A Method of Manufacturing Resource Negotiation Based on MapReduce

Zhi Yang[1, 2]

[1]Manufacture Information Research Center, Jiangsu University,
Zhenjiang 212013, Jiangsu, China
[2]School of Management, Jiangsu University,
Zhenjiang 212013, Jiangsu, China

Jinan Gu

Manufacture Information Research Center, Jiangsu University,
Zhenjiang 212013, Jiangsu, China

Abstract
With the development of modern manufacture, manufacturing resource becomes more and more complex, and the manufacturing resource provided by supply user may not fully meet the needs of demand user directly. Then the negotiation of manufacturing resource is needed to be performed between demand user and supply user. With the refinement of the manufacturing resource, its attributes are increasing, thus the solution spaces of negotiation become larger, and calculation of it is more complex. To solve this problem, we design a negotiation model of manufacturing resource, and propose a method based on MapReduce to calculate the offer in certain round of the negotiate which is the core of negotiation. And this method is implemented on Hadoop cluster. Experiments show that the method is stable and scalable and it can meet the needs of the manufacturing resource negotiation.
Key words: MANUFACTURING RESOURCE, MAPREDUCE, NEGOTIATION

1. Introduction

With the rapid development of information technology in the manufacture, the manufacturing is becoming more and more global. Some advanced production models have been proposed, such as agile manufacturing, lean production, mass customization and cloud manufacturing [1]. Among these models, the cloud manufacturing [2] has the most obvious advantages in solving the resource needs of enterprise. In the cloud manufacturing, the manufacturing resource of enterprise has the characters of diversity, distribution and dynamic, so the accessing and management of manufacturing resource is more complex.

In the modern manufacture, due to the continuous refinement and complex of manufacturing resource, it is possible that the manufacturing resource provided in cloud manufacturing cannot fully meet the needs of the demand user, so it is necessary to negotiate between the demand side and the potential provide side (manufacturing resource is partly qualified). In order to achieve efficient processing for massive manufacturing resource, a novel manufacturing resource negotiation method based on MapReduce is proposed in this paper.

2. Model of Manufacturing Resource Negotiation

At present, some models of negotiation has been proposed [3, 4], but these negotiation models are not designed for manufacturing resource. We design the negotiation model of manufacturing resource, which focus on the characteristics of manufacturing resource.

In the cloud manufacturing, manufacturing resource represents the software and hardware elements of production activities throughout the product life cycle, including all elements of the design, manufacture, maintenance and other related activities involved in the process.

In the negotiation model of manufacturing resource, user try to find a mutually acceptable solution with the highest total utility on the negotiation attributes of manufacturing resource in one or several rounds. The relevant definitions in the negotiation model of manufacturing resource are described as follows.

User Set: refers to the set of users in the manufacturing resource negotiation, it is denoted as $U = S \cup D$ where $S = \{s_1, s_2, \cdots, s_i\}$ denotes the set of supply users and $D = \{d_1, d_2, \cdots, d_j\}$ denotes the set of demand users.

Attribute Set: refers to the set of manufacturing resource's attributes in negotiation, is denoted as $A =< a_1, a_2, \cdots, a_n >$, where n denotes the size of attributes of the manufacturing resource in the negotiation, and $\forall a_k \in A, v^{a_k} \in \Omega_k$ where v^{a_k} denotes certain user's value of attribute a_k and Ω_k denotes the value range of the attribute a_k.

Utility Function: used to evaluate the effectiveness of each user. The utility function of user u is defined by $U_u(A) = \sum_{k=1}^{n} W_u^{a_k} \times \Phi_u^{a_k}$ and $\Phi_u^{a_k} = \frac{v^{a_k} - v^{\min}}{v^{\max} - v^{\min}}$. The term $v^{max}\left(v^{min}\right)$ denotes the maximal (minimal) value of user *u* on the attribute a_k. Weight factor $w_u^{a_k}$ denotes the particular preference of user *u* on the attribute a_k.

The goal of the negotiation among the users is to obtain the maximal social utility which is the total payoff of negotiation parties according to their utility function. If negotiation fails, utility of all parties is zero then the total utility is minimum, so the worst result of negotiation is negotiation failed. The negotiation can obtain maximize social utility only when negotiation is successes, meanwhile its result must be Pareto efficient solution.

In order to solve the negotiation, we construct a fitness function which estimates the utility of opponent by means of calculating the offer's distance between certain user and its opponent. The fitness function is defined as follows.

$$fitness(offer_u) = \alpha \times TP(t) \times \frac{U(offer_u)}{U(offer_{\max})} + (1 - \alpha \times TP(t)) \times (1 - \frac{dist(offer_u, offer_{apponent})}{dist(offer_{\max}, offer_{\min})}) \qquad (1)$$

Where the parameter $\alpha \in [0,\ 1]$ is the trade-off factor to control the relative importance of optimizing one's own payoff or reaching a deal, the term $TP(t)$ is the time factor, $dist(offer_x, offer_y)$ is the offer vectors distance between the user x and the user y.

