www.s2is.org
International Journal on Smart Sensing and Intelligent Systems
ISSN 1178-5608

The Design and Evaluation of a Strategy of Data Placement in Cloud Computing Platform

Wei Guo[1], Kaibo Luo[2], Xinjun Wang[1*] and Lizhen Cui[1]
1: School of Computer Science and Technology,
Shandong University, China
2: Bishan Power Supply Bureau, Chongqing Electric Power Company,
Bishan, China
*: Corresponding Author, Emails: guowei@sdu.edu.cn

Submitted: Oct. 10, 2013 ***Accepted: Feb. 2, 2014*** ***Published: Mar. 1, 2014***

Abstract- Cloud computing has become a new platform for personal computing. However, while designing the strategy of data placement, there still lacks the consideration of systematic diversity of distributed transaction costs. This paper proposes the use of genetic algorithms to address the data placement problem in cloud computing. This strategy has adequately considered the correlation between data slices to minimize the total cost of distributed transactions. Compared to other methods, genetic algorithms have proven to comprehensively consider the correlation between the data slices in cloud computing, therefore greatly reducing the amount and cost of distributed transactions.

Index terms*:* Cloud computing, data placement, distributed transaction, genetic algorithm, web service.

I. INTRODUCTION

Cloud computing is a novel type of network application and is gradually changing the developmental trend of information technology (IT) industry[1]. Currently, Google, IBM, Yahoo and other well-known enterprises have put forward their own cloud computing solutions. Commercial cloud services distribute whole computing tasks into a large number of resource pools composed of various computers, servers and data storage systems, thus allowing users to obtain computing capacity, storage space and information services to meet their demands[2,3,4].

Business collaborations between enterprises have become increasingly frequent. These businesses generally can't be completed within one department, but need more departments to do in collaboration. These businesses are often inter-departmental, even cross-regional[5,6].

Cloud computing service providers are tasked with building an infrastructure that can support the many applications, users and workloads, and with an ever increase usage from corporations and private consumers comes the need for platforms to efficiently distribute data[7,8]. Because a single data node cannot accommodate entire information, it is vital to partition data and place slices on appropriate data nodes. In the real cloud computing environment, we usually deploy and implement various transaction-intensive applications and these transactions often require access to a variety of data. If data is stored in different data nodes, it will inevitably bring some distributed transactions. However, the cost of distributed transaction is very high[9,10].

In the environment of multiple data nodes, the significance of choosing an appropriate storage position for large data are as follows: (1) in a cloud computing environment, a single data node cannot accommodate all data and hence it must be dispersed into multiple data nodes; (2) if the data placement strategy is unreasonable, the distributed transaction cost will definitely increase during access of the data nodes, thereby sharply reducing the computing capacity of the cloud platform; and (3) some data that can only be stored in specified data nodes[11,12].

Placing data on nodes of a cloud is a NP problem. In light of the above problems, we designed a cloud data placement strategy to minimize the distributed transaction cost of data access across data nodes. In consideration of the relationship between the data and distributed transactions costs, the main contribution of this paper is to propose a cloud data placement strategy supporting the minimization of the distributed transaction cost. We use the genetic algorithm to realize this data placement strategy by analysis and verification with experiments.

The rest of the paper is organized into the following sections: in section 2 related works on cloud data placement strategy are introduced, and differences between those works and the significance of this research are discussed; section 3 describes the related conceptions of cloud data placement problems and models and the cloud data placement strategy algorithm; we optimize the data placement strategy in section 4; the experimental analysis and evaluation is placed in section 5; and then the final section of this paper summarizes the paper and forecasts our further work.

II. RELATED WORKS

Mohamed y. Eltabakh et al. [13] studied the placement strategy that correlates the data node with the corresponding computing node, and designed the system architecture--CoHadoop. CoHadoop is a slight extension of Hadoop, allowing the application to control where data is stored. It improved related data placement strategy based on Hadoop's nodes. Compared with the plain Hadoop, the improvement of CoHadoop mainly brought further benefits for the following operations, indexing, grouping, aggregation, columnar storage, join, and sessionization. That paper didn't consider the problem of our research, and was based on the Hadoop. So it can only be used for processing file systems of large data blocks.

