A STUDY ON NEW DATA MINING ALGORITHM BASED ON MAPREDUCE AND HADOOP

Melbert D’Cunha, Sanford Fernandes and Mithun Dsouza and S Sathyanarayana

Abstract - The goal of data mining is to determine concealed valuable information in large databases. Mining patterns from databases is one of the major issues in data mining. As the database size increases, the computational time and required memory increases too. Based on this, we proposed the MapReduce programming mode which consists of parallel processing to analyze the larger-scale network. All the experiments were done under hadoop, deployed on the cluster which consists of better servers. Through observed evaluations in the various simulation conditions, the proposed algorithms are shown to deliver better performance regarding scalability and execution time.

Keywords – MapReduce, Hadoop, Data mining, Newman algorithm.

I. INTRODUCTION

Business intelligence and data warehouse can manage TeraByte level data or even higher level. Although many methods have been put forward to deal with high-end data, but the query process is a bottleneck [1]. The introduction of cloud computing to the large amount of data mining, Hadoop is a MapReduce programming model and mass data [2]. It has performed a lot of simulation system in the cloud computing, such as computing based on the aspects of cloud modeling and simulation platform of COSIM-CSP system [3], a new way of the networked manufacturing [4], cloud framework for visual simulation [5], and the military training system [6]. A simple MapReduce is done by McCreadie on the Hadoop [7]. Ralf came up with a basic designed to support cloud computing [8]. Moretti introduced an efficient data mining method, the data and computation is distributed to a cloud.

1. Department of MCA, Aloysius Institute of Management & Information Technology (AIMIT), St. Aloysius College (Autonomous) Mangaluru, Karnataka - 575022
2. Department of MCA, Aloysius Institute of Management & Information Technology (AIMIT), St. Aloysius College (Autonomous) Mangaluru, Karnataka - 575022
3. Department of MCA, Aloysius Institute of Management & Information Technology (AIMIT), St. Aloysius College (Autonomous) Mangaluru, Karnataka - 575022
4. Department of MCA, Aloysius Institute of Management & Information Technology (AIMIT), St. Aloysius College (Autonomous) Mangaluru, Karnataka - 575022
Gillick introduced the inquiry learning with Hadoop[10]. Almost all of the data mining algorithms are based on object oriented programmings (OOP) that are usually run on a single system. However, the aspects of the MapReduce mode is not perfectly suitable for data mining.

In order to solve the limitations and problems, this paper uses a Newman parallel search algorithm based on MapReduce, to improve the processing velocity of large data. The test proves that the parallel algorithm is efficient to large data sets in this paper.

II. PROPOSED ALGORITHM

A. The Newman Algorithm with Modularity –

The fast Newman algorithm is a clustering algorithm based on greedy algorithm. Its steps are as follows:

| Step 1: Initialize the network for $N$ communities. It means that every point is an independent association. The initial element $e_{ij}$ and $a_i$ satisfies the following formula:
| $e_{ij} = \begin{cases} 1 & i=j \\ \frac{1}{2m} & i \neq j, (\text{connected}) \\ 0 & i \neq j, (\text{not connected}) \end{cases}$
| When the side is connected between $i$ and $j$, $e_{ij} = 1$. When the side is not connected between $i$ and $j$, $e_{ij} = 0$.
| $a_i = \frac{k_i}{m}$
| Where the $k_i$ is the degree of node; $m$ is the total number of edges in the network.

| Step 2: Merge the communities successively which is connected by the side, and calculation the increment of module.
| $\Delta Q = e_{ij} + e_{ji} - 2a_ia_j = 2(e_{ij} - a_ia_j)$

According to the principle of greedy algorithm, each time with the $Q$ should be increased most or reduce the minimum direction. After the merger by each time, we will update the corresponding elements of $e_{ij}$, and add the row and column which is correspond to community.

| Step 3: Repeat step 2, constantly consolidated community, until the $Q$ value is no longer increases. This has been the best network community structure.