Based on the fitness function, a variety of genetic algorithms(GA) [4, 5] have been proposed to solve the problem of negotiation. The manufacturing resource is becoming more and more complex, and more and more attributes of manufacturing resource need to be negotiated, then solution spaces of GA in negotiation become larger. In order to get the global solution, more individuals in GA are required. So there are a lot of calculations in the manufacturing resource negotiation, and it is suit to be solved by MapReduce.

3. Framework of MapReduce

Map-Reduce model [6, 7, 8] is a parallel programming framework proposed by Google, it provides a distributed file system for user, which allows users to handle large-scale data easily. And all the procedures for computing are abstracted into two basic operations of Map and Reduce, data will be decomposed into smaller split in Map stage and executed on different nodes of the cluster, and the results are integrated summary in the Reduce phase. Map-Reduce model is a simple but very effective parallel programming model.

MapReduce of Apache Hadoop [9] is an open-source MapReduce style distributed data processing framework, which is an implementation similar to the Google Map-Reduce. Apache Hadoop MapReduce uses the HDFS [10] distributed parallel file system for data storage, which stores the data across the local disks of the computing nodes while presenting a single file system view through the HDFS API. When the Map and Reduce function is executed, Hadoop can optimizes data communication by scheduling computations near the data by using the data locality information provided by the HDFS file system. Figure 1 shows the overall flow of the MapReduce operation in Hadoop's implementation.

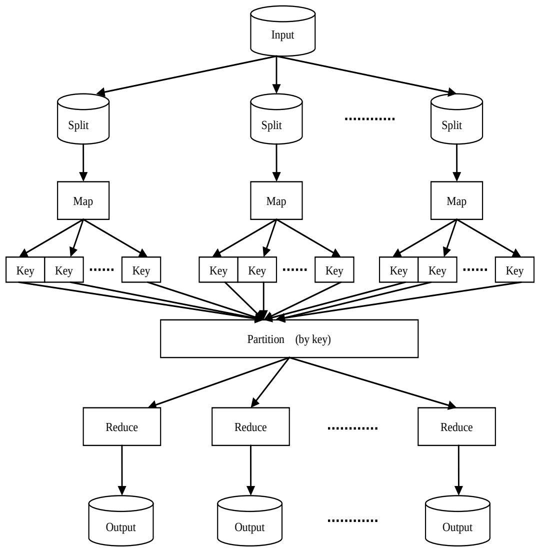

Figure 1. The overall flow of the MapReduce in Hadoop.

4. Method of Negotiation Based on MapReduce

Manufacturing resource negotiation is processed between the users (supply and demand), and after some rounds of negotiation, when the supply user's offer is meeting to the demand user's expectation, then the demand user accept the offer proposed by the supply user and the negotiation is success. If the supply user's offer can't meet the demand user's expectation, the demand user proposes its offer, and the negotiation goes into next round. The negotiation fails when time of negotiation is over.

Due to a lot of individuals need to be generated in the negotiation based on GA, and those individuals need to be crossed and mutated, so there are a large amount of calculate on the manufacturing resource negotiation. MapReduce can decompose the large scale data into a number of splits and process those splits concurrently, and then the calculation efficiency of the negotiation can be increased greatly, so we can process the manufacturing resource negotiation based on MapReduce.

It can find out that the core of negotiation is how to calculate the user's offer. There are some approaches [11, 12] try to solve it, but those approaches are not flexible and efficient. We propose a novel MapReduce method to calculate the offer. In this method, the calculation of the fitness function for each individual is performed in the Map phase (function), and the operation of crossover and mutation are processed in the Reduce phase (function). After multiple iterations, the final results of offer in negotiation can be calculated. Its overall flow is shown in figure 2.

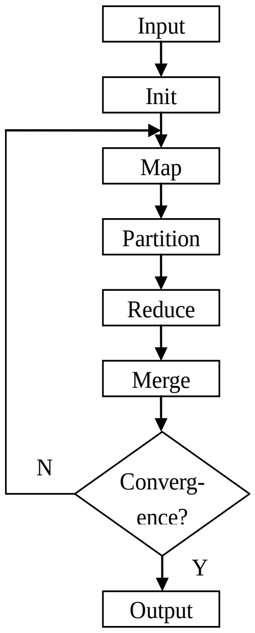

Figure 2. The overall flow of the offer calculate based on MapReduce

In this method, we process the iteration in offer’s calculation of negotiation as a separate MapReduce job. The MapReduce job contains those phases.

