The Large-scale Dynamic Data Rapid Reduction Algorithm Based on Map-Reduce

Jing-ling Yuan
Computer Science and Technology school, Wuhan University of Technology, Wuhan, China
Email:yuanjingling@126.com

Jing Xie, Yan Yuan, Lin Li
Computer Science and Technology school, Wuhan University of Technology, Wuhan, China
School of Urban Design, Wuhan University, Wuhan, China
Email:kk-yuhui@163.com

***Abstract*—With the advent of the era of "Big Data", the application of the large-scale data is becoming popular. Efficiently using and analyzing the data has become an interesting research topic. Traditional knowledge reduction algorithms read small data samples once into a computer main memory for reduction, but it is not suitable for large-scale data. This paper takes large-scale sensor monitoring dynamic data as the research object and puts forward an incremental reduction algorithm based on Map-Reduce. Using a Hash fast partitioning strategy this algorithm divides the dynamic data set into multiple subdatasets to compute, which has greatly reduced the calculation time and space complexity of each node. Finally, experiments are conducted on the data from UCI Machine Learning Repository using Hadoop platform to prove that the algorithm is efficient and suitable for large-scale dynamic data. Compared to the traditional algorithms, the highest speedup of the parallel algorithm can be increased up to 1.55 times.**

***Index Terms*— Large-scale dynamic data, increment knowledge reduction, Hash algorithm, Map-Reduce**

I. INTRODUCTION

With the advent of the era of "Big Data", the application of the large-scale data such as all kinds of sensor data, network data, mobile device data, RFID data and so on [1] is becoming more and more popular. We will confront the problem of ample data and poor knowledge. How to use and analyze this large-scale data efficiently is becoming an important issue. Researchers obtain effective information by various methods such as data mining, knowledge reduction and so on. As a form of data reduction, knowledge reduction is the preprocessing set of data mining and it deletes unnecessary or unrelated knowledge on the premises of keeping the data classification ability to reduce the time space exploration and improves the efficiency of follow-up work. One of the core problems of rough set theory (RST) put forward by Z.Pawlak[21] who is a mathematician from Poland is knowledge reduction. RST can effectively analyze and deal with all kinds of incomplete information which is inaccurate, inconsistent or incomplete, and discovers tacit knowledge and suggests potential rules. Thus RST plays an important role in many fields such as data mining, pattern recognition, decision analysis, image processing, medical diagnosis, artificial intelligence and so on.

The contributions of this paper are:

1. We state a traditional dynamic data reduction algorithm and mainly analyze the incremental reduction algorithm. The algorithm firstly obtains the reduction of a part of decision set, and adds the rest step by step. After several iterations it would get the final reduction set.

2. We further study the incremental reduction algorithm based on distributed computing framework Map-Reduce. With the idea of parallel, the algorithm divides large-scale data into small data fragmentations to calculate separately and gathers the results to get the final reduction. Through simulation experiment we know that for large-scale dynamic data, the parallel reduction algorithm based on Map-Reduce is more efficient compared to traditional dynamic data reduction algorithms.

II. RELATED WORK

Many scholars have carried out extensive researches for reduction algorithms.[2,3,4,5,6] Skowron A. et al. [7, 8, 9, 10] presented a knowledge reduction algorithm based on discernibility matrices and its time complexity was $O(|C^2||U^2|)$. Liu Shaohui et al. [11] presented an equivalence partitioning algorithm which used quick sort scheme to sort knowledge set and its time complexity was $O(|C||U||logU|)$. Xu Zhangyan[12] presented a positive region algorithm based on radix sort and its time complexity was $O(|C||U|)$. Wang Guoyin et al. [13, 14, 15] introduced a reduction algorithm based on information entropy whose time complexity was $O(|C^2||U^2|)$. Feng Lin et al. [19, 20] introduced a reduction algorithm based on continuous valued attributes whose time complexity was $O(|C||U|)$. In practice, when processing more data, large-scale data cannot be resided in memory at all, it will require a mass of I/O operations which consume a

doi:10.4304/jsw.9.4.1028-1035

lot of time and increase the time cost of reduction that causes inefficiency. For large-scale data set, Wang Guoyin et al. [16, 17] presented a rapid knowledge reduction algorithm based on the idea of divide and conquer. Combined with the idea of parallel computing the algorithm assigned the task of knowledge reduction to multiple processors to operate at the same time, which has greatly improved the efficiency. Its time complexity was $O(|C^2||U|)$ and space complexity was $O(|U|+\mathrm{P}\times|C|)$. Wu Zite et al. [18] put forward a fast scalable attribute reduction algorithm with the thought of SLIQ. The algorithm divided the decision table into different knowledge lists lengthways and stored them in the hard disk only when necessary to load into memory. Because only one decision table existed in memory, it could save a lot of memory space and help to improve the efficiency.