To address the data placement management issue of process-oriented data intensive applications in cloud computing environment, [14] proposed a data placement strategy based on cluster matrix. First, it constructed a dependency matrix for all of the application data, and this matrix represented all data sets' dependency relationships (including the data placed in specific data node). Then it used the BEA algorithm[15] to generate a clustering matrix and divide the matrix elements to obtain higher mutual dependency between the data sets in each partition. For each new data set, the algorithm calculated the dependency between this data set and all of the others, and moved it to the data node that obtains the highest dependency. The experimental results showed that the clustering data placement strategy, compared with random data placement strategy, could effectively reduce the data transmission across the data nodes in the process of application execution. This work mainly faced to the scientific workflow with large data sets and mainly considered the relationship of data sets in the process. However, it did not consider the total cost of distributed transactions between the data slices.

Carlo Curino designed the Schism[16] to drive the data partitioning through transactions. In this strategy, all of the data records were mapped into vertices, and accesses to transactions were mapped into edges. A graph model was constructed to classify the data through the graph partition technology.Then the established data copies through the transactions to correlate with data to avoid the distribution transactions.

Xin et al. proposed an intelligent data placement strategy to improve workflow performance while minimizing data transmission between the data centers simultaneously[17]. At the start-up stage, the entire data set was divided into a smaller data item, and then distributed into many data centers. At the placement stage, it considers over the data centers' computing capabilities, storage budget, correlations of data items and so on. At the run-time stage, the intermediate data were placed in the appropriate data center. It uses linear discrimination analysis, and considers the amount in start-up stage and past behavior of data center (i.e. the reliability in respect of task retardation). This strategy effectively improved the overall progress in calculation and minimizing the communication expenditures caused by data movement.

While the works discussed thus far addressed cloud data placement for related research, there still lack studies that attempt to resolve the cloud data placement problem among transaction intensive applications. Furthermore, none have yet considered the distributional transaction costs caused by data correlation.

III. DATA PLACEMENT STRATEGY

a. Formalization Definition

This section describes the conceptions about data placement problem under cloud computing environment, and constructs its formal model. Cloud computing environment is composed of multiple distributed data nodes. Since our research is mainly focused on data placement problems under cloud computing environment, we will only discuss data node capacity, the relationship among data slices, and data collaboration cost among data slices. We will not consider the amount of copies.

We can formally represent a cloud computing environment as a set that is composed of multiple distributed data nodes, $\mathrm{DN} = \{dn_i \mid i = 1,2,...,n\}$. dn_i represents the data node whose serial number is *i*, and *n* is the amount of data nodes. We also define dns_i as the data storage capacity of data node dn_i. Each data node storage capacity, dns_i , is determined by the user.

Since the storage capacity of data node is limited, due to the correlation of data in cloud and multi-tenancy features, we need to slice the data. The data slice is a set that is composed of a series of closely related data. We formally define the set of data slices, DS $= \{ds_i \mid i = 1,2,...,m\}$. ds_i represents the data slice which is numbered *i* while *m* is the number of data slices. We also define dss_i as the size of the data slice ds_i, and dsp_i as the fixed storage position of the data slice whose serial number is *i*.

If two different data slices are not stored in the same data node, then a collaboration cost is needed when the two different data slices are accessed in one transaction. This cost is called collaboration cost of the distributional transaction. It describes the correlation between two data slices. This collaboration cost apparently tends to be extremely high[9]. We define the collaboration cost matrix between the data slices as:

$$\mathrm{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mm} \end{bmatrix}$$

For arbitrary *i*, *j* = 1,2,...,*m* and $i \neq j$, the element c_{ij} in matrix C represents the collaboration cost between the data slice ds_i and the data slice ds_j. We assume that the collaboration costs between the various data slices are knowable so that it can be obtained through data log files and applications.