Input phase: input the user's offer of current round in negotiation and the others parameters of fitness function.

Init phase: present the data in offer and initialize the population of GA.

Data present: data calculated in GA is numeric, so the non-numeric data of manufacturing resource’s attributes need to be converted into the numeric data firstly, such as if the data of the attribute "class" is one of the set (excellent, good, fair, poor), then it can be mapped to (1,2,3,4), so the attribute "class" can be calculated by GA. After be calculated, the numeric data need to be converted into the non-numeric data by the data distance (Euclidean distance). For example, if the calculate result of attribute "class" is 2.9 and the data distance between 2.9 and 3 is the smallest, so the calculate result of attribute "class" in the offer is "fair".

Population initialize: the manufacturing resource’s attributes in the negotiation are denoted as $A=\{a_1,a_2,\cdots,a_n\}$, then the individual is presented by the binary string and the substring of the binary string can be mapped to an attribute of *A*. Those individuals (binary strings) are generated randomly and stored on HDFS of Hadoop as lines in HDFS file.

Map phase: evaluate the fitness of the individuals and store the individual which has the greatest fitness in current round. The function Map is shown as follows.

Algorithm 1: function Map

```
Map (key, value) {
    fitness=caculateFitness(value);
    indiVector=new IndividualVector(value, fitness);
    //store the current best individual and its fitness value in HDFS file result
    if (fitness>currentMaxFit){
        currentMaxFit=fitness;
        currentBestIndi=value; }
    result=HDFS.store(value,currentMaxfit);
    key=CreateKey(indiVector,random);
    value=indiVector; }
```

In the function Map, we create the vector IndividualVector to contain individual's value (binary string) and its fitness. The Cereatekey() creates the key of the individual, the Key impact the process of function Reduce, current generate method of key is random, and the other methods can be adopted according to the user's need.

Partition phase: determine how to distribute the <key, value> generated by Map to the Reduce nodes. Because we have assigned a random key by random function for each individual in Map phase, thus the key of the individual is uniform distribution, and the default partition method Hash() is able to balance the Reduce. So we employ the default partition method, and user can override it.

Reduce phase: generate new individuals and store those individuals in HDFS file. The function Reduce is shown as follows.

Algorithm 2: function Reduce

```
Reduce(key, values) {
    //get all the individuals on the Reduce node
    if values.hasNext() {
        individuals.add(value.getValue());
        fitness.add(value.getFitnessl()); }
    else (
        newindiividuals=SelectionAndCrossover(individuals,);
        newindiividuals =Mutation(newindividuals,finess); }}
    HDFS.strore(newindividuals); }
```

In Reduce function, each Reduce node generates the new individuals through the operation of selection, crossover and mutation and stores the new individuals (binary strings) in HDFS file separately.

Merge phase: merge the segregative HDFS files generated by Reduce nodes to a single HDFS file and this single HDFS file is the input of Map phase in the next iteration.

During the iterations, the deviation between the best individuals generated in each iteration is smaller and smaller. When it is less than the constraint, GA algorithm is convergent, and then the iteration of GA is stopped. The offer of current round in negotiation is generated.

The core problem of negotiation is the calculation of the offer, so we solve it by using the above method base on MapReduce. On this basis, the manufacturing resource negotiation can be solved easily.

5. Experiments and Results

The experiments were performed in a Hadoop cluster. This cluster contains 1 master node (intel i5 CPU, 8G RAM and 2T disk) and 10 slaver node (Intel i3 CPU, 4G RAM and 2T disk). This cluster runs on Centos 6.4 and Hadoop 1.0.4.

The negotiation is processed between the demand user and the supply user and the manufacturing resource has some attributes $a_i, v^{a_i} \in [1,5]$. The main parameters used in GA are similar to the setting in [5]. We implement the method described in figure 2 by Java.

The experiment is preformed to evaluate the method's stability with increasing the problem size. The number of the nodes in the Cluster is 10, the number of attribute is increasing, and that is 50, 100, 150, 200, 250 and 300 respectively. The experiment results are shown in figure 3.