However, at present, the study of methods for processing large-scale dynamic data is a few. Because of the accumulation of the sensor data, large-scale dynamic data exists in many practical engineering monitoring projects such as the monitoring of tunnel and bridge, building health detection and so on. Based on the large-scale dynamic data set, this paper learns from the traditional reduction algorithm and uses the idea of distributed architecture and parallel computing to achieve the large-scale dynamic data reduction algorithm. The algorithm calculates distributed file reduction results, then uses dynamic incremental reduction algorithm to get the final reduction results. The algorithm does not need a large number of I/O operations. So it greatly saves the computation time and improves the efficiency of reduction.

III. Traditional Reduction Algorithm

Recently, rough set theory is an effective mathematical tool for dealing with uncertainty besides probability theory, fuzzy sets and evidence theory. As a relatively new soft computing method, rough set has received increasing attention and its efficiency has been confirmed in some successful applications of many science and engineer fields. Rough set is one of the hot spots in current artificial intelligence theory and its applications. In many practical systems there are uncertainty factors to various degrees, and collected data often contains noises, indeterminacy and imperfection. The main idea of rough set theory is that using the known knowledge base to portray the imprecise and uncertain knowledge approximately.

A. Basic Concepts

In this section, several basic concepts are reviewed, such as information systems, decision tables, equivalence relation, partition, lower and upper approximations and partial relation of knowledge.

In the following, we first recall the concepts of information systems.

Definition 1. An information system is a pair $IS = (\mathrm{U}, \mathrm{A} = \mathrm{C}\cup \mathrm{D}, \mathrm{V}, \mathrm{f})$, where

(1) U is a non-empty finite set of objects;

(2) A is a non-empty finite set of knowledge; and

(3) For every knowledge $a \in A$, there is a mapping f, $f: U \to V_a$ with V_a being called the value set of $a \in A$.

A decision table is an information system $S = (U, A = C \cup D, V, f)$ with $C \cap D = \Phi$ where each element of C is called condition knowledge, C is called a condition knowledge set, each element of D is called decision knowledge, and D is called a decision knowledge set.

Definition 2.

Given a decision table $DS = (U, A = C \cup D, V, f)$ and a knowledge set $P \subseteq (\mathrm{C}\cup D)$. P defines an equivalence relation $IND(\mathrm{P})$ on U as

$$IND(\mathrm{P}) = \{(\mathrm{x}, \mathrm{y}) \mid ((\mathrm{x}, \mathrm{y}) \in \mathrm{U}\times \mathrm{U})(\forall\, \mathrm{a} \in \mathrm{P}(\mathrm{a}(\mathrm{x}) = \mathrm{a}(\mathrm{y})))\}.$$

Obviously, $IND(\mathrm{P})$ is an indiscernibility relation. The partition of P on U is marked by $U / IND(\mathrm{P})$ or just U/P by notation of $[\mathrm{x}]_{IND(\mathrm{P})} = [\mathrm{x}]_p$. We refer to the equivalence block of P containing the instance $\forall x \in U$.

Definition 3.

Given a decision table $DS = (U, A = C \cup D, V, f)$, if $P \subseteq (\mathrm{C}\cup D)$ and $x \in U$, the lower-approximation of X to P is $P\_ = \{\mathrm{x} \mid (\forall\, \mathrm{x} \in \mathrm{U}) \wedge ([\mathrm{x}]_p \subseteq \mathrm{U})\}$ and the upper-approximation of X to P is $P^{-}(\mathrm{x}) = \{\mathrm{x} \mid (\forall\, \mathrm{x} \in \mathrm{U}) \wedge ([\mathrm{x}]_p \cap X \neq \Phi)\}$

Definition 4[15].

Given a decision table $DS = (U, A = C \cup D, V, f)$, $P \subseteq (\mathrm{C}\cup \mathrm{D})$, the positive region of P with reference to D is defined as $POS_p(\mathrm{D}) = \cup_{X \in U / IND(D} P\_(\mathrm{X})$.

The decision values of instances in $POS_p(\mathrm{D})$ can be completely predicted according to their corresponding condition knowledge values vectors, this is, $POS_p(\mathrm{D})$ is the deterministic part of the universe of a decision table.

Definition 5[15].

A decision table $DS = (U, A = C \cup D, V, f)$ is consistent if and only if $POS_p(\mathrm{D}) = \mathrm{U}$.

Definition 6.