In the cloud computing environment, a data placement plan means a mapping from the data slices set DS to data nodes set DN and for arbitrary ds_i, there is an unique corresponding $dn_j \in$ DN. Then we define a data placement plan as DPP=$\{ds_i \rightarrow dn_j \mid i = 1,2,...,n\}$. $ds_i \rightarrow dn_j$ indicates that the data slice ds_i is placed in data node dn_j.

The ideal cloud data placement strategy we seek is to place data slices in appropriate data nodes so to make the data nodes not exceed capacity limits, as well as minimize the total collaboration costs between the data slices. Our goal is to design a strategy to quickly and accurately find the optimal solution for data placement problem.

Genetic algorithm is an effective searching method based on the principle of natural selection and genetics. It starts from a population, and then using selection, crossover and mutation operators to evolve the Chromosomes in the population to eventually reach a whole, optimal solution[18]. We use the fast random overall search ability of the genetic algorithm as the foundation for our data placement strategy to seek the lowest cost data placement plan.

b. The Encoding And Decoding Of Chromosome

The evolutionary process of genetic algorithm is based on the coding mechanism. Encoding can greatly affect performances such as the algorithm's search ability and population diversity. We encode the chromosome based on the binary coding method, and the coding character set applied here is composed of 0 and 1 binary symbols. Each chromosome is a binary coding string representing a data placement plan. For each data slice $ds_i \in$ DS in DS, we use a data node encoding whose length is $\lceil \log_2 n \rceil$ to denote its storage location. For each data placement plan, we use the $m \times \lceil \log_2 n \rceil$ bit binary number to denote it. For example, when *m* = *n* = 12, the data slice ds_i is stored in the data node dn_i and the corresponding code is: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011. The length of the chromosome is 48.

When decoding the chromosome, we can decode every $\lceil \log_2 n \rceil$ binary string according to the value of *m* and *n*, and work out the number that represents the location of the data node where the data slice lies.

c. The Validity Judgments Of The Chromosome

Not all chromosomes are valid data placement plans; we will later explain the various reasons and provide justification for validating a chromosome.

If there exists at least one gene fragment that cannot represent data nodes in DN in the chromosome of DPP, then the solution DPP is invalid. The formal expression is:

$$\exists \forall ds_i \in DS, ds_i \rightarrow dn_j \text{ and } j \geq n \text{, then DPP is invalid} \qquad (1)$$

In order to ensure normal operation, data nodes generally set the storage capacity as dns_i. If one data placement plan makes any data slices' size in a same data node exceed its storage capacity dns_i, then we can conclude that the data node is overloaded. The formal expression is:

$$\exists\forall dn_i \in DN, \sum_{ds_j \to dn_i} dss_j > dns_i \text{, then DPP is invalid} \qquad (2)$$

If there exists at least one location-fixed data slice whose storage location is different from its assigned location in the data placement plan DPP, then the solution DPP is invalid. The formal expression is:

$$\exists\forall ds_i \in DS, ds_i \to dn_j \textit{ and } dsp_{\mathrm{i}} \neq \mathrm{j}\text{, then DPP is invalid} \qquad (3)$$

During the generation phase of the initial population in the genetic algorithm, as well as the process of genetic selection, crossover and mutation stage, we need to judge the validity of the chromosome. If any of (1), (2) and (3) is met, we shall determine the data placement plan invalid and abandon this plan in the genetic process.

d. The Initial Population Of The Solution

Due to genetic algorithm operations requirements, the initial population is prepared to be composed of a number of initial solutions for the operations. Each chromosome in the initial population is generated randomly. The size of the population directly impacts on the convergence rate and the generation of the optimal solution. If the size of the population is too large, then the rate of convergence will be reduced. Contrarily, if the size of the population is too small, it is difficult to obtain the optimal solution. The generation of the initial population is random. The cloud computing environment has numerous data nodes and data slices, and if we produce the initial population using the completely random method, the population needs to be large enough to reflect the randomness. Otherwise, we cannot reach our goal. This paper selects the individuals that meet the three conditions in section above as the initial population. In this way we could generate the optimal solution and speed up the convergence rate.

e. The Evaluation On Collaboration Cost In DPP

The fundamental objective of the data placement plan we propose is to minimize the collaboration cost caused by the operations between different data nodes in the cloud computing environment. Thereby, we adopt the collaboration cost of the data slices for a special data placement plan as its evaluation function. For the given set of data nodes DN, data slices set DS, the collaboration cost matrix among the data slices C, and data placement plan DPP, we define the evaluation function of the data placement plan as *g*(DPP) = CollaborationCost (DN, DS, C, DPP), and obtain DPP's total collaboration cost through the following algorithm 1.