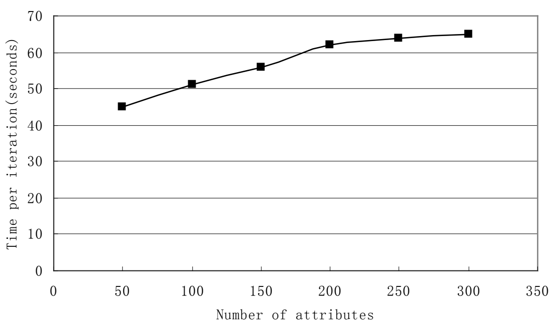

Figure 3. Stability with increasing the number of attribute

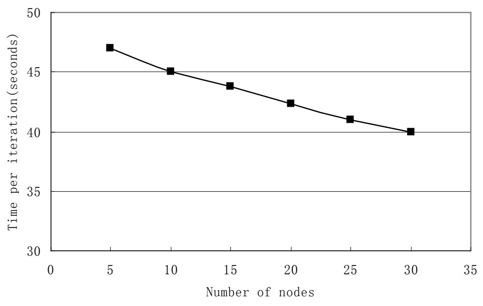

Figure 4. Scalability with increasing the node

Figure 3 shows that although the time per iteration is increasing with the increasing of the attribute's number, but the time per iteration is not linear increase with the number of the attributes. It shows that the method is stable with increasing the problem size, and means that the method has a relatively good performance in processing the manufacturing resource with a lot of attributes.

Scalability with increasing the cluster node of MapReduce is the main indicator of a method in terms of expansion. We evaluate the processing capabilities of this method with the same problem size (50 attributes) and the resource (the node in the cluster) is increasing by virtualization, the numbers of node are 5, 10, 15, 20, 25 and 30. The experiment results as shown in figure 4.

It figure 4, the time per iteration is decreasing with the increasing of the MapReduce node in the cluster, it means that the processing efficiency of this method can be significantly improved with the increasing the MapReduce node. So the method is scalable with increasing the cluster node.

From the experiments above, it can be seen that the manufacturing resource negotiation method based on MapReduce we proposed not only maintains stable performance when the problem space (number of attributes in manufacturing resource negotiation) is increasing, but also can improve the processing efficiency of the negotiation by increasing the MapReduce node in the cluster, So it is suit to process the negotiation of manufacturing resource.

6. Conclusions

In this paper, we describe a model of manufacturing resource negotiation, and propose a method based on MapReduce to calculate the offer of the negotiation. The method is flexible and efficient for processing the manufacturing resource negotiation.

How to improve the performance of this method, and achieve multiple user negotiation by MapReduce is need to be researched in the further work.

Acknowledgements

This work was supported by Jiangsu Province Academician Workstation project of China (BMZ01050551) and Jiangsu Province Graduate Science and Technology Innovation Fund projects of China (CXLX12_0627).

References

1. Li B.H., Zhang L., Ren L., Chai X.D. (2012) Typical Characteristics, Technologies and Applications of Cloud Manufacturing. *Computer Integrated Manufacturing Systems,* 18(07), p.p.1345-1356.
2. Zhang L., Luo Y., Tao F., Li B.H., Ren L. (2014) Cloud manufacturing: a new manufacturing paradigm. *Enterprise Information Systems,* 8(2), 167-187.
3. Okumura M., Fujita K., Ito T. (2013) An Implementation of Collective Collaboration Support System Based on Automated Multi-agent Negotiation. *Complex Automated Negotiations: Theories, Models, and Software Competitions*, 435(12), p.p. 125-141.
4. Lau, R. Y. (2005). Towards genetically optimised multi-agent multi-issue negotiations. The 38th *Annual Hawaii International Conference on System Sciences*, Hawaii, American, p.p. 35-45.
5. Yang Z., Geng X. (2011) Multilateral Multi-Issue Negotiation Model of Agent Based on Hybrid Genetic Algorithm. *International Conference on Management and Service Science*, Wuhan, China, p.p. 121-124.
6. Geng X., Yang Z. (2013) Data Mining in Cloud Computing. *International Conference on Information Science and Computer Applications*, Changsha, China, p.p. 223-230.
7. Dean J., Ghemawat S. (2008) MapReduce: simplified data processing on large clusters. *Communications of the ACM,* 51(1), p.p. 107-113.
8. Dean J., Ghemawat S. (2010) MapReduce: a flexible data processing tool. *Communications of the ACM,* 53(1), p.p. 72-77.
9. Holmes A. (2012) *Hadoop in practice*. Manning Publications Co. Greenwich.
10. Shvachko K., Kuang H., Radia S., Chansler R. (2010) The hadoop distributed file system. *IEEE 26th Symposium on Mass Storage Systems and Technologies*, Incline Village, American, p.p. 1-10.
11. Jin C., Vecchiola C., Buyya R. (2008) MRPGA: An extension of MapReduce for

parallelizing genetic algorithms. *IEEE Fourth International Conference on eScience,* Indianapolis, American, p.p. 214-221.

12. Di Geronimo L., Ferrucci F., Murolo A., Sarro, F. (2012) A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites. *IEEE Fifth International Conference on Software Testing, Verification and Validation*, Montreal, Canada, p.p.785-793.