Architecture for Large Scale Reasoning in Business Intelligence

Xinlong Zhang
College of Computer Science
Zhejiang University
zhangxinlong2012@gmail.com

Bin Cao, Yanming Ye
College of Computer Science
Zhejiang University
{caobin, yeym}@zju.edu.cn

***Abstract*-Business intelligence plays a crucial role in modern business. Nevertheless, present business intelligence is not in a position to provide comprehensive business advices owing to limitations on the scope of data and satisfy the indispensable timeliness for business activities. To address these problems, we propose an architecture for business intelligence which could reason on data from numerous domains and provide different users with disparate business advices and results. Furthermore, in our architecture, the production system used to reason depends on MapReduce programming model to implement production rule matching concurrently in different computers with the Rete algorithm. Adopting MapReduce programming model enables production system to obtain more impressive efficiency in rule matching, especially when it comes to a large-scale rules and facts. What's more, we also adopt two conflict-resolving polices to decide in which sequence matched production rules are executed. In this paper, we firstly describe the architecture and then illustrate the particular implementation of this architecture.**

Keywords-Business intelligence, Production System, MapReduce, Timeliness

I. INTRODUCTION

For the purpose of companies' continuous development as well as eliminating the risk from company's daily operation, business leaders are supposed to make decisions based on the past information and previous experience. However, due to the rapidly growing number of business data, business leaders have been incapable of analyzing these huge data, thereby being unable to make accurate conclusions from them. Consequently, people begin to introduce business intelligence into the process of business decision-making which is first proposed by Gartner Group.

Business intelligence [1, 2] is mainly composed of two crucial steps. The first one is to extract useful data from companies' database and transform these data into useful knowledge by the means of data mining and OLAP[3]. The other one is to provide useful advices to business leaders via knowledge reasoning on the knowledge generated from step one. In order to implement knowledge reasoning, people need certain techniques of knowledge representation to transform knowledge into certain data structures which are used to describe knowledge and can be accepted by computers. Generally, there are four basic techniques for representing the acquired knowledge which are logical representation, semantic networks, frames and production rules. In business intelligence, the technique of production rules is frequently used to represent knowledge. Through transforming knowledge into production rules, people can complete the process of reasoning by the firing of facts and production rules in the production systems.

Nevertheless, there are two limitations on present business intelligence systems. First, most of present business intelligence systems only serve a hint of companies, thereby only containing knowledge confined to certain related domains. Second, attend with increasing number of business data and the production rules transformed from such data, the central production system employed in Business intelligence nowadays need to spend a lot of time on production rules matching and thus cannot satisfy timeliness which is essential to business intelligence.

Therefore, we would propose architecture for business intelligence which could reason on information from multiple domains to serve a great many firm. In the meanwhile, this architecture introduces MapReduce[4] programming model which is widely used to perform large-scale data in the distributed and parallel way as well as employs necessary conflict-resolving polices. Therefore, this architecture can enable production system to perform production rules matching concurrently in different computers as well as provide many corporate with more comprehensive results, thereby improving the efficiency in production rules matching and finally achieve better timeliness.

This paper is organized as follows. Section 2 describes some rerlated works. Then we will generalize the architecture we proposed in section 3 and discuss in details the implementation and prototype of this architecture in section 4. In section 5, we would make conclusion about this paper and point out several negative aspects that should be eliminated for achieving better job in the future.

II. RELATED WORK

So as to improve the efficiency in production rule matching in all kinds of intelligence system, numerous researches and experiments have been done. Forgy proposed the efficient algorithm of Rete[5] in1970s and then several improved algorithms[6,7,8] for Rete are proposed.But all these algorithms only implement reasoning on single computer and thus could not provide satisfactory speed in rule matching, so some scientists began to study how to implement rule matching and firing concurrently. In the term of parallel rule matching, C. Dou proposed a highly-parallel two layer match architecture [9] using specific associative matching processor (AMP) to speed up the time for rule matching of production system. Different from above architecture, we propose a MapReudce-based

Published by Atlantis Press, Paris, France.

