MapReduce-based Similarity Measurement for Business Processes

Juntao Gao, Xueshan Wang, Yongan Wang
School of Computer and Information Technology
Northeast Petroleum University
Daqing, China
Email:gjt [AT] nepu.edu.cn

Abstract—**Similarity measurement of business processes is a basic operator which is useful in several scenarios, such as business process management, business process searching, business process re-engineering and so on. Firstly, the framework of the methodology is proposed; secondly, two kinds of trace generation method is discussed ; thirdly, the measurement of trace similarity is studied from information theory perspective; fourthly, the parallel algorithm to compute similarity of business processes is described based on MapReduce; at last, the experiments is illustrated and the conclusion is drawn.**

Keywords:business processes, similarity measurement, MapReduce

I. INTRODUCTION

Nowadays, Business process management has been widely adopted to improve the efficiency, reduce the cost and raise the quality. Along with time, plenty of business process models have been accumulated and become important intellectual assets which represent the business handling procedures of the organizations [1~3]. It is important to make a deep insight into these business processes and their mutual relationship. Similarity measurement is a basic operation in many applications in business process management, such as, process mining [4], process retrieval and process integration.

Researches on process similarity measurement have been conducted from different perspectives. In graph theory, graph isomorphism is often used to measure the similarity between two graphs [5]. However, the method usually only examine edges and nodes without catching the syntactical issues of business processes. The delta-algorithm is proposed to measure the difference between two models in database field [6]. Unfortunately, it still doesn't resolve the issues of the method mentioned above. In process algebra theory, Trace equivalence and bisimulation equivalence are usually employed to compare two process models [7-9]. A method based on trace equivalence is also proposed to assign weights to each trace based on execution logs which reflect the importance of a certain trace [10]. These methods determine whether two process models are identical or not, but they do not tell how much they differ.The common activity names are used to compute the similarity score[11].While this method is very simple and fast to implement, its major shortage is that the structure of processes does not be taken into considered.Furthermore, its application is limited to a domain with controlled vocabulary.Label matching similarity is proposed in [12], in which no information about the order of nodes has been taken into account. Ehrig[13] measure the similarity of process models based on so-called semantic business process models. The measure is not symmetric which is one of the important properties of similarity.Graph edit distance is employed to capture structural similarity[12,14~16]. However, the different graphs dose not represent different business processes.

So far, traces are one of the most widely acknowledged representation of the behavior of business processes. This paper studies the semantic similarity between business processes based on traces. Next section proposes the framework of the methodology which involves three stages; the third section analyzes the way to capture trace set from a business process and studies the algorithm to implement the trace generator; the forth section studies the method to comparing traces from different models that is the basic blocks in this methodology; the fifth section give the algorithm to compute the business process similarity based on MapReduce.

II. THE FRAMEWORK OF METHODOLOGY

As illustrated in Fig. 1, The methodology proposed in this paper employ trace set to specify the behavior of business processes. It involves three steps: Trace Generation, Trace Comparison and Model Similarity Computing. Firstly, generating traces from predefined business process models. The execution of business process results in a trace. The trace faithfully records the process of businesses. In some cases, traces can be directly retrieved from the log of PAIS(Process-Aware Information Systems), such as ERP(Enterprise Resource Planning), SCM(Supply Chain Management), PDM(Product Data Management) and WFM(Workflow Management system). In some other cases, traces can not be directly retrieved but be generated by simulation. The technology to achieve the traces from business processes is detailed in section 3.

Secondly, comparing the traces from different models. As we known, a trace is a sequence of actions(events) occurred in the system. The lengths of traces compared may be different and the position of actions have different influence on the semantic of the trace. The metric for traces is studied in the section 4.

Thirdly, The similarities of traces are merged to compute the similarity between business process models. The count of the traces in the computing is bigger, the result of similarity is more accurate. On the other hand, time complexity of the computing is higher. In order to resolve the problem of the high complexity, MapReduce is employed to implement the algorithm in distributed architecture.

