Designing Parallel and Distributed Algorithms for Data Mining and Unification of Association Rule

Imran Qureshi 1, Kanchi Suresh 2, Mohammed Ali Shaik 3, G. Ramamurthy 4

1,2 Assoc.Prof. Guru Nanak Institute of Technology, Ibrahimpatnam, RR District, Hyderabad
3 Asst. Prof., Aurora’s Research and Technological Institute, Warangal, Telangana, India
4 Assoc.Prof at IIT, Hyderabad, Gachibowli

Abstract. With the continually-increasing accessibility of information many methods have been evolved for encoding and storing the information that simultaneously grows all the time. Many available information sources include traditional databases such as relational database, flat file system, parallel or distributed knowledge bases, simple or complex programs, object-oriented or object-based, text documents with distinct proprietary procedures out of which few are variant of a conventional procedure. We are proposing four extensible architectures that provide flexibility to various data sources for handling data mining operations by creating or mining association rules.

Keywords: Parallel, Distributed, Data mining.

1. Introduction

Day by day knowledge bases are increasing along with increase in the sources of streams of information are also budding due to which the major issue arose is how to combine the distributed or parallel heterogeneous data repositories, an adhoc solution given to this problem is providing the storage location in separate geographic locations and in distinct formats as the cost of storage devices is diminishing and storage capacities are increasing drastically. We can extract many useful rules from data mining technique which generates or produces many association rules or policies from numerous diverse data sources which tend to amalgamate the results into a knowledge base using both parallel and distributed environments. Data mining association rules must repair knowledge base incompleteness if any such exist then we can use the support of generation of Probabilities among association rules, creation of Experts Systems, generating Knowledge out of the associations generated and Inference (PESKI) which is our proposed framework that integrates expert system development by utilizing knowledge using Bayesian Knowledge Base (BKB) representation technique. To provide flexibility and an simplicity of understanding for implementing many representation schemes by creating target oriented data mining process that will mechanize the procedure of inevitably resolve incompleteness in a BKB with the implementation of a database miner tool. Conventionally association rules are used to find one or more associations or relations that tend to be lost from a BKB is known when we mine the data using a database miner tool without a human intervention and in this paper we illustrate barely one procedure of knowledge base because many studies have revealed that PESKI is based on the process of realizing the incorporated knowledge oriented system framework which is intended to bring together the implementation of natural language interface implementation with the description and explanation of knowledge towards a single and secure application. BKB is designed such a way that it maps with the algebraic fundamental association of how a single randomly selected variable is related to one another and figure 1 denotes or represents the solo portion of information from BKB format which is intended to correspond in the form of sets to illustrate the reality that the given weather is cloudy or not and we take into consideration that there is 75% chance of climatic conditions in Warangal to be dry and the PESKI approach will use this data that is shown abruptly for inferring towards the knowledge base which is being created from implementation of the system.
Designing Parallel and Distributed Architecture for Data Mining and Unification of Association Rule

Fig. 1. Sample BKB Relationship Representation

2. Background Study

We have proposed few algorithms and implemented them in academic environment and yielded the results by using WEKA 3.5 tool.

a. Parallel Data Mining

The main aim of Data mining is to find the meaningful patterns or rules in large datasets by combining the investigations being held from the areas such as machine learning, mathematical algebraic statistics which tend to yield high performance computing over the neural or artificial intelligent networks that are mainly based on the neurons and typical algorithms. Resource intensive is the main feature intended in most of the data mining tasks that operates on huge deposits of data being salvaged from massive data resources determining in terabytes or zeta bytes are now totally considered to be frequent in data mining that tend to create data mining tasks and applications too deliberate to work and too gigantic to be executed on a solo processor computer thus is considered to be consequently emerging want to extend efficient parallel and distributed data mining algorithms to easily run on a deterministic distributed system and we are providing solution space for the problem space based on the famous Apriori algorithm. The below figure 2 provides us with an illustration of mining various available frequent item sets through assumed min support of 35% as from a database of a shopping mall.

Fig. 2. Example database and frequent itemsets.

