Parallel Rough Set Based Knowledge Acquisition Using MapReduce from Big Data

Junbo Zhang [†,‡]
JunboZhang86@163.com,
jbzhang@cs.gsu.edu

Tianrui Li [†]
trli@swjtu.edu.cn

Yi Pan [‡]
pan@cs.gsu.edu

[†] School of Information Science and Technology
Southwest Jiaotong University, Chengdu 610031, China
[‡] Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA

ABSTRACT

Nowadays, with the volume of data growing at an unprecedented rate, big data mining and knowledge discovery have become a new challenge. Rough set theory for knowledge acquisition has been successfully applied in data mining. The recently introduced MapReduce technique has received much attention from both scientific community and industry for its applicability in big data analysis. To mine knowledge from big data, we present parallel rough set based methods for knowledge acquisition using MapReduce in this paper. Comprehensive experimental evaluation on large data sets shows that the proposed parallel methods can effectively process big data.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: [Decision support]; D.2.7 [**Software Engineering**]: [Distribution]

General Terms

Algorithms, Design, Experimentation

Keywords

Big Data, Knowledge Acquisition, Rough Sets, MapReduce

1. INTRODUCTION

With the development of information technology, amount of data are collected from various sensors and devices in multiple formats. Such data processed by independent or connected applications will routinely cross the peta-scale threshold, which would in turn increase the computational requirements.

With the fast increase and update of big data in real-life applications, it brings a new challenge to quickly acquire the useful information with big data mining techniques.

BigMine '12, August 12, 2012 Beijing, China

For the purpose of processing big data, Google developed a software framework called MapReduce to support large distributed data sets on clusters of computers [2, 3], which is effective to analyze large amounts of data. As one of the most important cloud computing techniques, MapReduce has been a popular computing model for cloud computing platforms. Followed by Google's work, many implementations of MapReduce emerged and lots of traditional methods combined with MapReduce have been presented until now.

- *Implementations of MapReduce.*
  Apache Hadoop is a software framework that helps constructing the reliable, scalable, distributed systems [24]. Phoenix is a shared-memory implementation of Google's MapReduce model for data-intensive processing tasks [18]. Mars is a MapReduce framework on graphic processors (GPUs) [8]. Twister is a lightweight and Iterative MapReduce runtime system [4].

- *Traditional methods combined with MapReduce.*
  Apache Mahout can help to produce implementations of scalable machine-learning algorithms on Hadoop platform [25]. Menon et al. gave a rapid parallel genome indexing with MapReduce [15]. Blanas et al. proposed crucial implementation details of a number of well-known join strategies for log processing in MapReduce [1]. Ene et al. developed fast clustering algorithms using MapReduce with constant factor approximation guarantees [5]. Lin et al. presented three design patterns for efficient graph algorithms in MapReduce [13].

As one of data analysis techniques, rough sets based methods have been successfully applied in data mining and knowledge discovery during last decades [6, 16, 23], and particularly useful for rule acquisition [11, 19, 12] and feature selection [9, 10, 17].

To our knowledge, most of the traditional algorithms based on rough sets are the sequential algorithms and corresponding tools only run on a single computer to deal with small data sets. To expand the applications of rough sets in the field of data mining and knowledge discovery from big data, we discuss about rough set based parallel methods for knowledge acquisition in this paper. Based on MapReduce, we design corresponding parallel algorithms for knowledge acquisition on the basis of the characteristics of the data. The proposed algorithm is implemented on Hadoop platform [24]. Comprehensive experiments are conducted to eval-

uate the proposed algorithms and the results demonstrate that our algorithms can effectively process large scale data sets.

The paper is organized as follows. Section 2 includes a background introduction to MapReduce and rough sets. Rough set based methods for knowledge acquisition with MapReduce are presented in Section 3. Experimental analysis is given in Section 4. The paper ends with conclusions in Section 5.

2. PRELIMINARIES

In this section we will review MapReduce technique [2, 3] and some basic concepts of rough sets and knowledge acquisition [14, 16, 19, 22].

2.1 MapReduce Programming Model

MapReduce is a programming model, which is described as follows [2, 3].

The computation takes a set of input *key/value* pairs, and produces a set of output *key/value* pairs. The user of the MapReduce library expresses the computation by means of two necessary functions Map and Reduce, and one optional function Combine.

- **Map** takes an input pair and produces a set of intermediate *key/value* pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and transforms them with the Combine/Reduce function.
- **Combine (optional)** is local Reduce. It accepts a key I and a set of values for that key from local Map. Then, it merges together these values to form a possibly smaller set of values and transforms them with the Reduce function.
- **Reduce** accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically, just zero or one output value is produced per Reduce invocation.