An information system $DS = (U, A = C \cup D, V, f)$, where C is called a condition knowledge set, and D is called a decision knowledge set, $P \subseteq C$, $E = U / P$, $x_1, x_2 \in U$. If decision values of all elements in E are same, then E is called consistent classification. Otherwise, E is called inconsistent classification. That is to say for $\forall x_1, x_2 \in E$, if $d(\mathrm{x}_1) = \mathrm{d}(\mathrm{x}_2)$, E is called consistent classification. When $\mathrm{P} = \mathrm{C}$ consistent classification is also known as consistent subset or consistent record and inconsistent classification is also known as inconsistent subset.

Definition 7.

An information system $DS = (U, A = C \cup D, V, f)$, where C is called a condition knowledge set, $P \subseteq C$, $G = U / P$, if G is consistent classification, then G is called consistent state.

Definition 8.

An information system $DS = (U, A = C \cup D, V, f)$, where C is called a condition knowledge set, and D is called a decision knowledge set, $P \subseteq C$, $Ep = \{X \mid X \text{ is consistent subset and } X \subseteq POS_p(D)\}$, suppose $a \in C$, if $|POS_C(D)| \neq |POS_{C-\{a\}}(D)|$, then for D knowledge a is indispensable.

Supposing $R \subseteq C$, if satisfied to restrain:

(1) $|POS_C(D)| = |POS_R(D)|$

(2) $\forall a \in R$, $|POS_R(D)| \neq |POS_{R-\{a\}}(D)|$, we call R is a relative reduction of C. The interaction of relative reduction in C is called core of C noted as $CORE_D(C)$.

B. Traditional Reduction Algorithm

Generally speaking, the traditional dynamic data reduction algorithm is a kind of incremental algorithms. The algorithm firstly obtained the reduction of a part of decision set, and added the rest step by step. After several iterations it would get the final reduction set.

In an information system S_0={U,C,V,F} the core is CORE(S_0), reduction set is R_0, the incremental attribute set is X. In a new information system S={U,C',V',F'}, C'=C$\cup$X, the core is CORE(S), reduction set is R. In the information system S_0'={U,X,V',F'}composed by X, the core is CORE(S_0'), reduction set is R_0'. The relationship between the several sets is as follow:

$$CORE(S) = CORE(S_0) + CORE(C) -$$
$$\{m \mid m \in CORE(S_0) \cap POS_{R_0'}^{\Delta c|m|} = \Delta c \mid m|\} -$$
$$\{n \mid n \in CORE(S_0') \cap POS_{R_0}^{\Delta X|n|} = \Delta X \mid n|\}$$

The core of original system plus the core of incremental system, removes the redundant attributes to get the core of new information system. The reduction set is based on the reduction of original system and gets rid of the redundant attributes to get the final reduction set. Multiple attributes incremental reduction algorithm is described as follow:

Input: S_0={U,C,V,F}, S_0'={U,X,V',F'}
Output: CORE(S), R
1. S=$S_0 \cup S_0$'; R=R_0
2. S_0=NULL; for every attribute a of S_0, compute M_0=$\triangle a(\{a\})$, $N_0 = POS_{R_0'}^{M_0}(\{a\})$.
   If (M_0= =N_0), S0=S0$\cup a$
3. S_0'=NULL, for every attribute a of S_0', compute M_0'=$\triangle X(\{a\})$, $N_0' = POS_{R_0'}^{M_0'}(\{a\})$.
   If (M_0'= =N_0'), S_0'=S_0' $\cup a$
4. CORE(S)=CORE(S)-S_0-S_0'
5. Set T'=NULL; for every attribute a of R_1, compute M=$\triangle R_1(\{a\})$, $N = POS_{R_0}^{M}(\{a\})$
   If (M= =N&&$POS_{R1-\{a\}}(\{a\}) \in POS_{R1-\{a\}}(\{R_0\})$) T'=T' $\cup \{a\}$ go to 7
6. $t = \arg\min_{w \in t}(\text{card}(\Delta R_1(\{w\})))$, R_1=R1-{t}
7. If (T'!= NULL) go to 6; else go to 8
8. T= NULL, for every attribute a of R, compute M=$\triangle$R($\{a\}$), $N = POS_{R_1}^{M}(\{a\})$
   If (M= =N&&$POS_{R0-\{a\}}(\{a\}) \in POS_{R0-\{a\}}(\{R_t\})$) T'=T $\cup \{b\}$ go to 10
9. $t = \arg\min_{w \in T}(\text{card}(\Delta R_1(\{w\})))$, R=R-{t}, go to 8
10. If (T= =NULL) go to 9, else R=R$\cup R_1$, R is the reduction set we want to get.

Incremental attributes reduction algorithm can process the relationship between the original reduction and the assuring reduction well when attributes increase. But when the data volume is very big, computers need to process I/O operations frequently, memory utilization rate is higher, and computation needs a lot of time. The traditional incremental reduction algorithms cannot deal with time overhead of large-scale data reduction process. Therefore, for the reduction of large-scale dynamic data, this paper puts forward a reduction algorithm under the framework of Map-Reduce.