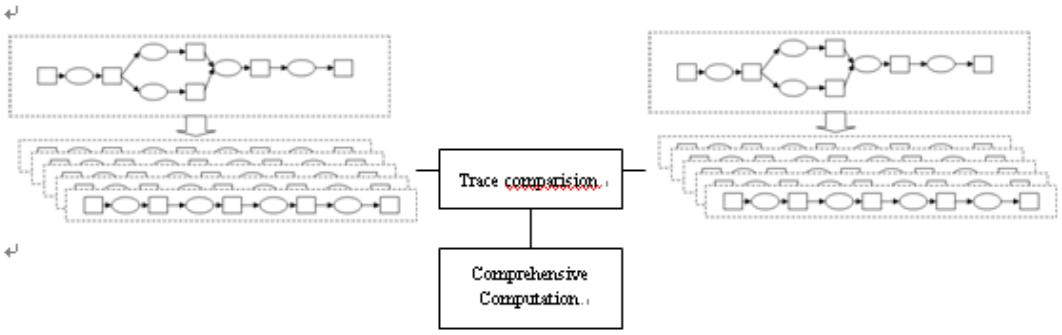

Figure 1. The framework of the methodology

III. TRACE GENERATION

Trace generation is the first step in this methodology. There two essential questions in this step: (1) how to select typical traces from the real process executions in the case of the PAIS already enacts; (2) how to simulate traces from business process models.

A. *Trace selector*

The simplest selecting algorithm is to select the universal set of available traces. However, the count of universal set is usually too large to handle efficiently. Therefore, it is important to find the least subset which has the approximate effect to the universal set.

Intuition 1:

The same traces imply the same semantic information of business process models.

Intuition 2:

Different traces may have different effect on the business process models. According to the above two intuitions, the algorithm is described as below pseudo-codes.

```
void simplify(array us, array ss, array count)
  {
    int count=0;
    for(int i=0;i<us.size();i++)
    {
      int j;
      for(j=0;j<count;j++ )
      {
        if (us[i]==ss[j])
        {
          count[j]++;
          break;
        }
      }
      if(j==count) {
        ss[count]=us[i];
        count++;
      }
    }
  }
```

The algorithm iterates the universal set of traces *us*, if the trace occurs in the array of ss, then increase the corresponding value of count, else append it on the tail of ss. Finally, the elements of ss are selected as candidates to compare the similarity of business processes.

B. *Trace simulator*

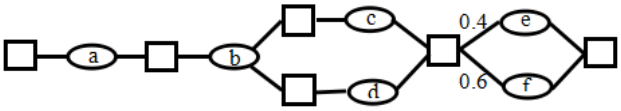

Figure 2. The example of business models

Trace simulator is designed to generate traces from some business process by simulation. During each simulation one trace is generated. There are two issues from loop structure need to address. If the business process model includes loops, the sets of traces may become infinite and the length of the trace may also be infinite. In order to address the two issues, the weight of traces is introduces into trace simulator. The basic logic is that the weight of traces depends on the its occurrence times. The algorithm is described as follows.

(1) trace-arrival. The case arrival function can be seen as defining the actual start of a process and how often this process is triggered within a certain time.

(2) decision making. Decision making function is employed to make the decision which route to take at the fork point. The rules of decision setup based on the possibility or random selection.

(3) weight determining. If the trace is generated, the weight is determined using this formula

$\omega = \prod_i Possibility_i$ i represents the i[th] decision making at the fork point.

TABLE I. THE RESULT OF TRACE BY SIMULATION

Number	sequence	weight
1	abcde	0.2
2	abcdf	0.3
3	abdce	0.2
4	abdcf	0.3

Consider the process shown in Fig. 2. There are six actions involved in the process. The probability to execute action e is 0.4 and the probability to action f is 0.6. The result of trace simulation is described in Tab. I. The weight is determined by the probability of each trace.

IV. TRACE COMPARISON

Little research has been conducted on comparison between traces [17][18][19]. These papers focus on the action traces in which each action is a symbol or numeric value. However, the real action traces are often described by a short text. The relationship between these actions is often not identical or completely different. Further more, the same action traces may be represented in different ways in different organizations. This is because that different organization using different terminologies and the problem of semantic heterogeneity makes it a tedious job to compare textual action traces. Next, a definition of action trace similarity is constructed according to information theory. The idea of similarity propagation is introduced to pick out a mapping between corresponding activities and data, and Hungarian algorithm is expanded to reduce its time complexity.

Suppose $\sum$ denotes the universe of actions, there exist two traces α and β,

$\alpha =< x_1, x_2, ..x_m >$ and $x_i \in \sum, i \in [1, m]$

$\beta =< y_1, y_2, ..., y_n >$ and $y_j \in \sum, j \in [1, n]$

x_i denotes the i-th action in trace α, i=1,2,...,n. $|\alpha|$ denotes the length of trace α, which is the number of actions in α. y_j denotes the j-th action in trace β, j=1,2,...,n. $|\beta|$ denotes the length of trace β, which is the number of actions in β.