2.2 Rough Sets and Knowledge Acquisition

Given a pair $K = (U, R)$, where U is a non-empty finite set called the universe, and $R \subseteq U \times U$ is an equivalence on U. The pair $K = (U, R)$ is called an approximation space. The equivalence relation R partitions the set U into several disjoint subsets. This partition of the universe forms a quotient set induced by R, denoted by U/R. If two elements, $x, y \in U$, are indistinguishable under R, we say x and y belong to the same equivalence class. The equivalence class including x is denoted by $[x]_R$.

An approximation space $K = (U, R)$ is characterized by an information system $S = (U, A, V, f)$, where

$$\begin{cases} U & \text{is a non-empty finite set of objects, called a universe.} \\ A & \text{is a non-empty finite set of attributes (features).} \\ V & \text{equals to } \bigcup_{a \in A} V_a, V_a \text{ is a domain of the attribute } a. \\ f & \text{is an information function } U \times A \to V, \\ & \text{such that } f(x, a) \in V_a \text{ for every } x \in U, a \in A. \end{cases}$$

Specifically, $S = (U, A, V, f)$ is called a decision table if $A = C \cup D$, where C is a set of condition attributes and D is a decision, $C \cap D = \emptyset$.

Definition 1. Let $B = \{b_1, b_2, \cdots, b_l\} \subseteq C$ be a subset of condition attributes. The information set with respect to B for any object $x \in U$ can be denoted by the tuple

$$\overrightarrow{x}_B = \langle f(x, b_1), f(x, b_2), \cdots, f(x, b_l) \rangle \quad (1)$$

An equivalence relation with respect to B called the indiscernibility relation, denoted by $IND(B)$, is defined as

$$IND(B) = \left\{ (x, y) | (x, y) \in U \times U, \overrightarrow{x}_B = \overrightarrow{y}_B \right\} \quad (2)$$

Two objects x, y satisfying the relation $IND(B)$ are indiscernible by attributes from B.

The equivalence relation $IND(B)$ partitions U into some equivalence classes given by:

$$U/IND(B) = \{[x]_B | x \in U\} \quad (3)$$

where $[x]_B$ denotes the equivalence class determined by x with respect to B, $[x]_B = \{y \in U | (x, y) \in IND(B)\}$. For simplicity, $U/IND(B)$ will be denoted by U/B.

Example 1. A decision table $S = (U, A, V, f)$ is given in Table 1, where Headache, Temperature are the condition attributes and Flu is the decision.

Let $B = \{\text{Headache}, \text{Temperature}\}$. We compute the equivalence classes. According to Definition 1, we have $U/B = \{E_1, E_2, E_3, E_4, E_5\}$, where

$$\begin{cases} E_1 = [x_1]_B = [x_5]_B = [x_7]_B = [x_{11}]_B = \{x_1, x_5, x_7, x_{11}\} \\ E_2 = [x_2]_B = [x_8]_B = \{x_2, x_8\} \\ E_3 = [x_3]_B = \{x_3\} \\ E_4 = [x_4]_B = [x_{10}]_B = \{x_4, x_{10}\} \\ E_5 = [x_6]_B = [x_9]_B = [x_{12}]_B = \{x_6, x_9, x_{12}\} \end{cases}$$

Definition 2. Let $B_1 = \{b_{11}, \cdots, b_{1l_1}\}$, $B_2 = \{b_{21}, \cdots, b_{2l_2}\}$ be two attribute sets, where $B_1 \cap B_2 = \emptyset$. The information set with respect to $B = B_1 \cup B_2$ for any object $x \in U$ can be denoted by

$$\begin{aligned} \overrightarrow{x}_B &= \overrightarrow{x}_{B_1 \cup B_2} \\ &= \overrightarrow{x}_{B_1} \wedge \overrightarrow{x}_{B_2} \\ &= \langle f(x, b_{11}), \cdots, f(x, b_{1l_1}) \rangle \\ &\quad \wedge \langle f(x, b_{21}), \cdots, f(x, b_{2l_2}) \rangle \\ &= \langle f(x, b_{11}), \cdots, f(x, b_{1l_1}), f(x, b_{21}), \cdots, f(x, b_{2l_2}) \rangle. \end{aligned}$$

Example 2. (Example 1 continued) In Table 1, the information set for the object of x_1 with respect to B, D and $B \cup D$ are shown as follows.

$$\begin{cases} \overrightarrow{x_1}_B = \langle no, normal \rangle \\ \overrightarrow{x_1}_D = \langle no \rangle \\ \overrightarrow{x_1}_{B \cup D} = \langle no, normal, no \rangle. \end{cases}$$

Definition 3. Let $B \subseteq A$ be a subset of attributes. The information set with respect to B for any $E \in U/B$ is denoted by

$$\overrightarrow{E}_B = \overrightarrow{x}_B, x \in E \quad (4)$$

Example 3. (Example 1 continued) According to Example 1, $E_2 = [x_2]_B = [x_8]_B = \{x_2, x_8\}$, hence, $\overrightarrow{E_2}_B = \overrightarrow{x_2}_B = \overrightarrow{x_8}_B = \langle no, high \rangle$.