IV. The Reduction Algorithm Based On Mapreduce

Under the challenges of big data, the scale of modern system is bigger and bigger and the data is more and more. In terms of large-scale data, we cannot read all data into memory all at once, so we need to use disk access. We can read a part of the data into memory firstly, compute the reduction set R_1 and write it to the file. And then read another part into memory, compute the reduction set R_2 and write it to the file. Until reading the data to the end, we can use the reduction sets we have gotten to get the final reduction set that we want.

A. Hadoop

Today, we are surrounded by data. People upload videos, take pictures on their cell phones, text friends, update their Facebook status, leave comments around the web, click on ads, and so forth. Machines, too, are generating and keeping more and more data. The exponential growth of data first presented challenges to current computing equipment. Existing tools were becoming inadequate to process such large data set like terabytes and petabytes. A system processing large-scale distribute data has aroused a lot of interest. Hadoop is a framework for writing and running distributed applications that process large-scale data. Distributed computing is a wide and varied field, but the key distinctions of Hadoop are as follow.

1) Accessible. Hadoop runs on large clusters of commodity machines or on cloud computing services.
2) Robust. Because it is intended to run on commodity hardware, Hadoop is architected with the assumption of frequent hardware malfunctions. It can gracefully handle most such failures.
3) Scalable. Hadoop scales linearly to handle larger data by adding more nodes to the cluster.
4) Simple. Hadoop allows users to quickly write efficient parallel code.

Hadoop's accessibility and simplicity give it an edge over writing and running large distributed programs. On

the other hand, its robustness and scalability make it suitable for even the most demanding jobs at Facebook. These features make Hadoop popular in both academia and industry.

B. Map-Reduce Framework

Map-Reduce is a programming model for processing large data sets with a parallel and distributed algorithm. It includes three aspects:(1) the distributed file system; (2) the parallel programming model and (3) parallel execution engine. A Map-Reduce program comprises a Map() procedure that performs filtering and sorting and a Reduce() procedure that performs a summary operation.

The Map () and Reduce () functions of Map-Reduce are both defined with respect to data structured in (key, value) pairs. Map () takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain: Map(k1,v1) → list(k2,v2). The Map function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. After that, the Map-Reduce framework collects all pairs with the same key from all lists and groups them together, creating one group for each key. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain: Reduce (k2, list (v2)) → (k3, v3). The specific calculation process is shown as Figure 1.

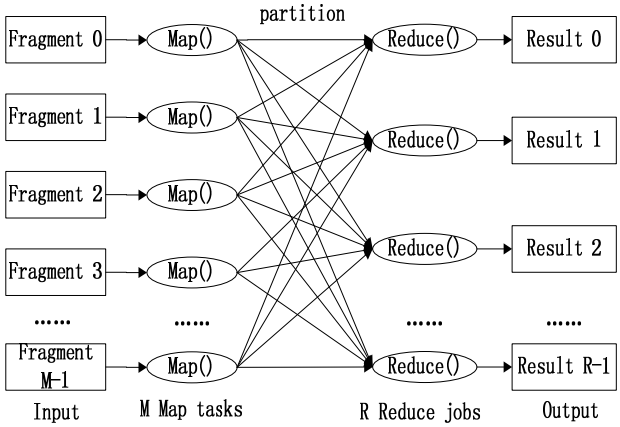

Figure 1. The computing process of Map-Reduce

To serve as the mapper, a class implements from the Mapper interface and inherits the MapReduceBase class. The Mapper interface is responsible for the data processing step. It utilizes Java generics of the form Mapper<K1, V1, K2, V2> where the key classes and value classes implement the WritableComparable and Writable interfaces, respectively. Its single method is to process an individual (key / value) pair:

```
Void map (K1 key,
          V1 value,
          OutputCollector<K2, V2> output,
          Reporter reporter
          ) throws IOException
```

The function generates a (possibly empty) list of (K2, V2) pairs for a given (K1, V1) input pair. The OutputCollector receives, and the Reporter provides the option to record extra information about the mapper as the task progresses.

When the reducer task receives the output from the various mappers, it sorts the incoming data on the key of the (key / value) pair and groups together all values of the same key. The reduce() function is then called, and it generates a (possibly empty) list of (K3, V3) pairs by iterating over the values associated with a given key. The OutputCollector receives the output of the reduce process and writes it to an output file. The Reporter provides the option to record extra information about the reducer as the task progresses.