The commonality of α and β is depicted by $common(\alpha, \beta)$, $common(\alpha, \beta) =< X \bigcap Y, lct >$, X is the action set of α, Y is the action set of β. Because the action may occur more than one time, X and Y are both multisets. The commonality of α and β includes two parts: one is the common action set $X \bigcap Y$, the other is the longest common subtrace, lcs for short, from the common action set.

The combination of race α and β is depicted by $description(\alpha, \beta)$, $description(\alpha, \beta) =< X \bigcup Y, \{\alpha, \beta\} >$, the combination of α and β also includes two parts, one is the union of action set $X \bigcup Y$, the other is two alternative action sequences $\{\alpha, \beta\}$.

According to information theory[5], the reference similarity of action traces is:

$$sim(\alpha, \beta) = \frac{\log P(common(\alpha, \beta))}{\log P(descritipn(\alpha, \beta))} \quad (1)$$

If the probability of trace is known, the above formula can be computed using the following formula.

$$sim(\alpha, \beta) = \sqrt{\varepsilon \times \frac{|X \bigcap Y|^2}{|X \bigcup Y|} + \phi \times \frac{|lct|}{|X \bigcap Y|}^2}$$, where $\varepsilon + \phi = 1$ (2)

The value of ε and ϕ is determined by the amount of information contained in the action sets and their orders. Generally, the cardinality of universal action set is very large, the probability of common actions occur is very little and so the amount of information contained in action sets is very large. While given the common action set, the probability of same order occur is relatively big and so the amount of information contained in it is relatively less.Therefore, ε is bigger than ϕ.

V. MODEL SIMILARITY COMPUTING

This section presents some preliminary information about MapReduce, describes in detail the algorithm to solve the Similarity computing problem in a parallel manner and presents the modifications to improve the algorithm's performance

The MapReduce framework was introduced in [20]. The Map-Reduce-Merge variant [21] extends the MapReduce framework with a merge phase after the reduce stage to facilitate the implementation of operations like join. Map-Join-Reduce [22] is another MapReduce variant that adds a join stage before the reduce stage. In this approach, mappers read from input relations, the output of mappers is distributed to joiners where the actual join task takes place, and the output of joiners is processed by the reducers.

MapReduce based Algorithm

Input: trace set M, N

Output: similarity collection.

Steps:

1.m mappers read the input trace slices. Compute the similarity of each trace pair $p \in M \times N$.

1.1. Constructing the Similarity Matrix

1.2. Picking up the best Matching
1.3. Computing the Commonality
1.4. Compute the similarity between traces
2.To output the similarity of the trace set.
2.1. Computing the sum of the similarity of trace pairs
2.2. Computing the average value of similarity
3.R reducers receive the results of trace slices
3.1. Computing the sum of the similarity of trace slices
3.2. Output the average value of similarity

Although there is no standard way to evaluate computational measures of model similarity, one reasonable way to judge can be agreement with human similarity ratings.

In the experiments, twenty subjects was chosen and given 20 model pairs. The subjects were all experienced clerks in an airplane manufacturing enterprise. The models were sent to the subjects in different order by email. According to the judgments, the subjects choose one of results: identical (1), very similar (0.8), similar (0.6), different (0.4), very different (0.2), and absolutely different (0).

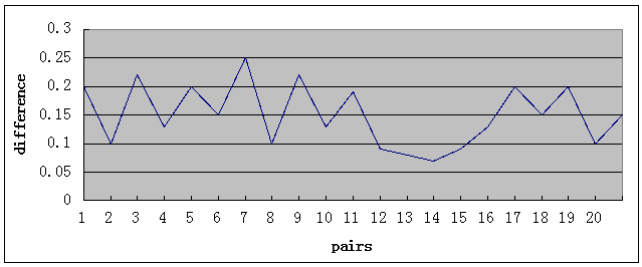

Figure 3. The result of experiment

Fig.3 shows the result of the experiment. The average similarity over the twenty subjects was compared with the computational similarity measurement. The average difference is 0.15, and the biggest difference is 0.25.