Algorithm 1. The algorithm of DPP's collaboration cost

```
input：DN,DS,C,DPP
output：CollaborationCost (DN,DS,C, DPP)
the main steps：
    initialization：set CollaborationCost = 0
    For $i$=1 to $m$ - 1   //$m$ is the number of data slices
        For $j$=$i$+1 to $m$
            If (the number of data nodes corresponding to $i$ and $j$ is not the same) then
                CollaborationCost= CollaborationCost+ $c_{ij}$
            End if
        End for
    End for
    Return (CollaborationCost)
```

f. The Generation Strategy Of The Next Population

After the initial population is generated, our strategy will generate the next population by using selection, crossover and mutation operations on the anterior population. The selection operation will choose individuals with the highest fitness from the initial population. These individuals will most likely have advantageous traits that will provide a competitive edge, procure more resources for survival, and, thus, pass those traits to their offspring in the next generation. The crossover operation randomly pairs individuals in the initial population and to randomly choose the crossover point to exchange information between each pair. The mutation operation randomly mutates some bit of the individual. When performing the crossover and mutation operations, we must consider the probability of crossover and mutation. Therefore, our strategy will set certain crossover and mutation rates.

g. The Algorithm To Seek The Data Placement Plan

The result returned by algorithm 2, to seek the data placement plan, is a set of data placement plans with low cost. Plans in this set are sorted according to the corresponding collaboration cost in ascending order.

Algorithm 2. The data placement plan algorithm using genetic algorithm

```
input：DN,DS,C,DPP
output：the set of data placement plan with lower collaboration cost
the main steps：
  initialization：define the variables NP,NG  //NP: the size of the population, NG: the
                                                  number of evolutional generations
  Set  i = 1
  While (i <= PopulationSize)
    Generate a random data placement plan dpp
    If (dpp is valid and not in SP) then   //judge using the formula (1)(2)(3)
        Join the dpp to SP
        Calculate the collaboration cost of this data placement plan
        i ++
    End if
  End while  //by now, the initial population is created
  Set curGen = 1
  While (curGen <= maxGen)
    Choose the Rg proportion of data placement plans into the next generation of SP',
          according to the collaboration cost of data placement plans in ascend
    While (sizeof(SP')< NP)
          If (random(0,1) < Pm)  then  //Pm is the rate of mutation operation
              Choose randomly the position to mutate at the gene in the dpp, and mark the
                 result as dpp'
            If (dpp' is valid and not in SP') then
               Join the dpp' to SP'
               Calculate the cooperation cost of this data placement plan
               continue
            End if
```

```
      End if
      If (random(0,1) < Pc)  then   //Pc is the rate of crossover operation
        Choose randomly two different solutions dpp1 and dpp2 in the SP
        Choose randomly the position to crossover at the gene in the dpp, and mark
          the result as dpp1' and dpp2'
        If (dpp1' is valid and not in SP') then
          Join the dpp1' to SP'
          Calculate the collaboration cost of this data placement plan
          continue
        End if
        If (dpp2' is valid and not in SP') then
          Join the dpp2' to SP'
          Calculate the collaboration cost of this data placement plan
          continue
        End if
    End if
  End while
End while
Return (SP')
```

In the process of generating the next generation and selecting the data placement plan to reserve, our algorithm choice is based on the collaboration cost through which we can greatly improve the quality of the genetic populations, and quickly converge at the optimal solution.

h. The Web Service Specification

Through the above algorithms, we can use the web services to solve the problems of the data placement. The web service in the cloud computing platform is represented by the web service definition language (WSDL)[19]. WSDL is an XML-based language created for describing web services and their operations. It regulates the location of the services and the operations or methods provided by the services. WSDL also describes the protocols and message formats when the web services interacts to bind, generally adopting the abstract language to describe the operation and information supported by the services. Only by the use of them, the actual network protocols and message formats are bound to the services.