C. Large-scale Dynamic Data Parallel Reduction Algorithm Based on Map-Reduce

The reduction based on Map-Reduce is a kind of parallel reduction, which should solve two problems such as finding the parallel point results and obtaining the final result by local ones. The data can be divided into disjoint subsets by Map-Reduce to form a number of data fragmentations. Map () procedure completes the equivalence class computing of different data fragmentations and Reduce () procedure accomplishes the computing of number of the positive region, information Entropy or undecipherable object in the same equivalence class. The two procedures realize the data and task parallelism in the reduction algorithm based on Map-Reduce. Figure 2 shows the Map-Reduce task flow chart.

Applying the idea of parallelism to traditional hash reduction algorithm, the parallelism of the algorithm is shown as follow.

1) Data parallelism.

ⅰ When calculating the hash table H_Q of the information system, we can divide the original data into different fragmentations and obtain the hash table H_i of each data fragmentation. We can get the hash table H_Q of the original data set through $H_i, H_Q = \cup_{i=1}^{n} H_i$.

ⅱ When calculating the importance of attributes, the optional attributes can be sliced to get the importance of every attribute in each fragmentation, and then summarize the importance of all attributes to get the attribute having the biggest importance.

2) Task parallelism.

ⅰ When calculating core attributes, we can use task parallelism. Each task computes whether a list of attributes are core attributes or not separately and summarizes all the results.

ⅱ When calculating the number of inconsistent set in hash table, we can use task parallelism for each fragmentation. We can get the importance of each optional attribute to improve the parallelism.

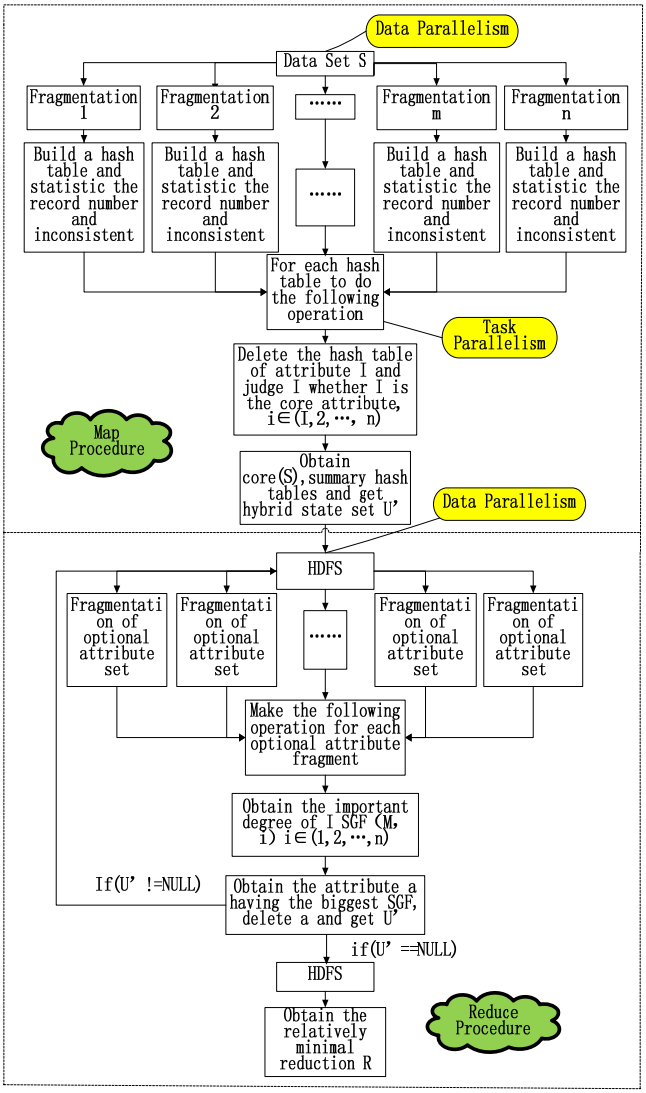

Figure 2. Map-Reduce task flow chart

The parallel reduction algorithm based on Map-Reduce mainly includes 3 parts which are Map function, Reduce function and Master control process described as follow.

1) Map function

Input: data fragmentation S_i, optional attribution C
Output: <hash *table* H_0, *Core*(S_i) >

① For data fragmentation S_i using hash operation to get hash table H_i and statistic the inconsistent records in hash tables.

② Using the idea of task parallelism to get the core of H_i Supposing there are m optional attributes and m copy of H_i which should delete the attribute column to judge whether the attribute is core attribute or not. If it is the core attribute, it can be add into *Core*.