Due to the time complexity of the algorithm is $O((m+n)n)$, so when the data quantity exceeds 10000, the required memory for $1 * 10^8$, then the memory overflow.
B. The Newman Parallel Algorithm with Modularity –

First initialize

MAP stage

Read the side information

\[ \text{\textit{Emit}}(<key_1, key_2>, value) \]

\(key_1\) and \(key_2\) is the Edge between vertices, \(value\) is the number of the edge.

Then, we will repeat the following procedure:

**Step 1:**

the edge sequence files will be converted into the degree sequence file

Map:

\[
\begin{align*}
& \text{if} \ (key_1 \neq key_2) \\
& \quad \text{Emit}(key_1, value) \\
& \text{else} \\
& \quad \text{Emit}(key_1, value) \text{ Emit}(key_2, value);
\end{align*}
\]

Reduce:

\[
\begin{align*}
& \text{sum} \ 0; \\
& \text{for} (\text{IntWritable value: values}) \text{ sum} \ \text{value.get} (); \\
& \text{SumValue.set(sum);} \\
& \text{Emit(key, SumValue);} \\
\end{align*}
\]

**Step 2:**

Calculated the value of \(Q\)

Map:

Read the edge sequential file

\[
\text{Emit(key_1, key_2, value, () , () )};
\]

Reduce:

Read the degree sequential file

\[
\text{Emit(key_1, key_2, value, () ); weight} \ 0; \\
\text{degree_e1} \ 0; \ \text{degree_e2} \ 0; \\
\ D \ Q \ 0; \\
\text{for(ValuePair value: values)}
\]
\{
    if(value.getFirst() == 0)
    weight = value.getFirst();
    else if(value.getSecond() == 0)
    {
        if(degree == 0)
            degree = value.getSecond();
        else
            degree = value.getThird();
    }
    / Q weighted_edge = (degree1_edge) * (degree2_edge): 2
\}
Emit(key1, key2, Q);

Step 3:
Find out the value of $\mathcal{Q}$ which has the most vertices
Reduce:

\[
\int max\ Value = 0
\]
\{
    while (value.hasNext())
    \{
        maxValue = Math.max(maxValue, values.next().get())
    }
    Emit(key1, key2, maxValue)
    if (maxValue = 0)
end
Repeat the Step 1 to Step 3, until the X value is no longer increases.

Figure 2. Parallel Data Processing Platform Based on the MapReduce

III. EXPERIMENT AND RESULT
The performance of parallel computing can be measured by its processing speed and scalability. [18-20]. The speedup is the performance improvement that is gained by the parallel computing to reduce the running time. It is an important index to verify the performance of parallel computing. The formula is $Sp = Ts / Tp$, where the $Ts$ denotes the
computation time of serial algorithm (i.e., in a single node), and \( T_p \) denotes calculation time of parallel algorithm (i.e., in the same \( p \) node). The acceleration is larger, the consumption of relative time of parallel computing is small, and the parallel efficiency and performance improvement is bigger.

We continuously give four tables and four graphs as below. They are the test results in the different number of nodes of the four algorithms.

**Table 1.** Run Times for the Newmans in MapReduce

<table>
<thead>
<tr>
<th>Unknowns</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>40000</th>
<th>60000</th>
<th>80000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>260</td>
<td>263</td>
<td>335</td>
<td>645</td>
<td>1955</td>
<td>3752</td>
<td>7561</td>
</tr>
<tr>
<td>2 nodes</td>
<td>256</td>
<td>260</td>
<td>287</td>
<td>505</td>
<td>1352</td>
<td>2489</td>
<td>4425</td>
</tr>
<tr>
<td>4 nodes</td>
<td>255</td>
<td>245</td>
<td>291</td>
<td>358</td>
<td>778</td>
<td>1403</td>
<td>2358</td>
</tr>
<tr>
<td>8 nodes</td>
<td>249</td>
<td>252</td>
<td>288</td>
<td>387</td>
<td>569</td>
<td>856</td>
<td>1562</td>
</tr>
<tr>
<td>16 nodes</td>
<td>229</td>
<td>238</td>
<td>268</td>
<td>300</td>
<td>358</td>
<td>568</td>
<td>954</td>
</tr>
</tbody>
</table>