Table 1: A decision table S

U	Headache	Temperature	Flu
x_1	no	normal	no
x_2	no	high	yes
x_3	yes	normal	yes
x_4	yes	veryhigh	yes
x_5	no	normal	no
x_6	yes	high	no
x_7	no	normal	no
x_8	no	high	yes
x_9	yes	high	yes
x_{10}	yes	veryhigh	yes
x_{11}	no	normal	no
x_{12}	yes	high	yes

In the earlier studies, Wong et al. used the confidence and resolution factor for inductive learning [20]. Tsumoto proposed the accuracy and coverage to measure the degree of sufficiency and necessity, respectively, and acquired classification rules with high accuracy and high coverage [19]. Han et al. introduced the rule support and rule confidence to measure the degree to which a rule is interesting rule in the field of data mining [7]. Liu et al. used the three parameters: support, accuracy and coverage to discover the interesting knowledge [14].

Given a decision information system $S = (U, C \cup D, V, f)$ where $B = \{b_1, b_2, \cdots, b_l\} \subseteq C$ and $D = \{d\}$. $\forall x \in U$, we generate a decision rule in the following way. The conjunction $\bigwedge_{b \in B}(b = f(x, b))$ and the decision attribute d with value $f(x, d)$ are taken as the predecessor and successor of a decision rule, respectively. Therefore, the constructed decision rule for the object x is of the form: $\bigwedge_{b \in B}(b = f(x, b)) \rightarrow d = f(x, d)$.

Definition 4. [14] Let $U/B = \{E_1, E_2, \cdots, E_M\}$ be a partition of condition attributes and $U/D = \{D_1, D_2, \cdots, D_N\}$ be a partition of decision attributes. $\forall E_i \in U/C$, $\forall D_j \in U/D$, the support, accuracy and coverage of $E_i \rightarrow D_j$ are defined respectively as follows:

Support of $E_i \rightarrow D_j$: $Supp(D_j|E_i) = |E_i \cap D_j|$;

Accuracy of $E_i \rightarrow D_j$: $Acc(D_j|E_i) = \frac{|E_i \cap D_j|}{|E_i|}$;

Coverage of $E_i \rightarrow D_j$: $Cov(D_j|E_i) = \frac{|E_i \cap D_j|}{|D_j|}$.

where $|\bullet|$ denotes the cardinality of the set.

Remark: It is easy to notice that $E_i \cap D_j \in U/C \cup D$. We call $E_i \cap D_j$ a union class.

Next, we will give three rule acquisition methods that all use the accuracy and the coverage.

Definition 5. [11] (Method 1) $\forall E_i$ $(i = 1, 2, \cdots, M)$, $\forall D_j$ $(j = 1, 2, \cdots, N)$, if $Acc(D_j|E_i) = 1$ holds, we call the rule $E_i \rightarrow D_j$ a consistent rule with the coverage $Cov(D_j|E_i)$.

Definition 6. [14, 19] (Method 2) $\forall E_i$ $(i = 1, 2, \cdots, M)$, $\forall D_j$ $(j = 1, 2, \cdots, N)$, if $Acc(D_j|E_i) \geq \alpha$ and $Cov(D_j|E_i) \geq \beta$ hold, we call the rule $E_i \rightarrow D_j$ a probabilistic rule where $\alpha \in (0.5, 1)$ and $\beta \in (0, 1)$.

Definition 7. (Method 3) $\forall E_i$ $(i = 1, 2, \cdots, M)$, $\forall D_j$ $(j = 1, 2, \cdots, N)$, if $Acc(D_j|E_i) = \max_{k=1,2,\cdots,N}\{Acc(D_k|E_i)\}$ $\geq \alpha'$ and $Cov(D_j|E_i) \geq \beta'$ hold, we call the rule $E_i \rightarrow D_j$ a max-accuracy probabilistic rule where $\alpha' \in (0, 1)$ and $\beta' \in (0, 1)$.

3. KNOWLEDGE ACQUISITION FROM BIG DATA BASED ON MAPREDUCE

In this section, we give three parallel methods for knowledge acquisition based on rough set theory using MapReduce.

Definition 8. [22] Given a decision table $S = (U, C \cup D, V, f)$. Let $S = \bigcup_{i=1}^{m} S_i$, where $S_i = (U_i, C \cup D, V, f)$. Suppose (1) $U = \bigcup_{i=1}^{m} U_i$; (2) $U_j \cap U_k = \emptyset$, $\forall j, k \in \{1, 2, \cdots, m\}$ and $j \neq k$. This means the decision table S is divided into m decision sub-tables. Then we call S_i is a decision sub-table of S.