H_i = hash(S_i); // Do hash operation for S_i
for(each $a \in$C)// Operate each attribute in m
{
if(is Core(a))// Judge whether attribute a is a core attribute
{
Core = Core$\cup a$;// If a is the core attribute, then add it into Core
}
}

2) Reduce function

Input: the fragmentation of inconsistent set S_i, Core
Output: <attribute set C, attribute weighting score *SGF*>
For each attribute data fragmentation D_i, we can use the idea of task parallelism. Supposing there are m optional attributes and m copy of D_i, we can get the weighting score of each attribute in D_i and output <attribute set C, attribute importance *SGF*>.
for(each $a \in$ C)// for each attribute a in C
{
SGF(a);// compute attribute weighting score which is the number of the inconsistent set in D_i
}

3) Master Control Process

Input: information system $U(C, D)$
Output: relatively minimal reduction $R(U)$
① Set R to *NULL*
② The information system U can be sliced and then start a Map function. After all fragmentations performing the Map function, it will summarize the results of fragmentations and get the hash structure and core of U.
③ Taking the *Core* as effective attributes, we can do hash operation for hash tables of U and get the inconsistent set recording U'.
④ For the data partition of the optional attributes in U', starting a process and performing the Reduction function to find the attribute a which has the greatest *SGF* and put a into the *Core*.
⑤ If $U' \neq NULL$, then skip to step ④, else continue.
⑥ Check the redundancy of the *Core* and delete the redundant attributes.
At this time, the *Core* is also the final reduction set R.

V. Algorithm Analysis and Experiment

A. Map-Reduce Reduction Model Analysis

Supposing there are n nodes in parallel, in original algorithm, the time for hash procedure is T_H, and the time for computing *Core* is T_C. In the Map function, because of fragmentation, the ideal time for hash is T_H / n and for

Core is T_C / n. Adding to the communicating time, in the whole Map procedure, the ideal speedup is :

$$speed = (T_H + T_C) \Big/ \left(\frac{T_H}{n} + \frac{T_C}{n} + T_S \right)$$

In Reduce function, the time for getting all attributes significance is T_i, because of fragmentation, the ideal time is T_i / n. The time for counting the sum of the importance is T_m, and adding the communicating time T_S, the ideal speedup is:

$$speed = (T_i) \Big/ \left(\frac{T_i}{n} + T_m + T_S \right)$$

From the two speedups, we can know that the parallel reduction algorithm based on Map-Reduce is difficult to achieve the ideal speedup n , because with the increase of the data size, the communication time also increases. When the decision set is small, time and reduction efficiency of parallel algorithm is worse than the serial algorithm. Only when the data volume is large and the computation time of each fragmentation is far greater than communication overhead, parallel algorithm can reflect its value and function.

B. The Comparison and Analysis of The Experiment

Experiments are conducted on the data from UCI Machine Learning Repository using Hadoop platform on a PIII800PC(512M RAM, Win). In experiment 1, we use the traditional reduction algorithm (algorithm a) and the reduction algorithm proposed by this paper (algorithm b) respectively. We reduce six decision tables from UCI machine learning repository. The results are shown in Table 1.

TABLE 1
THE COMPARISON OF EXPERIMENT 1

Data Set	Number of instances	Number of attributes	Algorithm a	Algorithm b
			T/ms	T/ms
Livedisorder	345	6	82	98
Tic-tac-toe	958	9	296	327
Mushroom	8124	22	5705	5016
Letter-recogn ition	20000	16	11514	9821
Chess	28056	6	10170	8959
covtype	581012	54	—	97257

In experiment 2, we take Letter-recognition, Chess and n copies of the two sets as our experimental data to reflect the superiority of Map-Reduce work better when dealing with the large-scale dynamic data. L_1 and C_1 are the results of Letter-recognition and Chess copying twice, and L_2 and C_2 are the results of Letter-recognition and Chess copying four times. The results are shown in Table 2.

TABLE 2
THE COMPARISON OF EXPERIMENT 2

Data Set	Number of instances	Number of attributes	Algorithm a	Algorithm b
			T/ms	T/ms
Letter-reco gnition	20000	16	11514	9821
Chess	28056	6	10170	8959
L_1	40000	16	19354	12728
C_1	56112	6	16803	11625
L_2	80000	16	30013	19367
C_2	112224	6	31278	21569