Parallel Data mining Algorithm. Most of the Parallel Association Rule Mining (PARM) is established on paralleling the most often used Apriori algorithm to iteratively create and examine the created itemsets from the span of 1 to k itemsets unless the specified frequent itemsets are identified and be able to classify into Count Distribution or Data Distribution or Candidate Distribution techniques where the Count Distribution technique tracks the strategy of parallel designated data which algebraically splits a given database into parallel or flat splits that are independently scrutinized to cross check the local counts of all newly generated candidate itemsets that are generated on each and every process at the end of each phase by merging all the local counts of processes to generate the global counts by this process all the frequent itemsets can be easily identified.
In the process of data Distribution strategy we try to completely utilize the cumulative main memory that is available in parallel machines by distributing separately both the available database and the produced candidate itemsets that are intended to be measured by each and every candidate item set uniquely and later on all the processes have to exchange the newly created database partitions throughout each process iteration for generating the global counts for each and every process that are related to almost every candidate item set. And the candidate Distribution strategy is implied on the partitions that are created for the candidate itemsets and randomly duplicates the data sets as a substitute of partition and swap over the database transactions by which the course of action can continue or progress autonomously.

Algorithm Used for parallel-memory multiprocessors:

Step 1: Start

Step 2: Partition the database horizontally and distribute among process: p_0, p_1, p_2 respectively.

Step 3: Process P_i scans its partition for generating the local count of each and every item separately.

Step 4: Assign constant K=2.

Step 5: Process P_i generates candidate K-itemsets from the extracted repeated item sets (K-1).

Step 6: Process P_i scrutinizes its partition to gather the local count of each and every candidate K-item set.

Step 7: Process P_i exchange and calculate global counts=sum (local counts); for all frequent candidate k-itemsets.

Step 8: k=k+1.

Step 9: if frequent itemsets>0 then, goto step 5.

Step 10: Stop

Experiments conducted on the parallel implementation of apriori algorithm are shown in the above Fig with countDistribution method that is intended to exhibit improved presentation and scalability that of the other dual approaches that are based on the distribution process which is simplified for distributed memory in multiprocessors.

Fig. 3. Mining itemsets in parallel by imposing 3 processes, infrequent itemsets in (c), (d), (e) are removed.
b. Distributed Shared Memory Algorithm for PESKI:
To utilize cumulative processing on the available components of parallel machines we suggest and demonstrate an algorithm that is intended to depict outstanding scalability by utilizing the vertical data layout procedure for converting a horizontal database transaction or operation into a vertical list of itemsets which are sorted by attribute such as name basing on which the ID’s are produced for all the transactions available in the buffer memory which includes the itemset along with the frequent K-itemsets that are intended to be organized into disjoint correspondence classes using regular k-1 prefixes by which in the next iteration candidate k+1 itemsets can be yielded or generated by combining two or more frequent itemsets from the same intended class that posses the minimum support of a candidate itemset which will be further be computed very basically by merging the transaction id lists of two or more component subsets by which the concept of parallelism is attained by isolating the mining tasks for distinct classes of itemsets along with the available processes. The classes which are same in nature with two frequent itemsets assigned to the processes along with the transaction ID lists that are being circulated among each process and then the process of performing data mining with all frequent itemsets generated through assigned similarity classes that are separately being scanned and intersected with the intended or generated local transaction ID lists which are further processed in the process.

Algorithm Used for distributed-memory multiprocessors:
Step 1: Start
Step 2: Partition the database horizontally and distribute among process: p0, p1, p2 respectively.
Step 3: Process Pi scans its assigned partition for the accumulated local count of the first and second itemset.
Step 4: Partition both first and second frequent itemsets into classes by producing the prefixes.
Step 5: Assign one class to only one process Pi.
Step 6: Transform partitions into Vertical transaction ID for each and every two frequent itemsets.
Step 7: class=exchange (local tid, global tid);
Step 8: Pi=class.
Step 9: Mine (Pi).
Step 10: if frequent_item_set (tid)> 0 goto Step 7.
Step 11: Stop

![Fig. 4. Mining frequent itemsets using tid and process: P1, P2, P3.](image)
In the above implementation of the algorithm we have taken same database as shown in figure 2 where the Step 1, Step 2 and Step 3 are intended to work in a same way as that of in the previous proposed algorithm and the variation is in Step 5 where the scheduling of the similar classes on the different processes is individually and cautiously developed towards the process of reduce the workload unevenness. We have designed the best simple approach for calculating the workload of every individual class and arrange the classes in a sorted order with the descending workload to the slightest weighted process for all the pairs of itemsets and then one of the equality class will be processed to mine bottom level itemsets depicted as S2 can be used for denoting the probable workload for a probable class of |S| itemsets and we can also schedule other mechanisms on tasks to be performed as once they are available for the processor. Steps 6 and 7 are used to generate the transaction ID lists for all the two frequent itemsets that are intended to execute in parallel because each process is proposed to scan only one horizontal partition of the existing database as intended and gets a fractional list of transaction ID’s for each item set and then the partial list of itemsets are assigned or merged with all the processes to produce the global transaction ID’s list that comprises of all the intended transactions in a database.