Lemma 1. [22] Let $B \subseteq C$ be a subset of attributes, E, F be two equivalence classes with respect to B from two different sub-tables of S. One of the following results holds:

(1) If $\overrightarrow{E_B} = \overrightarrow{F_B}$, then these two equivalence classes E and F can be combined as one equivalence G with respect to B, where $G = E \cup F$ and $\overrightarrow{G_B} = \overrightarrow{E_B} = \overrightarrow{F_B}$;

(2) If $\overrightarrow{E_B} \neq \overrightarrow{F_B}$, then these two equivalence classes E and F cannot be combined as one equivalence with respect to B.

Zhang et al. proposed a parallel strategy to compute equivalence classes and decision classes using MapReduce [22]. Here, we give the following corollary.

Corollary 1. Given a decision table $S = (U, C \cup D, V, f)$. Let $S = \bigcup_{i=1}^{m} S_i$, where $S_i = (U_i, C \cup D, V, f)$. For any subset $B \subseteq C$, $U/B = \{E_1, E_2, \cdots, E_t\}$ and $\forall i \in \{1, 2, \cdots, m\}$, $U_i/B = \{E_{i1}, E_{i2}, \cdots, E_{ip_i}\}$. Let $E_{all} = \bigcup_{i=1}^{m} U_i/B = \{E_{11}, E_{12}, \cdots, E_{1p_1}, \cdots, E_{m1}, E_{m2}, \cdots, E_{mp_m}\}$. Therefore, for any $E_j \in U/B$, $j \in \{1, 2, \cdots, t\}$, we have

$$|E_j| = \sum_{F \in E_{all},\ \overrightarrow{F}_B = \overrightarrow{E_j}_B} |F| \tag{5}$$

According to Corollary 1, each sub-decision table can compute numbers of elements in equivalence classes, decision classes and union classes independently. At the same time, the classes of different sub-decision tables can combine together if their information sets are the same. Hence, it could be changed to a MapReduce problem and we design three parallel methods based on rough set theory for knowledge acquisition according to Definitions 5-7. Figure 1 shows our parallel models for three kinds of knowledge acquisition methods, which all contain two steps.

- **Step 1:** By the above analysis, the computation of the cardinality of the equivalence classes $|E|$, the cardinality of decision classes $|D|$ and the cardinality of union classes $|E \cap D|$ can be executed in parallel. In detail, we present the corresponding algorithms based on MapReduce/Combine, which are outlined in Algorithms 1, 2 and 3, respectively.

- **Step 2:** After computing the cardinalities of the equivalence classes, decision classes and union classes, by Definition 4, the accuracy $Acc(D|E)$ and the coverage $Cov(D|E)$ are computed. Then, three kinds of rule sets are generated according to Definitions 5-7, respectively.

Remark: Step 1 of these three methods are same and are executed in parallel. Step 2 of these three methods due to Definitions 5-7 and are executed in sequential.

Algorithm 1: Map($key, value$)

Input:
//*key*: document name
//*value*: $S_i = \{U_i, C \cup D, V, f\}$
//*Global variable*: $B \subseteq C$
Output:
//*key'*: the information set of the object with respect to the sets B, D and $B \cup D$
//*value'*: the count

```
1  begin
2      for each x ∈ U_i do
3          let key' = 'E' + x→_B; //Here, 'E' is flag, which means the equivalence class
4          output.collect(key', 1);
5          let key' = 'D' + x→_D; //Here, 'D' is flag, which means the decision class
6          output.collect(key', 1);
7          let key' = 'F' + x→_{B∪D}; //Here, 'F' is flag, which means the association between the equivalence class and decision class
8          output.collect(key', 1);
9      end
10 end
```

Algorithm 2: Combine($key,\ V$)

Input:
//*key:* the information set of the object with respect to the sets B, D and $B \cup D$
//V: a list of counts
Output:
//*key'*: the information set of the object with respect to the sets B, D and $B \cup D$
//*value'*: the count.

```
1  begin
2      let value' = 0 and key' = key;
3      for each v ∈ V do
4          value' = value' + v;
5      end
6      output.collect(key', value');
7  end
```

Example 4. In Table 1, let $S_1 = (U_1, C \cup D, V, f)$ and $S_2 = (U_2, C \cup D, V, f)$ be two sub-decision tables of S, $S = S_1 \cup S_2$, where $U_1 = \{x_1, x_2, \cdots, x_6\}$ and $U_1 = \{x_7, x_8, \cdots, x_{12}\}$. We set thresholds $\alpha = 0.6$, $\beta = 0.2$ for Method 2 and $\alpha' = 0.1$, $\beta' = 0.1$ for Method 3.

Table 2 shows Map, Combine and Reduce phases in Step 1 of three parallel methods. Then, the accuracy and the coverage of rules are computed and rule sets are generated as shown in Table 3, where $\surd$ and $\times$ means it is a rule or not, respectively. Obviously, the number of rules in sets are 4, 4, 5 for these three methods.