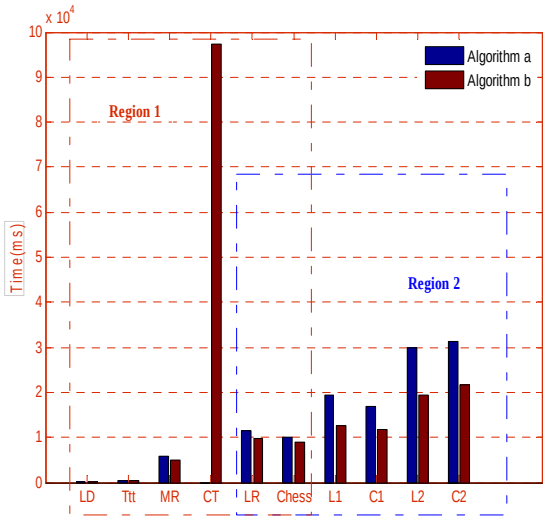

Figure 3. The experiment results

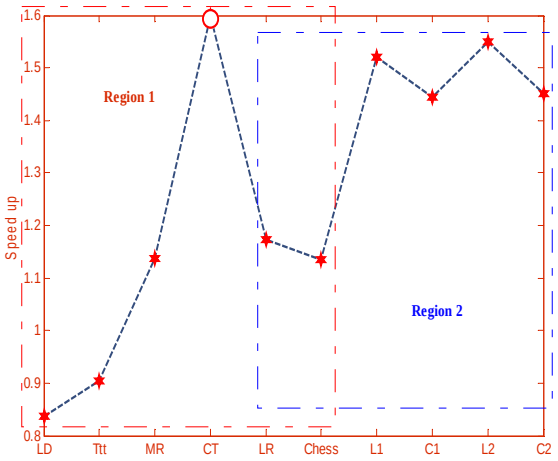

Figure 4. The comparison of speedup

In Figure 3, from the Region 1, we know that for small-scale data the speed of traditional reduction algorithm (Algorithm a) is faster than the parallel reduction algorithm based on Map-Reduce (Algorithm b), but when the data size is bigger, algorithm b is superior to algorithm a, especially when the data size is more than 500000 lines the advantage is more obvious.

In Figure 3, from the Region 2, we can know that with the increase of the data size, for the same data set the running time gap of the two algorithms is larger and larger.

In Figure 4, from the Region 1, we know that when the data size is very small the speedup is less than 1. That is to say for sample data the traditional reduction algorithm is superior to the parallel reduction algorithm based on Map-Reduce. When the data size is large the efficiency is shown. This is due to the fragmentation time is greater than the communication time in the parallel reduction algorithm.

In Figure 4, from the Region 2, we can know that for the same data set the speedup increases with the increase of the data size.

The experiments simulate 3 nodes under the Hadoop pseudo-distributed mode, and the results cannot fully reflect the advantages of Map-Reduce in dealing with the large-scale data. But it also can be seen that with the increase of the data size the advantages of parallel algorithm are more obvious and the overall time grows gently. In this paper we only extract some parallel points from the hash reduction algorithm to use. There are some serial parts in Map-Reduce, when the data set is very large it would significantly affect the efficiency of parallelism and the speedup. So we can continue to improve the parallel algorithm in the future research and design specific algorithm for parallel reduction. Our experiments were conducted in pseudo-distributed mode and the communication overhead is very small so as to have a certain influence on the results. However in general, Map-Reduce is efficient, and can effectively deal with large-scale data reduction problems and obtain ideal results. Setting up Hadoop in fully-distributed mode, we can simulate more nodes so that Map-Reduce can play its advantages better.

VI. Conclusion

In order to realize the reduction of large-scale dynamic data, parallel knowledge reduction model based on Map-Reduce is put forward by deeply analyzing the parallelizable operation in traditional Hash algorithm in this paper. Simulation experiments are conducted using Hadoop platform and results are compared with a traditional dynamic data reduction algorithm. We find that the algorithm proposed is efficient in processing large-scale dynamic data. The highest speedup is up to 1.55 times. In this paper we have obtained some preliminary results but there is a large room to improve. There are some serial parts in the algorithm so that we should design new parallel algorithms to meet the requirement of the parallel reduction for large-scale data in the future research.

Acknowledgement

This paper is supported by National Natural Science Foundation of China under Grant Number 51208388 and 61303029, National Key Technology R&D Program under Grant Number 2012BAH89F00, and The Fundamental Research Funs for the Central University under Grant Number 2013-IV-054. At the same time, I would like to express my gratitude to Yawar Abbas Bangash who helped me during the modification of this paper.

References

[1] Qin XiongPai, Wang HuiJu, Du XiaoYong, Wang Shan. Big Data Analysis—Competition and Symbiosis of RDBMS and MapReduce. Journal of Software, 2012.23(1):32-45.

[2] Chouchoulas A., Shen Q. Rough set-aided keyword reduction for text categorization. Applied Artificial Intelligence, 2001, 15(9): 843-873.

[3] Qianjin Wei, Tianlong Gu. Ymbolic Representation for Rough Set Attribute Reduction Using Ordered Binary Decision Diagrams. Journal of Software. 2011, 6(6):977-984.

[4] Xu E, Yuqiang Yang, Yongchang Ren. A New Method of Attribute Reduction Based On Information Quantity in An Incomplete System. Journal of Software. 2012, 7(8):1881-1888.