architecture for production system which could implement rule matching in any number of computers, thereby can get better efficiency in rule matching. In the term of parallel rule firing, Ishida and Stolfo employed data dependency graph[10,11] to analyze the interdependency of two rules and ultimately proposed a selection algorithm for parallel rule matching. Moreover, Ching-Chi Hsu and Feng-Hsu Wang proposed an Object Pattern Matching model[12] to interpret the dependency between production rules and based on the rule dependency analysis they developed a new conflict resolution principle, Search Ahead Conflict Resolution(SACR), to implement parallel rule firing. Compared with them, the architecture we propose could fire rules concurrently through dividing firing results into certain categorizes.

III. ARCHITECTURE

The architecture (Figure 1) we proposed consists of user, master, worker, data mining, OLAP and data pool.

The working process of this architecture is made up of two parts. The first part is to collect information and transform them into production rules. Another one is to reason based on these production rules in the production system. Now we will discuss these two parts respectively.

A. The process of managing data

Compared with traditional business intelligence, our architecture could include information from multiple domains and thus has the capacity to provide more comprehensive results for users. For instance, a company engaged in transportation industry must consider numerous factors regarding many domains such as weather, the price of oil, political environment and so on. Therefore, due to the limitation on the necessary information, traditional business intelligence cannot offer this company such useful advices as which transport routes this company should choose. However, due to the inclusion of many domains, the business intelligence based on our architecture would definitely help this company with the route-choosing problem.

The key steps of this process are as follows:

1. System collects a myriad data from many domains such as meteorology, stock market, every industry and so on and stores these data in the data warehouse.
2. System searches the data in the warehouse for useful information through data mining and then employs OLAP to obtain knowledge from such useful information.
3. System transforms knowledge obtained in the last step into production rules and passes these production rules to production system for reasoning.

B. The process of reasoning

The process of reasoning in the MapReduce-based production system could be divided into five phases as follows:

1)Preparation stage:

In this phase, firstly masters would analyze production rules from multiple domains into sub-rules. Then these sub-rules would be distributed to multiple workers under the consideration of load balance.

2)Build phase:

In this phase, each worker would build received sub-rules into a Rete net respectively with the Rete algorithm.

3)Map Phase:

In this phase, the master would pass facts coming from multiple sources to workers with certain strategy firstly. Then facts on every worker would go through the accordingly Rete net one by one and workers would transmit fired sub-rules and corresponding facts to workers which implement reduce phase.

4)Reduce phase:

In this phase, workers would receive results generated from last phase and reduce them so as to find the production rules which can be fired. Then workers would return these fired production rules to master.

5)Conflict-resolving phase:

In this phase, master would insert all of fired production rules into agenda which could decide the execution order of these rules. After every production rule matching with certain fact is executed, master would transmit results to corresponding persons or organizations.

From the description above, we could know that this architecture would definitely improve the efficiency for rule matching especially in terms of large-scale production rules and could provide results for clients who expect them.

IV. IMPLEMENTATION AND PROTOTYPE

Before illustrating our architecture, we will give some definitions and notations.

Definition 1 (Production rule)

Production rule is made up of LHS and RHS. LHS is finite set of condition and RHS is a finite set of action or conclusion.

Definition 2 (Sub-rule):

Sub-rule is the rule which contains only one condition coming from specific production rule. Therefore, LHS of sub-rule possesses merely one condition. And RHS of sub-rule also only contains one action which could generate a mark indicating such sub-rule has been fired by certain fact.

Definition 3 (Sub-rule base):

After the production rules in the rule base is analyzed into sub-rules, the sub-rules of this rule base can be viewed as the following matrix:

$$MR_{(n,m,SR)} = \begin{pmatrix} SR_{<1,1>} & \cdots & SR_{<1,m>} \\ \vdots & \ddots & \vdots \\ SR_{<n,1>} & \cdots & SR_{<n,m>} \end{pmatrix}$$

In the matrix above,

1. n represents for the number of production rules.

2. m represents for the max number of sub-rules generated from all of production rules.