An example of service represented by WSDL in cloud computing platform is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
 <definitions name="PersonInsuranceInfo"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
   <schema targetNamespace=" http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      <element name="PersonInsuranceRequest">
      <complexType>
        <all>
          <element name="PersonInsuranceID" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="PersonInsuranceResult">
      <complexType>
        <all>
          <element name=" PersonInsuranceSet " type="string"/>
        </all>
      </complexType>
    </element>
   </schema>
 </types>
</definitions>
```

IV. STRATEGY OPTIMIZATION AND IMPROVEMENT

As described in this paper, our data placement strategy is based on the genetic algorithm, a commonly used method to resolve combinatorial optimization by means of simulating genetic biology. In the genetic algorithm the initial population is encoded, and the algorithm is tasked to exert operations on individuals after assessing their fitness to the environment, so by way of an evolutionary process. Because we are restricted by system computational capability, population size and number of iterations are limited. Therefore, the selection of initial population becomes a vital factor, which may result in either success or failure of the algorithm and can directly affect the optimization efficiency and effectiveness of the algorithm. In order to obtain an initial population with higher quality when generating an initial population, we can evaluate each individual and obtain a fitness value by calculating the fitness function. Individuals below a specific threshold value will not be retained so as to converge onto the same result. When the genetic algorithm is adopting an effective initialization method, the required number of iterations is much less than that through random or uniform kinds of methods, and thus the search speed is improved[20, 21].

The population iteration of the genetic algorithm is achieved through selection, crossover and mutation. This can result in many deficiencies like computational complexity, unstable controllability, and wide range of the search area. There often appear such phenomena as difficult in shared searching radius, poor convergence prone and precocity. Thus, it leads to serious partial accumulation of evolutionary individuals. We can apply elite reserved strategy and shared function niche genetic algorithm to solve these problems because the elitist strategy ensures the convergence of the algorithm and the sharing function niche guarantees the results' diversity[22].