Algorithm 3: Reduce($key,\ V$)

Input:
//*key:* the information set of the object with respect to the sets B, D and $B \cup D$
//V: a list of counts
Output:
//*key'*: the information set of the object with respect to the sets B, D and $B \cup D$
//*value'*: the count.

```
1  begin
2      let value' = 0 and key' = key;
3      for each v ∈ V do
4          value' = value' + v;
5      end
6      output.collect(key', value');
7  end
```

4. EXPERIMENTAL ANALYSIS

In this section, we only evaluate the performance of the proposed parallel methods but not the accuracy since the parallel methods produce the same results as those of the sequential methods. All experiments run on the Apache Hadoop platform [24]. Hadoop version 1.0.1 and Java 1.6.0.12 are used as MapReduce system.

We utilize the large data set KDD99 from the machine learning data repository, University of California at Irvine [26], which consists of approximately five million records. Each records consists of 1 decision attribute and 41 condition attributes, where 6 are categorical and 35 are numeric. Since our method can only deal with categorical attributes, we discretize the 35 numeric attributes firstly. In addition, three synthetic data sets have been generated by means of the WEKA data generator [27]. The data sets are outlined in Table 4, where HDFS means Hadoop distributed file system [24]. The data sets KDD99, Weka-1.8G, Weka-3.2G and Weka-6.4G are split into 64, 64, 128 and 256 blocks, respectively when we upload these data to HDFS.

Our experiments are conducted on large clusters of compute machines. In detail, the task nodes consist of two kinds of machines. One kind of machines have 16 GB main memory and use AMD Opteron Processor 2376 with 2 Quad-Core CPUs (8 cores in all, each has a clock frequency of 2.3 GHz). The other kind of machines have 8 GB main memory and use Intel Xeon CPU E5410, comprising two Quad-Core CPUs (8 cores in all, each has a clock frequency of 2.33 GHz). The operating system in these machines was Linux CentOS 5.2 Kernel 2.6.18. Here, we compared results using 1,2,4,8,16 and 32 cores.

We executed the aforementioned three kinds of knowledge acquisition methods on Hadoop MapReduce system. Figure 2 shows the computational times of Step 1, Step 2 and total over the three parallel methods on 32 cores. Here, Methods 1-3 correspond to Definitions 5-7. It shows that Step 2 only costs very little time. Moreover, three parallel methods almost cost the same time in Step 1, Step 2 and total, respectively. Because of that, we merely discuss