[5] Lin T. Y., Yin P. Heuristically Fast Finding of the Shortest Reducts. Rough Sets and Current Trends in Computing (RSCTC2004), Uppsala, Sweden, 2004: 465-470.

[6] Swiniarski R.W., Skowron A. Rough set methods in feature selection and recognition, Pattern Recognition Letters, 2003, 24(6): 833-849.

[7] Skowron A., Rauszer C. The discernibility matrices and functions in information systems // R. Slowinski. Intelligent Decision Support: Handbook of Applications and Advances to Rough Sets Theory. Dordrecht: Kluwer Academic, 1992: 331-362.

[8] Ye Dongyi, Chen Zhaojiong. A New Discernibility Matrix and the Computation of a Core. Chinese Journal of Electronics,2002,30(7):1086-1088

[9] M. Yang, Z.H. Sun, Improvement of discernibility matrix and the computation of a core, Journal of Fudan University (Natural Science) 43 (2004) 865–868. in Chinese.

[10] Yang Ming. An Incremental Updating Algorithm for Attribute Reduction Based on Improved Discernibility Matrix. Chinese Journal of Computers, 2007, 30 (5):815-821.

[11] Liu ShaoHui, Sheng QiuJian, Wu Bin, Shi ZhongZhi, Hu Fei. Research on efficient algorithms for rough set methods. Chinese Journal of Computers, 2003, 26(5): 524-529.

[12] Xu ZhangYan, Liu ZuoPeng, Yang BingRu, Song Wei. A quick attribute reduction algorithm with complexity of max (O(|C||U|),O(|C|2|U/C|)), Chinese Journal of Computers, 2006, 29(3):391-399.

[13] Wang Guoyin, Yu Hong, Yang Dachun. Decision Table Reduction Based on Conditional Information Entropy. Chinese Journal of Computers, 2002, 25 (7):759-766.

[14] Guoyin Wang, Jun Zhao, Jiujiang An, Yu Wu. A Comparative Study of Algebra Viewpoint and Information Viewpoint in Attribute Reduction. Fundamenta Informaticae 68(2005)289-301.

[15] Li Xiaoyong, Gui Xiaolin, Mao Qian, Leng Dongqi. Adaptive Dynamic Trust Measurement and Prediction Model Based on Behavior Monitoring. Chinese Journal of Computers. 2009, 32(4):664-674.

[16] Hu Feng, Wang Guo-Yin.Quick Reduction Algorithm Based on Attribute Order. Chinese Journal of Computers. 2007, 30(6):1429-1435.

[17] Xiao Da-wei,Wang Guo-yin,Hu Feng. Fast Parallel Attribute Reduction Algorithm Based on Rough Set Theory. Computer Science. 2009, 36(3):208-211.
[18] Wu Zite, Ye Dongyi. A Fast Scalable Attribute Reduction Algorithm. Pattern Recognition and Artificial Intelligence. 2009, 22(2):234-239.
[19] Feng Lin, Li Tianrui ,Yu Zhiqiang . Attributes Reduction Based on the Variable Precision Rough Set in Decision Tables Containing Continuous-valued Attributes. Computer Science. 2010,37(9):205-208.
[20] Feng Lin ,Wang Cuo-yin,Li Tian-rui. Knowledge Acquisition from Decision Tables Containing Continuous-Valued Attributes. Chinese Journal of Electronics, 2009 , 37 (11): 2432-2438.
[21] Dianhong Wang, Xingwen Liu, Liangxiao Jiang, Xiaoting Zhang, Yongguang Zhao. Rough Set Approach to Multivariate Decision Trees Inducing. Journal of Computers,7(4) (2012): 870—879.
[22] Jingling Yuan, Hongfu Du, Luo Zhong. Knowledge Reduction Algorithm based on Relative Conditional Partition Granularity. IEEE International Conference on Granular Computing 2010. GrC.2010. 2010:604-608.

Jingling Yuan, Dr. Yuan Jingling is an associate professor in Computer Science and Technology School at Wuhan University of Technology. She received a Ph.D. in Civil Engineering from Wuhan University of Technology. Her research interests include artificial intelligent, machine learning, multi-core architecture analysis and resource allocation.

Jing Xie, Miss Xie is a master student in Computer Science and Technology School at Wuhan University of Technology. Her research interests include intelligence algorithm and data mining.

Yan Yuan, Dr. Yuan Yan is an associate professor in School of Urban Design at Wuhan University. She received a Ph.D. from Tongji University. Her research interests include urban space, sustainable design of architecture and urban and BIM.

Lin Li, Dr. Li Lin is an associate professor in Computer Science and Technology School at Wuhan University of Technology. She received a Ph.D. from University of Tokyo. Her research interests include artificial intelligence and data mining, especially Web mining and information retrieval.