3. SR represents for sub-rule defined in the definition 2.

4. We use $SR_{<i,j>}$ to represent for the j sub-rule generated from i production rule, where $1 \le i \le m$ and $1 \le j \le n$.

5. If the number of sub-rule generated from certain production rule is less than m, then we could equate the rest sub-rule of such production rule with null.

Definition 4 (index):

We could denote certain sub-rule by index uniquely. Then $SR_{<r,s>}$ is the s sub-rule of r production rule, when $index =< r, s >$.

Now we would discuss the particular implementation and prototype of the production system in our architecture.

1) Transformation phase

At first, the system searches the data from the data warehouse for useful information through data mining and employs OLAP to obtain knowledge from such useful information. Then the system transforms knowledge obtained into production rules which would be passed to the master.

2) Preparation phase

At first, the master collects production rules coming from business intelligence and analyzes these production rules into sub-rules to constitute a sub-rule base. Then the master retrieves the load information of every worker in this distributed system and depends upon such load condition to distribute sub-rules to workers. In this process of distribution, besides load balance we should also take net transmission into account because the time for transmitting sub-rules may be quite huge comparatively.

3) Build phase

At first, every worker that have received sub-rules builds a Rete net respectively based on its own sub-rules with the Rete algorithm. After that, every worker sends a message to inform master that the Rete net is built. The Rete algorithm can provide impressive efficiency for rule matching, though is at the expense of some space costs.

4) Map phase

At first, the master receives facts from many sources such as sensors, web server, ERP system and stock market and passes such facts to all of workers implementing mapping in a queue. When facts arrive at workers, the process of rule matching begins. The pseudo code for Map function is showed in Figure 2:

```
Function Map ( $fact key, array($Sub-Rule) value)
{
    for every Sub-Rule sub-rule in value
    {
     if fact matches with this sub-rule;
     {
      Obtaining the rule generating this sub-rule and the index of this sub-rule;
       Store the pair of Key/value: $fact, array($rule, $index);
      }
    }
}
```

Figure 2. The pseudo code for Map function

As Figure 2 shows, this function receives facts in a queue and an array of sub-rule as arguments. Then this function implements the process of rule matching. When certain sub-rule is fired by fact, this fact would be stored as key as well as the rule generate such sub-rule and the index of such rule would be stored as value.

After Map phase, the pairs of key and value would be transmitted to other workers which implement reduce phase. In this process of transmission, every pair of key and value with the same key would be passed to the same worker

5) Reduce phase

The workers implementing reduce phase can be the workers implementing the map phase totally but workers would not implement reduce phase unless their map phases have been completed. The pseudo code for Reduce function is showed in Figure 3:

```
Function Reduce ($fact key, array ( $rule, $index) value)
{
    for every ($rule, $index) in array
    {
     obtaining rule's present annexing result ( $rule, $index') and annex index as ( $rule, $index' & $index);
        if ( all indexes rule possesses have been annexed)
            Store the pair of key and value: $fact, $rule;
    }
}
```

Figure 3. The pseudo code for Reduce function

As Figure 3 shows, reduce function receives the pairs of key and value generated from map phase as arguments. When all sub-rules that certain production rule possesses are fired by a fact, this fact would be stored as key as well as this production rule would be stored as value and this pair of key and value would be returned to the master.

6) Conflict-resolving phase

After the completion of rule matching, facts and rules which match successfully would be inserted into agenda. At this time, there are three conflicting situations: one fact matches with multiple production rules, one production rule is fired by multiple facts and multiple facts match with multiple production rules. Therefore, we need certain polices to decide which sequence we can adopt to execute the actions of production rules.

As the business intelligence based on our architecture could serve users from many industries, the results from

business intelligence could be categorized according to different industries.

In our architecture, we design two polices to resolve conflicts.

Parallel executing policy: When the execution results of certain production rules are not contained in the same category of industry, these production rules can be executed concurrently. This strategy is conducive to save time for rule executing and thus users can acquire their expected results timely.

Priority policy based on complexity of rules: When the execution results of certain production rules belong to the same category of industry, these production rules ought to be executed sequentially. At this time, we would choose the most complex production rule at present to execute firstly.