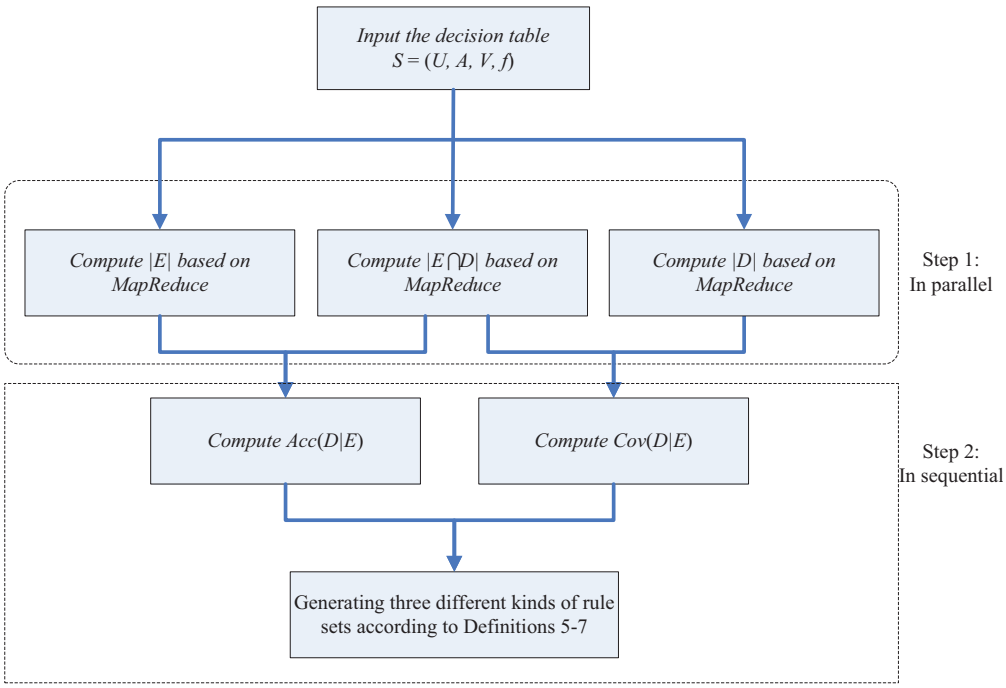

Figure 1: Parallel methods for knowledge acquisition

Table 2: Step 1 - Map, Combine, Reduce

	S_1			S_2		
	$\|E\|$	$\|D\|$	$\|E \cap D\|$	$\|E\|$	$\|D\|$	$\|E \cap D\|$
Map	$\langle no, normal\rangle$,1 $\langle no, high\rangle$,1 $\langle yes, normal\rangle$,1 $\langle yes, veryhigh\rangle$,1 $\langle no, normal\rangle$,1 $\langle yes, high\rangle$,1	$\langle no\rangle$,1 $\langle yes\rangle$,1 $\langle yes\rangle$,1 $\langle yes\rangle$,1 $\langle no\rangle$,1 $\langle no\rangle$,1	$\langle no, normal, no\rangle$,1 $\langle no, high, yes\rangle$,1 $\langle yes, normal, yes\rangle$,1 $\langle yes, veryhigh, yes\rangle$,1 $\langle no, normal, no\rangle$,1 $\langle yes, high, no\rangle$,1	$\langle no, normal\rangle$,1 $\langle no, high\rangle$,1 $\langle yes, high\rangle$,1 $\langle yes, veryhigh\rangle$,1 $\langle no, normal\rangle$,1 $\langle yes, high\rangle$,1	$\langle no\rangle$,1 $\langle yes\rangle$,1 $\langle yes\rangle$,1 $\langle yes\rangle$,1 $\langle no\rangle$,1 $\langle yes\rangle$,1	$\langle no, normal, no\rangle$,1 $\langle no, high, yes\rangle$,1 $\langle yes, high, yes\rangle$,1 $\langle yes, veryhigh, yes\rangle$,1 $\langle no, normal, no\rangle$,1 $\langle yes, high, yes\rangle$,1
Combine	$\langle no, normal\rangle$,2 $\langle no, high\rangle$,1 $\langle yes, normal\rangle$,1 $\langle yes, veryhigh\rangle$,1 $\langle yes, high\rangle$,1	$\langle no\rangle$,3 $\langle yes\rangle$,3	$\langle no, normal, no\rangle$,2 $\langle no, high, yes\rangle$,1 $\langle yes, normal, yes\rangle$,1 $\langle yes, veryhigh, yes\rangle$,1 $\langle yes, high, no\rangle$,1	$\langle no, normal\rangle$,2 $\langle no, high\rangle$,1 $\langle yes, high\rangle$,2 $\langle yes, veryhigh\rangle$,1	$\langle no\rangle$,2 $\langle yes\rangle$,4	$\langle no, normal, no\rangle$,2 $\langle no, high, yes\rangle$,1 $\langle yes, high, yes\rangle$,2 $\langle yes, veryhigh, yes\rangle$,1
Reduce		$\|E\|$		$\|D\|$		$\|E \cap D\|$
		$\langle no, normal\rangle$,4 $\langle no, high\rangle$,2 $\langle yes, normal\rangle$,1 $\langle yes, veryhigh\rangle$,2 $\langle yes, high\rangle$,3		$\langle no\rangle$,5 $\langle yes\rangle$,7		$\langle no, normal, no\rangle$,4 $\langle no, high, yes\rangle$,2 $\langle yes, normal, yes\rangle$,1 $\langle yes, veryhigh, yes\rangle$,2 $\langle yes, high, no\rangle$,1 $\langle yes, high, yes\rangle$,2

Table 3: Step 2 - Accuracy, Coverage, Rules

$E \to D$	$Acc(D\|E)$	$Cov(D\|E)$	Method 1	Method 2	Method 3
$\langle no, normal\rangle \to \langle no\rangle$	$4/4 = 1.00$	$4/5 = 0.80$	$\surd$	$\surd$	$\surd$
$\langle no, high\rangle \to \langle yes\rangle$	$2/2 = 1.00$	$2/7 = 0.29$	$\surd$	$\surd$	$\surd$
$\langle yes, normal\rangle \to \langle yes\rangle$	$1/1 = 1.00$	$1/7 = 0.14$	$\surd$	$\times$	$\surd$
$\langle yes, veryhigh\rangle \to \langle yes\rangle$	$2/2 = 1.00$	$2/7 = 0.29$	$\surd$	$\surd$	$\surd$
$\langle yes, high\rangle \to \langle no\rangle$	$1/3 = 0.33$	$1/5 = 0.20$	$\times$	$\times$	$\times$
$\langle yes, high\rangle \to \langle yes\rangle$	$2/3 = 0.67$	$2/7 = 0.29$	$\times$	$\surd$	$\surd$

Table 4: A description of data sets

	Data sets	Samples	Features	Classes	Size	Number of Blocks in HDFS
1	KDD99	4,898,421	41	23	0.48 GB	64
2	Weka-1.8G	32,000,000	10	35	1.80 GB	64
3	Weka-3.2G	40,000,000	15	45	3.20 GB	128
4	Weka-6.4G	80,000,000	15	58	6.40 GB	256