V. EXPERIMENTAL EVALUATION

The experimental environment is Intel (R) Core (TM) 2 Duo 2 x 2.93 GHz, 8 GB of memory, 500 GB of hard disk and 1 GB of network bandwidth. The experiment compares the data placement strategy described in this paper with the random placement strategy, the greedy strategy.

The random placement strategy: The data slices are placed in data nodes randomly, only meeting the requirements of data node's data storage capability. For data slices requiring fixed position, we place them on fixed data node.

The greedy strategy: This strategy is to firstly arrange the data slices in descending order according to the collaboration costs among them, and then to place the data slices with the higher collaboration cost in a same data node as much as possible. When data slices exceed the storage capacity of the data node, we place the data slice on a new data node.

For the same number of data slices, we randomly construct different collaboration cost matrices and calculate the total collaboration cost of the selected plan for each strategy above. Each data slice runs 100 times and we adopt the average from the final results. Collaboration costs in Figure 1 and Figure 2 are all simulated costs. The strategy represented in this paper is briefly named GA strategy.

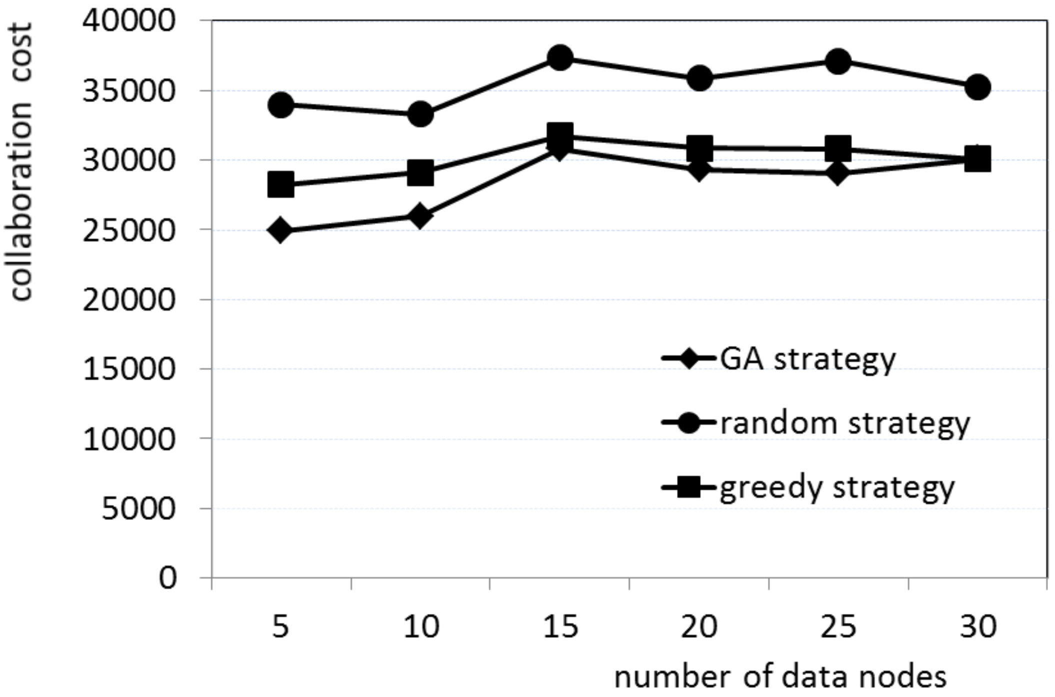

Figure 1. The collaboration cost of 50 data slices with varying number of data nodes

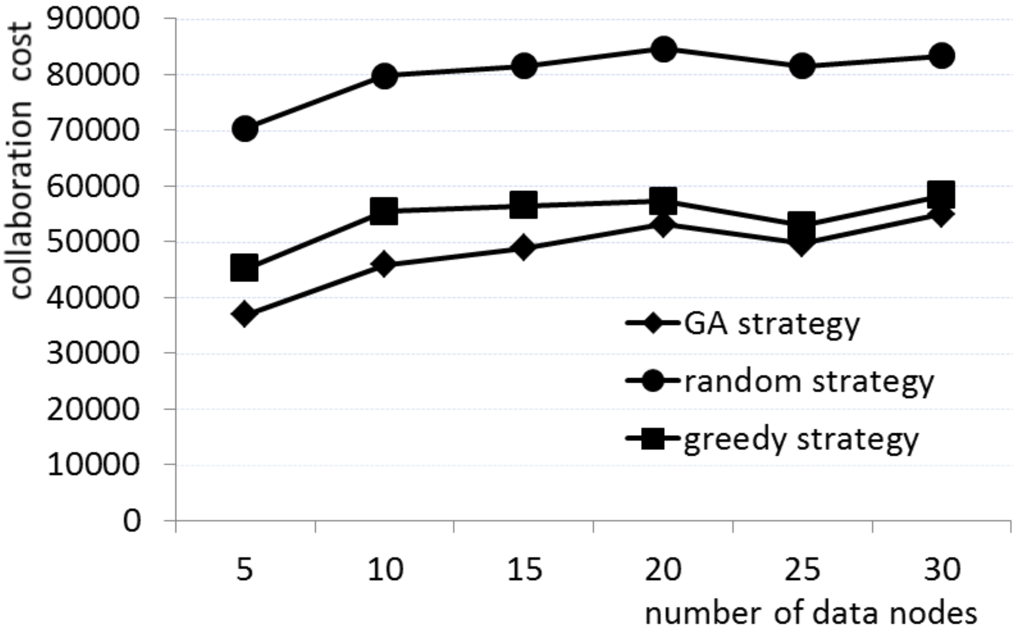

Figure 2. The collaboration cost of 100 data slices with varying number of data nodes

Figure 1 is for 50 data slices, and the number of data nodes is 5, 10, 15, 20, 25, and 30 respectively. By applying the above three strategies to evaluate collaboration cost, we finally take the average value after running them 100 times. Figure 2 is a calculated coordination cost for 100 data slices while other environmental factors are similar to the above one. It can be obviously seen from the Figure 1 and Figure 2 that coordination cost of the final chosen data placement plan by applying genetic algorithm is lower than those by the other two strategies The gap is tremendous when compared with the random placement strategy.

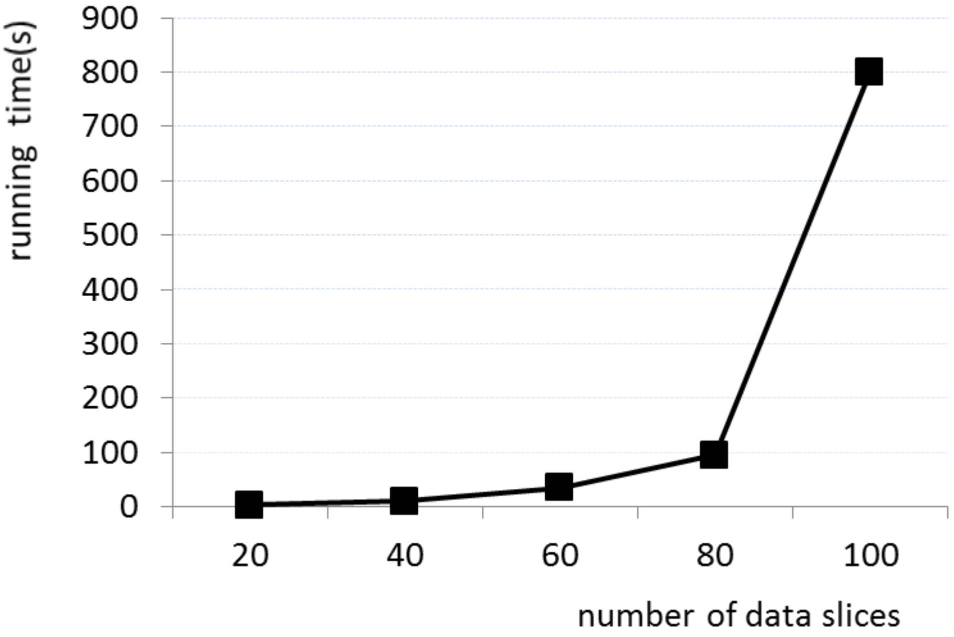

Figure 3. The running time of 10 data nodes with varying number of data slices

As is shown in Figure 3 for different numbers of data slices, the strategy described in this paper has a different running time. Increasing the number of data slices dramatically raises the strategy operation time. For example, as the number of data slices increased from 80 to 100, the running time increased eight-fold. Therefore, our proposed strategy cannot be applied to large numbers of the data slices and data nodes.