VI. CONCLUSION

In this paper a new approach to measure similarity between business process models on distributed architecture is proposed. In order to address the issues that the complexity of computation is too high, MapReduce is employed to implement the parallel algorithm. The work in this paper has been applies into a project of cross organization ERP implementation. In the future, the method still needs more projects to verify.

ACKNOWLEDGMENTS

The research is supported by the Education Department of Heilongjiang province science and technology research projects (No. 12541094).

REFERENCE

[1] M. Dumas, W.M.P. Van der Aalst, A.H.M. Ter Hofstede, Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley & Sons, 2005

[2] M. Rosemann. Potential pitfalls of process modeling:part b. Business Process Management Journal,12(3):377-384, 2006

[3] M. Dumas, L. Garcia-Banuelos, R. Dijkman:Similarity search of business process models. IEEE Data Engineering Bulletin 32 (2009) 23-28

[4] Rakesh Agrawal, Dimitrios Gunopulos, Frank Leymann. Mining Process Models from Workflow Logs. Lecture Notes in Computer Science Volume 1377, 1998, pp 467-483

[5] E. B. Krissinel and K. Henrick, "Common subgraph isomorphism detection by backtracking search," Software: Practice and Experience, vol. 34, pp. 591-607, 2004.

[6] W. M. P. van der Aalst, "Business Alignment: Using Process Mining as a Tool for Delta Analysis," Requirements Engineering, vol. 10, pp. 198-211, 2005.

[7] W. Yongxiang, W. Jinzhao, and J. Jianmin, Process algebra ─ symmetry and action decomposition: Science Press, 2007.

[8] J. Hidders, M. Dumas, W. M. P. van der Aalst, A. H. M. T. Hofstede, and J. Verelst, "When Are Two Workflows the Same?," in Australaian Symposium on theory of Computing, vol. 41: ACM, 2005, pp. 3-11.

[9] W. M. P. van der Aalst and T. Basten, "Inheritance of Workflows: An approach to tackling problems related to change," Theoretical Computer Science, vol. 270, pp. 125-203, 2002.

[10] W. M. P. van der Aalst, A. K. A. de Medeiros, and A. J. M. M. Weijters, "Process Equivalence: Comparing Two Process Models Based on Observed Behavior," presented at BPM'06, Vienna, Austria, 2006.

[11] R. AKKiraju, A. Ivan, Discovering business process similarities: An empirical study with SAP best practice business processes. In: Service-Oriented Computing- 8th International Conference. Volume 6470 of LNCS. (2010) 515-526

[12] R. Dijkman, M. Dumas,B. Van Dongen, R. Kaarik, J. Mendling, Similarity of business process models: Metrics and evaluation. Inf. Syst. 36(2011) 498-516

[13] M. Ehrig, A. Koschmider, A. Oberweis, Measuring similarity between semantic business process models. In: Fourth Asia-Pacific conference on Conceptual modelling - Vol. 67 (2007) 71-80

[14] R. Dijkman, M. Dumas, L. Garcia-Banuelos, Graph matching algorithm for business process model similarity search. In: Business Process Management. Volume 5701 of LNCS. Springer(2009) 48-63

[15] C. Li, M. Reichert, A. Wombacher, On measuring process model similarity based on high-level change operations. In:27th International Conference on Conceptual Modeling, Springer(2008) 248-264

[16] J. Bae, J. Caverlee, L. Liu, H. Yan, Process mining by measuring process block similarity. In: Business Process Management Workshops 2006. Volume 4103 of LNCS., Springer(2006) 141-152

[17] LI Yan, FENG Yu-qiang. A Quantitative Analysis Method of Business Process based on Sequence Alignment. System Engineering Theory and Practice,2007, 27(4):54-61

[18] Gerke, K., Cardoso, J.,Claus, A.: Measuring the compliance of processes with reference models. In: On the Move to Meaningful Internet Systems – Confederated International Conferences 2009, Proceedings, Part I, Springer(2009) :76-93

[19] David Maier (1978). "The Complexity of Some Problems on Subsequences and Supersequences". J. ACM (ACM Press) 25 (2): 322–336

[20] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In OSDI, 2004.

[21] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-reduce-merge: simplified relational data processing on large clusters. In ACM SIGMOD, 2007.

[22] D. Jiang, A. K. H. Tung, and G. Chen. Map-join-reduce: Toward scalable and efficient data analysis on large clusters. IEEE Trans. on Knowl. And Data Eng., 23:1299 -1311, September 2011.