V.. Conclusion and Future Work

In this paper we presented the definitions and particular implementation of MapReudce-based architecture for production system which is designed for business intelligence. Our architecture is capable of reasoning on information from multiple domains for users of many fields such as transportation, IT and so forth and satisfying the timeliness required by business intelligence through implementing matching rules on different workers and certain conflict-resolving policies.

Even if this architecture can improve the efficiency for rule matching, it still suffers from several limitations expected solved in the future. First, we are supposed to improve the stability and reliability of this architecture by the means of checkpoints, redundancy policy and so forth. Moreover, we need to provide certain policy to ensure information security because production rules are spread among different workers. Anyway, we still have a lot of work to improve this architecture.

REFERENCES

[1] F. Cody, T. Kreulen, V. Krishna, S. Spangler. The integration of business intelligence and knowledge management. IBM System Journal, 2002, PP 697-713.

[2] P. Luhn. A business Intelligence System. IBM journal of Research and Development, 2010, PP 314-319.

[3] Surajit Chaudhuri, Palo Alto. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 1997.

[4] Jeffery Dean, Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. Communications of the ACM, V.51, N.1, 2008.

[5] Charlee L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, V. 19, N. 1, 1982, PP 17-37.

[6] Anurag Acharya, Milind Tambe. Collection oriented match: Scaling Up the Data in Production System. Proceedings of the second International Conference on Information and Knowledge Management, 1993, PP 516-526.

[7] Ian Wright, James Marshall. The execution kernel of RC++: RETE, a faster RETE with TREAT as a special case. International Journal of Intelligent Games and Simulation, 2003, PP 36-48.

[8] B. Berstel. Extending the RETE algorithm for eventmangement. Temporal Representation and Reasoning, 2002, PP 49-51.

[9] C. Dou. A highly-parallel match architecture for AI production system using application-specific associative matching processors. Application-Specific Array Processors, 1993, PP 180-183.

[10] Toru Ishida, S. Stolfo. Towards Parallel Execution of Rules in Production system Programs. ICPP-85,1985, PP 568-575.

[11] Toru Ishida. Parallel Firing of Production System Programs. IEEE Trans. KDE, V.3, N. 1, 1991, PP 11-17.

[12] Ching-Chi Hsu, Feng-Hsu Wang. The Search Ahead Conflict Resolution for Parallel Firing of Production Systems.

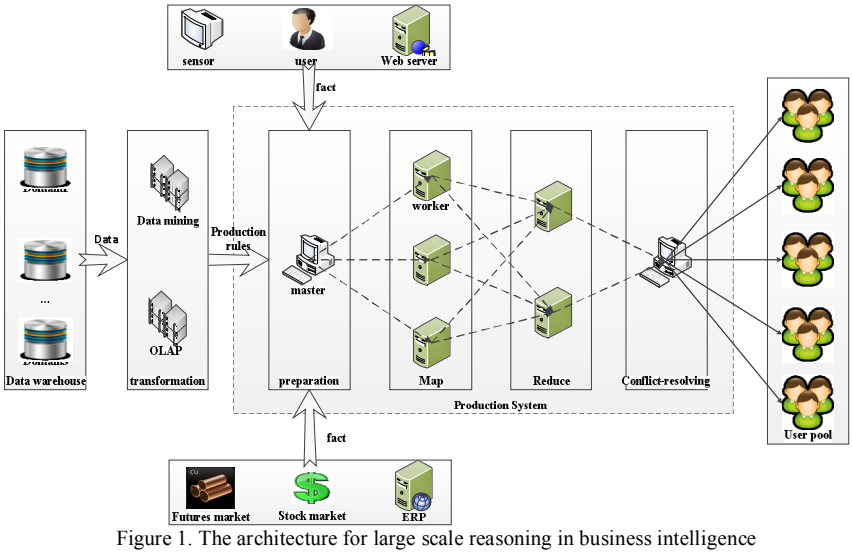

Figure 1. The architecture for large scale reasoning in business intelligence