Towards the Optimization of Data Mining Execution Process in Distributed Environments

Yan ZHANG1,†, Luoming MENG2, Feng LIU1, Honghui LI1, Peter BREZANY3

1 School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
2 State Key Lab of Networking & Switching, Beijing University of Posts and Telecommunications, Beijing, China
3 Institute of Scientific Computing, University of Vienna, Vienna, Austria

Abstract

The distribution and heterogeneity of data resources in loosely coupled distributed environments bring the challenges to data mining applications, because the parallelism and collaboration of nodes should be considered during the optimization and execution. Data mining algorithms can be represented as data mining execution processes which are composed of finer grained operators, and optimizing the execution processes according to the distribution of resources is critical for the execution performance and presents several algorithmic and systemic challenges. The work described in this paper offers such optimization functionality to transform the central data mining execution processes into distributed ones according to the distribution of resources in loosely coupled distributed environments, such as Grid or Clouds. First, data mining operators are analyzed and classified, and then optimization of data mining execution processes is divided into three phases in which data mining operators are processed according to their characteristics. The aim of optimization is to employ parallelism and collaboration to achieve the better performance. The evaluation results show that the optimization approach can reduce the execution times and use the resources in balance.

Keywords: Distributed Data Mining; Optimization; Data Mining Operators; Data Mining Execution Process; Parallelism

1. Introduction

The distribution and heterogeneity of large amount of data bring new challenges to data mining applications in loosely coupled distributed environments, such as Grid or Cloud. Data mining is a non-trivial process [1], large quantities of data and many complex operations are involved during the data mining execution, so it is a data- and computation-intensive task. Running data mining applications on the distributed environments makes data mining processes more complicate, because 1) the correct final result models can not be achieved by simply merging the data mining results generated on distributed nodes, 2) the parallelism of all nodes should be taken into account in order to reduce the execution times, and 3) all nodes should collaborate and work in load balance. So just focusing on data mining algorithms is not enough to solve these problems, it is inevitable that the execution of data mining processes should combine with underlying distributed infrastructure and should dynamically perceive the distribution of resources. The work presented in this paper is based on the decomposed data mining execution processes which were
discussed in our previous research [2], and an optimization algorithm is presented to dynamically transform the central data mining execution process into distributed ones that can be executed on the horizontal distributed environments.

In distributed database, a SQL query is rewritten as an execution plan which is composed of basic database operators, and then the execution plan is optimized according to data distribution [3]. The optimization of data mining execution processes is in analogy to the query optimization in distributed databases but with the following differences: 1) data mining operators are more complex and coarse-grained than the operators provided in database systems, 2) the optimization of data mining execution processes need to combine the environments’ information, such as load status and free resources, 3) more constrains should be considered beside data movement and execution time. Our research attempts to address the challenges in data mining execution process in loosely coupled distributed environments by designing an optimization algorithm.

The rest of the paper is structured as follows: Section 2 describes the data mining execution process. Section 3 represents the kernel part of this paper by describing the optimization algorithm of execution process in detail. Section 4 reviews the related work. In section 5 the evaluation results of Apriori and GSP are shown. We finish in Section 6 with conclusions and an outlook on future work.

2. Data Mining Execution Process

In this section, the previous work about data mining operators and execution processes will be reviewed, including the concepts, concrete operators, Apriori and GSP execution process, on which the optimization of next step is based.

Data mining operator is the function unit and many data mining operators can be assembled to form data mining execution processes. The granularity of data mining operators is finer than that of data mining algorithms. On the one hand, every operator can complete more complex tasks which can not be expressed by SQL statements, and on the other hand, operators are simpler and easier to understand than the whole data mining algorithms. Data mining operators are reusable and extendable.

Data mining execution process consists of a sequence of data mining operators and can describe the every step of the execution of data mining algorithms, and its effect is similar to that of the execution plan in database systems. Execution processes can be represented as a directed acyclic graph, namely operator graph, where operators are nodes and edges between nodes describe the relationships of consumer and producer. The execution process can be transformed as XML format file.

2.1. Data Mining Operators

The concrete operator algebra are described in [2]. According to the data requirements, data mining operators can be divided into three groups:

- Independent operators: the operator’s input only comes from the antecedent operator’s output;
- Local operators: the operator’s input comes from the antecedent operator’s output and the local data;
- Global operators: the operator’s input is the union of all distributed output of the antecedent operator, and it can generate global result.
Table 1 lists the data mining operators that are used in Apriori and GSP algorithms, including the description and group of the operators.

