ***Q*** into three subsets and sends to processing processors subsets Q_1, Q_2, Q_3, where

Q_1 ***={14,16,17,20,23,24,25,26,28,29}***,

Q_2 ***={30,32,33,35,37,38,40,41,42,43}***,

Q_3 ***={44,45,46,47,50,53,55,56, 59,60}***.

Then processing processors construct irreducible coverings of the sets Q_1, Q_2, Q_3, where

$R(Q_1)$***={(14,3,6), (16,4,4), (23,1,4)}***,

$R(Q_2)$***={(30,5,3), (32,3,4), (33,2,3), (40,1,4)}***,

$R(Q_3)$***={(44,1,4) ,(44,3,6), (45,5,4)}***.

The control processor prepares the data to be combined AP of neighboring sets $R(Q_1)$ and $R(Q_2)$ for the processing processor p_1 and the pair $(R(Q_2), R(Q_3))$ for the processor p_2.

The processing processors, working in parallel, combine and extend progressions.

When the processor p_1 combines AP $(14,3,6) \in R(Q_1)$ with $(32,3,4) \in R(Q_2)$ it receives the new AP $(14,3,10) \in R(Q_1 \cup Q_2)$. Thus, the processor p_1 receives the decision

$R(Q_1 \cup Q_2)$ ***={(14,3,10), (16,4,4), (23,1,4), (30,5,3), (33,2,3), (40,1,4)}***.

The processor p_2 combines AP $(30,5,3) \in R(Q_2)$ with $(45,5,4) \in R(Q_3)$. The result is the progression $(30,5,7) \in R(Q_2 \cup Q_3)$. In addition, the p_2 combines $(40,1,4) \in R(Q_2)$ with $(44,1,4) \in R(Q_3)$ and receives the new AP $(40,1,8) \in R(Q_2 \cup Q_3)$. The result of combining of AP $(32,3,4) \in R(Q_2)$ with $(44,3,6) \in R(Q_3)$ is the AP $(32,3,10) \in R(Q_2 \cup Q_3)$. Thus, the processor p_2 receives the decision

$R(Q_2 \cup Q_3)=\{(30,5,7), (32,3,10), (33,2,3), (40,1,8)\}$.

Further, the processor p_0 generates data for the processing processor p_1. It is the pair

$(R(Q_1 \cup Q_2), R(Q_2 \cup Q_3))$.

The processing processor p_1 combines and extends progressions.

When the processor p_1 combines the AP $(14,3,10) \in R(Q_1 \cup Q_2)$ with the AP $(32,3,10) \in R(Q_2 \cup Q_3)$ it receives the new AP $(14,3,16) \in R(Q_1 \cup Q_2 \cup Q_3)$.

The AP $(16,4,4) \in R(Q_1 \cup Q_2)$ can be extended to the progression $(16,4,5) \in R(Q_1 \cup Q_2 \cup Q_3)$. The AP $(30,5,3) \in R(Q_1 \cup Q_2)$ can be combined with the AP $(30,5,7) \in R(Q_2 \cup Q_3)$. Then the AP $(30,5,7) \in R(Q_2 \cup Q_3)$ can be extended to the progression $(20,5,9) \in R(Q_1 \cup Q_2 \cup Q_3)$.

Thus, the processing processor p_1 gets coverage

$R'(Q) =\{(14,3,16), (16,4,5), (23,1,4), (20,5,9), (33,2,3), (40,1,8)\}$.

The result $R'(Q)$ the processor p_1 sends to the processor p_0.

The control processor p_0 prepares the Boolean matrix N with 30 columns, that correspond to elements of the set $Q=\{14, 16, 17, 20, 23, 24, 25, 26, 28, 29, 30, 32, 33, 35, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 50, 53, 55, 56, 59, 60\}$ and 6 rows, tha correspond to elements of the set $R'(Q)=\{AP1=(14,3,16), AP2=(16,4,5), AP3=(23,1,4), AP4=(20,5,9), AP5=(33,2,3), AP6=(40,1,8)\}$ to eliminate redundant elements of $R'(Q)$.

Our Boolean matrix N has the form

	14	16	17	20	23	24	25	26	28	29	30	32	33	35	37	38	40	41	42	43	44	45	46	47	50	53	55	56	59	60
AP1	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	1	1	1	1	0	1	0
AP2	0	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AP3	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AP4	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	1
AP5	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AP6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Then the control processor builds the shortest coverage of the set Q.

The $AP3= (23,1,4)$ is a redundant element of the optimal covering.

Finally, we obtain the irreducible optimal covering

$R(Q) =\{(14,3,16), (16,4,5), (20,5,9), (33,2,3), (40,1,8)\}$.

5 Conclusion

Our studies suggest the following conclusions.

For an increase of the efficiency of lossless data compression (Huffman coding and encoding of a set of numbers by the family of arithmetic progressions) you can use multiprocessor computing clusters.

The task of decomposition of initial data for processing processors and the task of summation of solutions obtained by processing processors can be solved trivially for parallel Huffman coding.

The task of summation of solutions obtained by processing processors for parallel coding of set of numbers by the family of arithmetic progressions can be solved with the help of tools proposed in this paper.

The process of decompression of Huffman code is simple. For decompression you can use ***(m+1)***-cluster and block marks $D(T_i)$ needed to break the stream of prefix codes $C(T)$ on m blocks, each of which control processor p_0 sends (with the table of codewords $KH(T)$) to processing processors. Then each processing processor decodes the block $C(T_i)$ and sends to the control processor p_0 the subtext T_i for summation.

The process of decompression of set of numbers Q, encoded by the family of arithmetic progressions also is simple. The control processor p_0 sends to processing processor the AP from $R(Q)$. The processing processor builds the $|Q|$-dimensional bit vector, which uniquely identifies the AP. Then the processing processor sends the $|Q|$-dimensional bit vector to the control processor, which summarizes the vectors using the disjunction operation.

References

1. Huffman D.A., A Method for the Construction of Minimum-Redundancy Codes, Proceedings of the I.R.E., September 1952, pp 1098-1102.
2. Новиков С.В., Олексин С.В. О покрытии множества арифметическими прогрессиями. // Известия Академии Наук БССР, Серия физ-мат наук, 6, 1979, с. 25-27.
3. Novikov S., Adamus A., Investigations of the efficiency of a parallel program for construction of the shortest covering of a Boolean matrix. Studia Informatica. Systems and information technology, Volume 1-2(14)2010, Wyd. UPH, Siedlce, 2011, p. 67-76.