In summary, the strategy to place the data slices in data nodes discussed in this paper is optimal when the number of the data slices and data nodes is not too high. Compared to the random placement and the greedy strategies, our strategy provides an efficient data placement plan with relatively lower collaboration cost.

VI. CONCLUSIONS

Cloud computing platform has become a relatively novel platform for enterprise and personal computing, and is convenient for business collaboration across data nodes. In regards to the data placement strategy in cloud computing platform, this paper proposes a brand new method that adequately considers the dependencies between the data slices and minimizes the total cost of distributional transactions. This paper provided a detailed performance assessment of our strategy and compares it to those of others. We demonstrate that our strategy considers the relationship among data in cloud computing to greatly reduce collaboration cost. With the development of the business in the cloud computing, the placement of data in data nodes changes dynamically. How to adapt to this change more quickly and effectively and improve the capacity of cloud computing platform will be our future research directions.

ACKNOWLEDGEMENT

This work has been supported by:(1) the National Key Technologies R&D Program No.2012BAH54F04; (2) the National Natural Science Foundation of China under Grant, No.61003253; (3) the Natural Science Foundation of Shandong Province of China under Grant No. ZR2010FM033, No. ZR2010FM031, No. ZR2010FQ010; (4) Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 200804221031.

REFERENCES

[1]Mell P, Grance T. The NIST definition of cloud computing, Version 15, Gaithersburg: National Institute of Standards and Technology, Information Technology Laboratory, Technical Report, SP-800-145, 2009.

[2]Michael Armbrust,Armando Fox,and Rean Griffith,et al. Above the Clouds:A Berkeley View of Cloud Computing, mimeo, UC Berkeley, RAD Laboratory, 2009.

[3]Ian Foster, Carl Kesselman, and Steve Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15(3), 2001.

[4]Rui J. Advanced secure user authentication framework for cloud computing. In International Journal on Smart Sensing and Intelligent Systems, v6, n4, September 2013: 1700-1724.

[5]Christian Pichler, Manuel Wimmer, Konrad Wieland,Marco Zapletal, and Robert Engel. Towards Collaborative Cross-Organizational Modeling. BPM 2011 Workshops, Part I, LNBIP 99, pp. 280–292, 2012.