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataProcess η</td>
<td>retrieves the tuples from source data tables</td>
<td>independent</td>
</tr>
<tr>
<td>Cand1Patterns μ</td>
<td>generates all 1-patterns</td>
<td>independent</td>
</tr>
<tr>
<td>Count1Patterns ν</td>
<td>calculates the occurrence of every 1-pattern</td>
<td>independent</td>
</tr>
<tr>
<td>CandKItemsets ζ</td>
<td>generates K-itemsets by the self-join of (K-1)-itemsets</td>
<td>independent</td>
</tr>
<tr>
<td>CountKItemsets φ</td>
<td>calculates the support count of every itemset</td>
<td>local</td>
</tr>
<tr>
<td>Transformation β</td>
<td>transforms the itemsets into candidate itemsets</td>
<td>local</td>
</tr>
<tr>
<td>FreqPatterns γ</td>
<td>filters the data records that do not satisfy the formula</td>
<td>global</td>
</tr>
<tr>
<td>ARGeneration α</td>
<td>generates qualified rules from all frequent itemsets</td>
<td>independent</td>
</tr>
<tr>
<td>CandKSeqs ξ</td>
<td>generates K-sequences by the self-join of (K-1)-sequences</td>
<td>independent</td>
</tr>
<tr>
<td>PruneSeqs ρ</td>
<td>deletes the sequences, at least one child sequence of which is non-frequent.</td>
<td>local</td>
</tr>
<tr>
<td>CountSeqs σ</td>
<td>calculates the support count of every sequence</td>
<td>local</td>
</tr>
<tr>
<td>MaximalSeqs ψ</td>
<td>filters the non-maximal frequent sequences</td>
<td>independent</td>
</tr>
</tbody>
</table>

2.2. Apriori Execution Process

Apriori algorithm is one of the most classical algorithms for association rules mining and it generates frequent itemsets level-wisely, and k-frequent itemsets are self-joined to generate (k+1)-candidate itemsets [4]. Figure 1 shows Apriori execution process. DataProcess selects the needed attributes from the transaction data sets; Cand1Patterns gets all 1-candidate itemsets from the transaction data sets; Count1Patterns counts the occurrence of 1-candidate itemsets and FreqPatterns deletes unqualified candidate itemsets; the next four operators generate 2-frequent itemsets, and the four operators are repeated until the generated L-frequent itemsets is empty; ARGeneration generates rules from all frequent itemsets.

![Fig.1 The Execution Process of Apriori Algorithm](image)

2.3. GSP Execution Process

GSP algorithm is used to generate frequent sequences and it adopts generate-prune-test to get frequent sequences level-wisely, and k-frequent sequences are used to generate (k+1)-candidate sequences [5]. Figure 2 depicts the execution process of GSP algorithm. Here, the first four operators reuses the four operators of Apriori; then the following CandKSeqs generates 2-candidate sequences; CountSeqs counts the occurrence of sequences and FreqPatterns deletes unqualified candidate sequences; the next four operators generate 3-frequent sequences, and the four operators are repeated until the generated L-frequent sequences is empty; at last, MaximalSeqs filters the non-maximal frequent sequences.
3. The Optimization Process

In the distributed environments, the aim of optimization is to transform the central data mining execution process into distributed ones which can be described with data mining operations and communication operations for transferring data between sites. During this process, many factors should be taken in account, such as the distribution of data, the processing capacity of nodes, etc. In our research, the problem of optimization can be divided into several subproblems: data localization, global optimization and local optimization. In figure 3, a generic layering scheme for optimization is shown where each layer solves a well-defined subproblem. The input is a central data mining execution process which is posed on global data view, meaning that data distribution is hidden. Three main layers are involved to map the central execution process into an optimized sequence of local operations, each acting on local data resources. The optimization process need to traversal the operators graph twice, data localization and global optimization are executed in the first traversal and local optimization is executed in the second. The complexity of optimization algorithm is $O(K \times N)$, where $K$ is the number of nodes and $N$ is the number of operators.

3.1. Data Localization

The main role of data localization is to localize the involved data using data distribution information. When data mining algorithms are decomposed as execution processes, the distribution of data are not taken into account. Data localization translates an execution process on global data view into an execution process expressed on physical partitions. Localization uses information stored in the partition schema. Since the data mining operations can be synchronously executed on nodes on which the physical data partitions are located, data localization can add the inter-node parallelism. For independent and local operators $\theta$, if global data view $D = D_1 \cup D_2 \cup ... \cup D_k$, then data localization can be defined as: $\theta(D) = \theta(D_1 \cup D_2 \cup ... \cup D_k) = \theta(D_1) \cup ... \cup \theta(D_k)$. But for global operators, the equation does not hold, and global operators will be processed in global optimization phase. When localizing data, the processing of present operator depends on its type and the previous operator’s type, algorithm 1 shows data localization.