**Table 2.** Run Times for the PAM in MapReduce

<table>
<thead>
<tr>
<th>Objects (thousand)</th>
<th>10000</th>
<th>25000</th>
<th>50000</th>
<th>750000</th>
<th>100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>1425</td>
<td>1466</td>
<td>2125</td>
<td>3785</td>
<td>7003</td>
</tr>
<tr>
<td>2 nodes</td>
<td>1154</td>
<td>1852</td>
<td>1995</td>
<td>2158</td>
<td>6251</td>
</tr>
<tr>
<td>4 nodes</td>
<td>811</td>
<td>793</td>
<td>1258</td>
<td>2368</td>
<td>2685</td>
</tr>
<tr>
<td>8 nodes</td>
<td>635</td>
<td>611</td>
<td>1325</td>
<td>1157</td>
<td>1774</td>
</tr>
<tr>
<td>16 nodes</td>
<td>635</td>
<td>512</td>
<td>427</td>
<td>788</td>
<td>1121</td>
</tr>
</tbody>
</table>

**Table 3.** Run Times for the CLARAI in MapReduce

<table>
<thead>
<tr>
<th>Objects (thousand)</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>125</td>
<td>126</td>
<td>138</td>
<td>177</td>
<td>275</td>
<td>882</td>
<td>1658</td>
</tr>
<tr>
<td>2 nodes</td>
<td>80</td>
<td>88</td>
<td>90</td>
<td>153</td>
<td>231</td>
<td>512</td>
<td>814</td>
</tr>
<tr>
<td>4 nodes</td>
<td>60</td>
<td>66</td>
<td>72</td>
<td>125</td>
<td>133</td>
<td>342</td>
<td>475</td>
</tr>
<tr>
<td>8 nodes</td>
<td>52</td>
<td>66</td>
<td>61</td>
<td>124</td>
<td>132</td>
<td>253</td>
<td>325</td>
</tr>
<tr>
<td>16 nodes</td>
<td>44</td>
<td>50</td>
<td>58</td>
<td>100</td>
<td>102</td>
<td>121</td>
<td>168</td>
</tr>
</tbody>
</table>

**Table 4.** Run Times for the CLARA2 in MapReduce

<table>
<thead>
<tr>
<th>Objects (thousand)</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>112</td>
<td>123</td>
<td>134</td>
<td>172</td>
<td>276</td>
<td>882</td>
<td>1658</td>
</tr>
<tr>
<td>2 nodes</td>
<td>80</td>
<td>88</td>
<td>95</td>
<td>153</td>
<td>235</td>
<td>518</td>
<td>817</td>
</tr>
<tr>
<td>4 nodes</td>
<td>60</td>
<td>66</td>
<td>72</td>
<td>124</td>
<td>134</td>
<td>343</td>
<td>476</td>
</tr>
<tr>
<td>8 nodes</td>
<td>53</td>
<td>65</td>
<td>67</td>
<td>125</td>
<td>132</td>
<td>253</td>
<td>324</td>
</tr>
<tr>
<td>16 nodes</td>
<td>45</td>
<td>51</td>
<td>57</td>
<td>98</td>
<td>99</td>
<td>115</td>
<td>151</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

Regardless of the issues encountered, all implemented algorithms were able to achieve speedup from using multiple nodes, as shown in the Figure 7, with Newman algorithm having the best and PAM the worst speedup in our tests. In this paper, by using the Newman algorithm for parallel study, and the algorithm compared with the other three algorithms. The test results show that the parallel algorithm of error rate is smaller, and it has very good validity. Especially, the dataset size is bigger, the efficiency will be higher.

REFERENCES