In many sample scenarios which we have taken into consideration is mostly the number of frequent dual itemsets that can be so huge for us to accumulate all their transaction ID list that tend to designate to be exceptionally costly in terms of both processing time and memory usage so we intend to construct the transaction ID’s based on the list of recurring items instead of selectively duplicating on all the processes by which each process has the transaction ID list of all the member items in the allocated alike classes which requires to generate the transaction ID list of a frequent dual item set in a dynamic approach by overlapping the transaction ID list of the dual element items. Our step 8 is intended to implement as the asynchronous part where each and every process attempt to mine the recurring itemsets without depending on each of the allocated class and mutually depends only on the local transaction ID lists by computing on each similar class that usually produces many child corresponding classes that are designated to process in a iterative fashion.

c. The model Growth Method:
The model growth method originates recurring itemsets precisely from the database by bifurcating the expensive creation and analysis of the large number of candidate itemsets and our Frequent pattern growth algorithm is formulated such that it utilizes the frequent pattern tree structure in which the iterative transactions are compressed into complex instructions and structured into the incidence ordered prefix tree structure such that it assign and distribute the general prefix section as much as possible and also performs the duplication of items or itemsets that are inevitably counted and navigated to mine all the frequent or iterative patterns also called as the frequent itemsets. The model based approach is also implemented in some researches as divide and govern policy using which we can reduce the mining task into a group of slighter subtasks for mining restricted models in the basis of the novel conditional pattern in which the process or each item is basically tiny database of measured models that transpire with the item and the database which is tend to be distorted into a qualified frequent pattern tree that can be practiced in a iterative fashion.

In general, the FP-growth algorithm can be modulated as:
Step 1: Start
Step 2: modeling an intended database uniformly into the horizontal with the processes P1,P2,P3.
Step 3: Scrutinize each database model Pi analogy and mine all the iterative items.
Step 4: Construct a local tree using process pi by using the confined database model to generate the global iterative items.
Step 5: Process pi generates conditional pattern for local FP-tree for all frequent items.
Step 6: \( pi = \sum_{i=1}^{n} \text{frequentItemset}_i \).
Step 7: All local conditional patterns that is intended to collect and convert into the stipulated FP-tree on a specific predefined process.
Step 8: All the process iteratively go across each of the allocated prerequisite of FP-tree to mine the iterative itemsets in the data items presence.
Step 9: Stop
Designing Parallel and Distributed Architecture for Data Mining and Unification of Association Rule

The above algorithm is implemented as a parallel algorithm that is classified broadly into two stages or sections where Step 1 to 3 is considered to be the first section which is intended to create multiple local FP-trees from a pre desired database transaction using the occurrence rate of transaction depicted in the local database model where each process is designed to be able to build its own dynamic FP-tree based on the global occurrence items that are pre selected and pre sorted by occurrence rate in descending order and then pumped in to the local FP-tree as follows. Navigating from head node to the tail node verification of the very initial item as one of the child node of the root node by raising the counter value for this node and insert a new child node under the root node for this specific item with 1 count up by considering the present node item as the new provisional root node to replicate the identical process for the preceding item in the variant transaction where the nodes of every item be inter linked with one another from root in the header table as shown in figure 1.5 depicts the parallel creation of the several local FP-trees on both the processes for example as shown in database of figure 1.2.

![Fig.5. Creation of two local FP-trees depicting local database models on 2 processes.](image)

3. Implementation

The concepts described in this paper are implemented in Microsoft.Net 2010 all the algorithms are operated user-interactively are shown below:

![Fig.6. Implementation Screen on page load.](image)
4. Conclusion and Future Scope

Due to availability of enormous gigantic volumes of data which is not in the reach of processing power of a single processor computer and to tackle such situation intervention of parallel data mining techniques grown to attain additional importance for data practitioners and business decision architect to dig out concise and observant knowledge composed of information in an adequate quantity of time. In this paper we have proposed four algorithms in parallel and distributed environments for generating parallel and distributed association rule mining and creation of clusters using the datasets by visiting through distributed and shared memory based systems. We have resolved the main issue of workload balancing because of the active nature of association rule mining where it uses static task scheduling mechanisms by focusing on minimizing the data dependence across processes in multiprocessor algorithms which are based on parallel computing environment. Still we have to work more on parallel environment where we have to implement the branch penalty for all the algorithms and we also have to work on cross cutting issues while generating association rules.

References