* When the present operator and the previous operator are all independent or local operators, the present operator keeps the consistence with the previous operator, meaning that the two operators are executed on the same node, as shown line 4 in algorithm 1.
* When the present operator is independent or local, and the previous operator is global, the
communication operators should be used to transfer data from the node that global operator is located to the nodes which involve data that present operator need, as shown line 8 in algorithm 1.

3.2. Global Optimization

The aim of global optimization is to process global operators according to the distribution of intermediate results generated by the previous operators. Algorithm 2 shows the global optimization process.

- When the previous operator is independent or local operator, the communication operators should be used to get intermediate results from all distributed nodes, and then the global operator use the merged intermediate data as input, as shown line 3-9 in algorithm 2.

- When the previous operator is a global operator, the present global operator can connect with the previous operator directly and they can be executed on the same node in order to avoid data movement, as shown line 10 in algorithm 2.

3.3. Local Optimization

Local optimization aims to add intra parallelism on a single node in order to reduce the execution time, and it is performed by all the sites having data partitions involved in the execution process. Each sub-process execution at one node is optimized using the local schema of the node. The rule of local optimization is: when the data size is greater than a specified threshold, the data should be divided into several fragments on which data mining operators will execute, which is similar to distributed share memory environment. Algorithm 3 shows local optimization, we specify the fixed values for $THRESHOLD$ and the number of data fragments $M$.

3.4. Examples of Distributed Execution Process

Based on the central data mining execution processes shown in Fig. 1 and Fig. 2, the corresponding optimized distributed execution processes are depicted in Fig. 4 and Fig. 5, where FreqPatterns is the global operator and the local intermediate results generated by the previous operators should be transported to the
central node and then merged by union to form the global result. Other data mining operators are independent or local, so they can be executed at different nodes simultaneously.

Algorithm 2  GlobalOptimization(op_i)

Input: The Operator op_i, Output: OperatorsGraphOfEPM
1  op_j = InputAdjOperator(op_i) /*Get the previous operator that input op_i*/
2  WHILE op_j ≠ NULL DO
3    IF (TypeOf(op_j) = LOCALOP) OR (TypeOf(op_j) = INDEPOP) THEN
4      CreateOperator(op_union)
5      FOR j = 1 TO n DO
6        CreateOperator(op_comm);
7        SetFrom(op_comm) = S_j; SetTo(op_comm) = S_i; op_ij → op_comm → op_union;
8      END FOR
9      op_union → op_i
10     ELSE op_j → op_i END IF
11   END WHILE

Algorithm 3  LocalOptimization(OperatorsGraphOfEPM)

Input: OperatorsGraphOfEPM Output: OperatorsGraphOfEPM
1  FOR i = 1 TO v DO
2    FOR j = 1 TO n DO
3      IF card(op_{(i-1)j}) > THRESHOLD THEN
4        M = NumberOfFragments(card(op_{ij}))
5        IF (TypeOf(op_{ij}) = LOCALOP) OR (TypeOf(op_{ij}) = INDEPOP) THEN
6          IF IsOptimized(op_{(i-1)j}) THEN
7            op_{ij} = \bigcup_{k=1}^{M} op_{ijk}
8            FOR k = 1 TO M DO op_{(i-1)jk} → op_{ijk} END FOR
9          ELSE op_{ij} = \bigcup_{k=1}^{M} op_{ijk} END IF
10        ELSE op_{ij} = \bigcup_{k=1}^{M} op_{ijk} END IF
11      END IF
12    END FOR
13  END IF

4. Experiments and Evaluation

In this section, we evaluate the optimized distributed data mining execution processes against the central ones and the performance in time units are evaluated. The result shows that proposed optimization algorithm can significantly improve the performance of distributed data mining applications. The simulator used for the evaluation is built by extending data access and integration middle-ware OGSA-DAI. The cost model estimates the execution time of data mining execution process, by estimating the cost of every data mining operator instance separately, and this cost is composed of computation time at nodes and communication cost between nodes. Data is horizontally distributed at many PC nodes. The RAM of PC is 1G and the CPU clock rate is 1GHz. The OS is Fedora 9 and data is stored in MySQL5 databases.
OGSA-DAI-4.1 is deployed and the data mining operators are implemented as user-defined activities and are deployed on the basis of OGSA-DAI. All nodes are connected with internal network and the bandwidth is 100M/s. The data sets used for evaluating Apriori and GSP are generated according to the methods introduced in [4] and [5] respectively. The parameters are shown in table 2.