[6]Michele Amoretti, Alberto Lluch Lafuente, Stefano Sebastio. A Cooperative Approach for Distributed Task Execution in Autonomic Clouds. 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing,274-281.

[7]Fazio M, Paone M, Puliafito A, Villari M. HSCLOUD: Cloud architecture for supporting Homeland Security. In International Journal on Smart Sensing and Intelligent Systems, v5, n1, p 246-276, March 2012.

[8]Xue S, Xu X, Wang D, Zhang J, Ji F. METECLOUD: A Private Cloud Platform for Meteorological Data Storage Using Hadoop. In International Journal on Smart Sensing and Intelligent Systems, v6, n2, April 2013: 648-663.

[9]E. P. Jones, D. J. Abadi, and S. Madden. Low overhead concurrency control for partitioned main memory databases. In SIGMOD, pages 603–614, 2010.

[10]A. Pavlo, E. P. Jones, and S. Zdonik. On predictive modeling for optimizing transaction execution in parallel OLTP systems. VLDB, 5:85–96, October 2011.

[11]Hu J, Deng J, Wu J. A Green Private Cloud Architecture with global collaboration Telecommun System (2013) 52:1269–1279.

[12]Cao J, Wu Z, Zhuang Y, Mao B, Yu Z. A Novel Collaborative Filtering Using Kernel Methods for Recommender Systems. Chinese Journal of Electronics Vol.21, No.4, Oct. 2012:609-614.

[13]Mohamed Y. Eltabakh, Yuanyuan Tian, Fatma Ozcan, Rainer Gemulla, Aljoscha Krettek, John McPherson. CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop, Proceedings of the VLDB Endowment, Vol. 4, No. 9,2011.

[14]Yuan D, Yang Y, Liu X, Chen J J. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, Volume 26, Issue 8, 1200-1214, 2010.

[15]McCormick W T, Sehweitzer P J, White T W. Problem decomposition and data reorganization by a clustering technique. Operations Research, 1972, 20: 993-1009.

[16]CURINO C,JONES E,ZHANG Y, et al. Schism: a Workload-Driven Approach to Database Replication and Partitioning. The VLDB Endowment,2010,3(1),48-57.

[17]Xin L, Anwitaman Datta. Towards intelligent data placement for scientific workflows in collaborative cloud environment. 2011 IEEE International Parallel&Distributed Processing Symposium, 1052-1061.

[18]Srinivas M,Patnaik L M. Genetic algorithm: a survey. IEEE Computer,1994,27(6):17-26.

[19] http://www.w3.org/TR/2007/REC-sawsdl-20070828/.

[20]HYUN C J，KIM Y K． A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines computers［J］． Operations Research，1998,25(7 /8) : 675-690.

[21]P. Ponterosso, D. S. J. Fox. Heuristically Seeded Genetic Algorithms Applied to Truss Optimisation[J].Engineering with Computers 1999.4(15): 345-355.

[22]Zhou X, Yang X, Xiang C,Xie C. Semantics Web service component evaluation model based on ontology. Computer Integrated Manufacturing Systems, 2008,14(12) : 2346-2353.