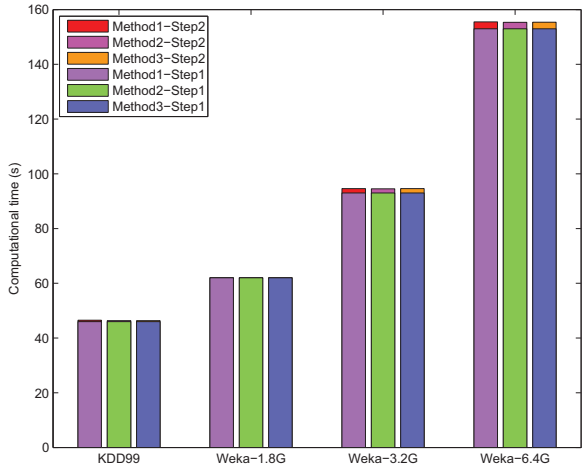

Figure 2: Computational time of 32 cores

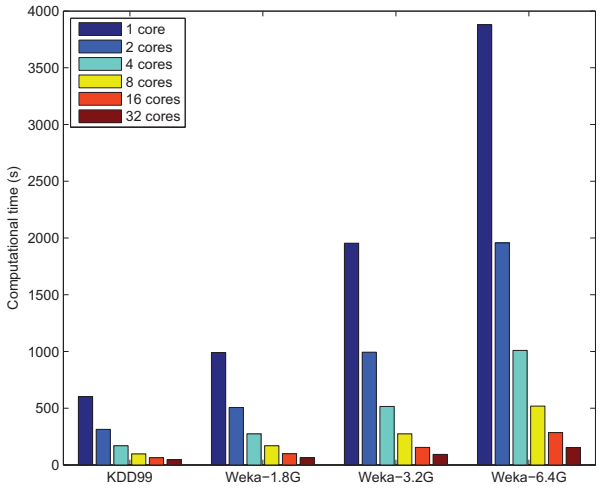

Figure 3: Computational times over different cores

about the performance of one method in the the following evaluations.

Figure 3 shows the computational times of the parallel method on different cores. As the number of the cores increases, the computational time of the parallel methods becomes smaller.

Further, we examine the speedup characteristic of the proposed parallel methods.

To measure the speedup, we keep the data set constant and increase the number of cores in the system. Speedup given by the larger system is defined by the following formula [21]:

$$Speedup(p) = \frac{T_1}{T_p}$$

where p is the number of cores, T_1 is the execution time on single core, T_p is the execution time on p cores.

The ideal parallel algorithm demonstrates linear speedup: a system with p times the number of cores yields a speedup of p. However, linear speedup is difficult to achieve because the communication cost increases with the increasing number of clusters.

We perform the speedup evaluation on data sets with quite different sizes and structures. The number of processes varied from 1 to 32. Table 5 and Figure 4 show the speedup for over all data sets for 2, 4, 8, 16 and 32 processing cores. As the result shows, the proposed parallel methods has a very good speedup performance. KDD99 has a lower speedup curve, because the size of KDD99 is too small. As the size of the data set increases, the speedup performs better. Therefore, the proposed parallel methods can treat big data efficiently. In fact, since our cluster is heterogeneous with two different kinds of machines, we run experiments on 1 core using the better machine and the proposed parallel methods would have a better speedup in a homogeneous setting.

Table 5: Speedups achieved on clusters

Data sets	number of cores				
	2	4	8	16	32
KDD99	1.923	3.562	6.206	9.556	13.087
Weka-1.8G	1.957	3.613	5.858	10.000	15.968
Weka-3.2G	1.968	3.794	7.131	12.688	21.011
Weka-6.4G	1.983	3.844	7.488	13.611	25.353
Average	1.958	3.703	6.671	11.464	18.855

5. CONCLUSIONS

Data mining from big data has been a new challenge in recent years. Traditional rough sets based methods for knowledge discovery fail to deal with the enlarging data in applications. In this paper, we proposed three rough set based methods for knowledge acquisition using MapReduce. We used speedup to evaluate the performances of the proposed parallel methods. Comprehensive experimental results on

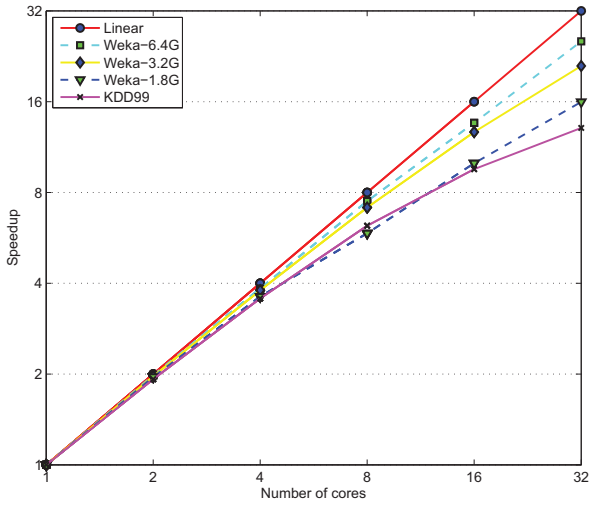

Figure 4: Speedup

the real and synthetic data sets demonstrated that the proposed methods could effectively process large data sets in data mining. Our future research work will focus on unstructured data processing by using rough set theory and MapReduce techniques.

6. ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation of China (Nos. 60873108, 61175047, 61100117), the Fundamental Research Funds for the Central Universities (No. SWJTU11ZT08), the Doctoral Innovation Foundation of Southwest Jiaotong University (No. 2012ZJB), and the Young Software Innovation Foundation of Sichuan Province (No. 2011-017), China.

7. REFERENCES

[1] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian. A comparison of join algorithms for log processing in mapreduce. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD'10, pages 975–986, New York, NY, USA, 2010. ACM.

[2] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1): 107–113, Jan. 2008.

[4] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC'10, pages 810–818, New York, NY, USA, 2010. ACM.

[5] A. Ene, S. Im, and B. Moseley. Fast clustering using mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD'11, pages 681–689, New York, NY, USA, 2011. ACM.

[6] J. W. Grzymala-Busse and W. Ziarko. Data mining and rough set theory. *Commun. ACM*, 43(4): 108–109, Apr. 2000.

[7] J. Han, M. Kamber, Data Mining: Concepts and Techniques, 2nd Edition, Morgan Kaufman, San Francisco, 2006.

[8] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: a mapreduce framework on graphics processors. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, PACT'08, pages 260–269, New York, NY, USA, 2008. ACM.

[9] Q. Hu, W. Pedrycz, D. Yu, and J. Lang. Selecting discrete and continuous features based on neighborhood decision error minimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(1): 137–150, feb. 2010.

[10] Q. Hu, Z. Xie, and D. Yu. Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. *Pattern Recognition*, 40(12): 3509–3521, Dec. 2007.

[11] K. Kaneiwa. A rough set approach to mining connections from information systems. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC'10, pages 990–996, New York, NY, USA, 2010. ACM.

[12] Y. Leung, W.-Z. Wu, and W.-X. Zhang. Knowledge acquisition in incomplete information systems: A rough set approach. *European Journal of Operational Research*, 168(1): 164–180, Jan. 2006.

[13] J. Lin and M. Schatz. Design patterns for efficient graph algorithms in mapreduce. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG'10, pages 78–85, New York, NY, USA, 2010. ACM.

[14] D. Liu, T. Li, D. Ruan, and J. Zhang. Incremental learning optimization on knowledge discovery in dynamic business intelligent systems. *Journal of Global Optimization*, 51: 325–344, 2011.

[15] R. K. Menon, G. P. Bhat, and M. C. Schatz. Rapid parallel genome indexing with mapreduce. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce'11, pages 51–58, New York, NY, USA, 2011. ACM.

[16] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko. Rough sets. *Commun. ACM*, 38(11): 88–95, Nov. 1995.

[17] Y. Qian, J. Liang, W. Pedrycz, and C. Dang. Positive approximation: An accelerator for attribute reduction in rough set theory. *Artificial Intelligence*, 174(9-10): 597–618, June 2010.

[18] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, HPCA'07, pages 13–24, Washington, DC, USA, 2007. IEEE

Computer Society.

[19] S. Tsumoto. Automated extraction of medical expert system rules from clinical databases based on rough set theory. *Information Sciences*, 112(1-4): 67–84, Dec. 1998.

[20] S. Wong, W. Ziarko, and R. Ye. Comparison of rough-set and statistical methods in inductive learning. *International Journal of Man-Machine Studies*, 25(1): 53–72, July 1986.

[21] X. Xu, J. Jäger, and H.-P. Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Min. Knowl. Discov.*, 3(3): 263–290, Sept. 1999.

[22] J. Zhang, T. Li, D. Ruan, Z. Gao, and C. Zhao. A parallel method for computing rough set approximations. *Information Sciences*, 194(0): 209–223, July 2012.

[23] W. Ziarko. Discovery through rough set theory. *Commun. ACM*, 42(11): 54–57, Nov. 1999.

[24] Hadoop: Open source implementation of MapReduce, $< http : //hadoop.apache.org/mapreduce/ >$.

[25] Mahout: Scalable machine learning and data mining, $< http : //mahout.apache.org/ >$

[26] KDDCup-99, $< http : //kdd.ics.uci.edu/databases/kddcup99/kddcup99.html >$.

[27] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update, *SIGKDD Explorations*, 11 (1): 10–18, 2009.