Two groups of experiments are designed for Apriori and GSP respectively in order to show how the optimized data mining execution process can achieve better performance. The experiments are following:

- The transaction data set is randomly distributed at $M$ ($M=1, 3, 5, 7, 9$) nodes, where $M=1$ means that data mining execution process is executed on the central environment and $M>1$ means the execution on distributed environments. For Apriori, three sets of transaction data are $T_{500k}$-$K_{10}$-$I_5$-$N_5$, $T_{1000k}$-$K_{10}$-$I_5$-$N_5$, and $T_{2000k}$-$K_{10}$-$I_7$-$N_5$, the minimal support is 0.2. Figure 6 shows the result. For GSP, three sets of transaction data are $T_{500k}$-$C_{50k}$-$K_{10}$-$S_5$-$N_5$, $T_{1000k}$-$C_{100k}$-$K_{10}$-$S_5$-$N_5$, and $T_{2000k}$-$C_{200k}$-$K_{10}$-$S_7$-$N_5$, the minimal support is 0.2. Figure 7 shows the result. From this experiment’s result, we can see the parallelism of inter-nodes can reduce the execution time, especially when data size is large. Moreover, if $u_k$ is the execution time on $k$ nodes, it often holds that $u_1 < ku_k$ because the global operator can not be executed in parallel like independent and local operators; when data size is small, $u_1 \approx u_k$ because data movement and collaboration between nodes occupy more cost than computation at nodes.

- This group experiments are made on the same transaction data set, and these data is distributed at $M$ ($M= 1, 3, 5$) nodes, the execution time for generating L-frequent patterns are recorded respectively. For Apriori, the data set is $T_{1000k}$-$K_{10}$-$I_5$-$N_5$, and $L = 1, ..., 9$, the results are shown in figure 8. For GSP, the data set is $T_{1000k}$-$C_{100k}$-$K_{10}$-$S_5$-$N_5$, and $L = 1, ..., 9$, the results are shown in figure 9. From this experiment, we can see when generating L-frequent itemsets, which length of pattern contributes more to the execution cost. We can see from figure 8 generation 5- and 6-frequent itemsets are more time consuming, because when generating data set, the value of $|I|$ is 5 and it is consistent with the Poisson distribution; in GSP, generating 5-frequent sequences is more time consuming.

5. Related Work

Grid [6] and Cloud [7], as two popular loose-coupled distributed architectures, receive many attentions in academia and industry and they can manage distributed resources and provide services to customers. At
present, many distributed applications are built on them. The work presented in [8] describes a system that offers data processing and optimization functionalities on top of Grids or Clouds, operators and dataflow are employed in the application, but how to deploy data mining process on this system is not discussed. A distributed data mining application was developed using Sector storage Cloud and Sphere compute Cloud, where the storage cloud provides storage services, while the compute cloud provides computation services [9]. MapReduce mechanism is programming model and an implementation for processing large-scale data sets by dividing data set into many small ones [10]. A Grid data mining system based on MapReduce paradigm of computing was presented in [11], in which MapReduce scheduling system is used in Grid in order to integrate large scale data mining. The researchers recognized that data mining would be far more effective if it can not interact with other components in execution environment, such as database systems. The work presented in [12] summaries processing alternatives, storage mechanisms, algorithms, data structures and optimizations that enable data mining on large data sets, and then focuses on SQL and MapReduce as two completing technologies for large scale processing. Different frameworks were proposed to support knowledge discovery process in a uniform manner. The 3W model and algebra use regions, dimensions and hierarchies to define a uniform framework and operators [13]. Geist proposed a framework based on constraint database concepts in which the overall KDD process is divided into three main steps, pre-processing, data mining and post-processing, and for each of these different operators were defined [14]. But these models all use coarse-grained data mining operators which make it impossible to optimize data mining process automatically.

6. Conclusion

Data mining applications in distributed environments are more complicated than in traditional central environments, because the correct final result models, the parallelism of nodes as well as collaboration between nodes should be taken into account. So it is inevitable that the optimization and execution of data mining processes should combine with underlying distributed infrastructures and should dynamically perceive the distribution of resources. In order to solve these problems we elaborated on the optimization algorithm to dynamically transform the central data mining execution process into distributed ones that can be executed on the horizontal distributed environments. Our contribution in this work is threefold: 1) The classification of finer-grained data mining operators according to their characteristics; 2) The optimization algorithm composed of data localization, global optimization and local optimization phases; 3) The evaluation of Apriori and GSP execution process in Grid environments.
The next step of research work will focus on designing and implementing the execution engine which can dynamically schedule the distributed data mining execution process in distributed environments.

Acknowledgement

This work has been supported by Project No.2006AA01A121 of the National High-Technology Research and Development Program of China and Project No.KKJB10029536 of the University Foundation.

References