A Novel Parallel Scheme for Fast Similarity Search in Large Time Series

YIN Hong[1,3], YANG Shuqiang[1], MA Shaodong[2], LIU Fei[1], CHEN Zhikun[1]

[1] College of Computer, National University of Defense Technology, Changsha 410073, China
[2] School of Engineering, The University of Hull, CottinghamRoad,Hull, United Kingdom, HU6 7RX
[3] Xiangyang School for NCOs, Xiangyang 441118, China

Abstract: The similarity search is one of the fundamental components in time series data mining, e.g. clustering, classification, association rules mining. Many methods have been proposed to measure the similarity between time series, including Euclidean distance, Manhattan distance, and dynamic time warping (DTW). In contrast, DTW has been suggested to allow more robust similarity measure and be able to find the optimal alignment in time series. However, due to its quadratic time and space complexity, DTW is not suitable for large time series datasets. Many improving algorithms have been proposed for DTW search in large databases, such as approximate search or exact indexed search. Unlike the previous modified algorithm, this paper presents a novel parallel scheme for fast similarity search based on DTW, which is called MRDTW (MapRedcue-based DTW). The experimental results show that our approach not only retained the original accuracy as DTW, but also greatly improved the efficiency of similarity measure in large time series.

Keywords: similarity; DTW; warping path; time series; MapReduce; parallelization; cluster

I. INTRODUCTION

Time series is a common data type that has been widely used in a variety of domains, such as scientific research and economic analysis. Many methods for mining time series databases to get useful information in an efficient manner have been proposed. Das et al. presented that how association rules can be learned from time series [1]. Catalano et al described an algorithm for discovering patterns in time series [2]. Keogh and Pazzani considered the problem for scaling up time series classification algorithm [3]. Debregeas and Hebrail attempted to introduce a new technique for improving time series clustering algorithm to deal with massive datasets [4]. The similarity search, as a core subroutine, has been applied in most time series data mining algorithm. For example the clustering algorithm needs to measure the similarity in time series to ensure whether they are the same class. The mining algorithm of association rules also need to translate the sub-sequences of time series into items according to the similarity. Therefore the computational expense of similarity search will become the bottleneck of time series mining algorithms.

Most of similarity search methods are based on Euclidean Distance but it may be an ex-

tremely brittle distance measure. Euclidean Distance is very fragile to small temporal distortions and, in some particular application, notch distance measure is required. This means that no adjustment is made for difference in scale. Dynamic Time Warping (DTW) improves those weaknesses by reaching optimal alignment between multiple time series, which are warped in a nonlinear fashion to match with each other. Therefore DTW can be used to measure the similarity in time series with patterns shifted in time or distorted in sharp.

Unfortunately, due to the quadratic $O(N^2)$ time and space complexity, it is hard for DTW to measure similarity in large time series with more than thousands of data points. On the other hand, the growing volume of datasets make the implementation of mining algorithm complicated in modern medical and industrial database. For example, many research hospitals have EEG data with trillions of data points. The telemetry data of domestic flights collected by NASA Ames has tens of trillions of data points, and some power companies will record a trillion data points in every four months. An improved algorithm of the DTW is desirable to reduce both temporal and spatial complexities and it is likely to produce the optimal warp path between large time series.

In this paper the MapReduce[5]-based dynamic time warping (MRDTW) parallel algorithm is developed, which is able to search through an more accurate warp path between two large time series. The working principle of MRDTW algorithm is achieved by breaking the larger series into sub-sequence and comparing their similarities in the segmented time series. The comparison of sub-sequence is independent, which allows the MRDTW to use parallel scheme like MapReduce. Unlike the previous methods modified the DTW algorithm, the main contribution of this paper is try to improve the performance of DTW using the parallel scheme when dealing with large time series. Experimental evaluation results illustrate our approach retains the searching accuracy as the DTW, and the efficiency of similarity measure has been greatly improved in large time series.

The rest of this paper is organized as follows. Section II outlines the related work about the modifications of DTW. Section III contains the measure of similarity. Section IV introduces the optimization of DTW. Section V is the algorithm implementation of MRDTW. The experimental results are discussed in Section VI. We conclude the paper in Section VII.

This paper presents a novel parallel scheme for fast similarity search based on Dynamic Time Warping(DTW), which is called MapReduce-based DTW.

II. Related Work

Dynamic time warping (DTW) has been successful in many areas. Typical applications include speech processing [6], gesture recognition [7], astronomy [8], mining of historical manuscripts [9], and electronic patient records [10], etc.

Conventional DTW is considered incompetent and slow in time for large databases mining. The critical constraint is the computational burden that has been widely recognized by other researchers. For example, Chadwick et al. found that due to time constraints, DTW did not perform well for large time series [10]. Likewise, Bemdt and Clifford realized that the performance of DTW on very large databases may be a limitation [11], and [12] noted that it is not be sufficient for scaling DTW methods to truly massive databases.

However, huge potential for DTW has also been seen if some of the technical issues could be addressed. To reduce the computational complexity, many improvement algorithms have been proposed for DTW search in large databases. Existing methods can be categorized into two types: *approximate search* and *exact indexed search*. Keogh et al. proposed a Piecewise Dynamic Time Warping (PDTW) which took advantage of the fact that most time series can be efficiently approximate by the Piecewise Aggregate Approximation (PAA) [13]. Salvador et al. proposed an approximation of DTW (FastDTW) [14] which uses a multilevel approach that recursively projects a solution from a coarse resolution. The proposed approach can also refines the

projected solution to approximate linear time and space relationship. The above methods, including Iterative Deepening Dynamic Time Warping (IDDTW) [15], Segmented Dynamic Time Warping (SDTW) [16] belong to the first type. The main limitation of the approximate search is that it is not guaranteed to find the optimal solution [14]. Apart from approximate time series, guiding the search to access only parts of the database is another candidate and various index structures for DTW distance have been studied. Keogh et al. proposed the LB_PAA which shows that PAA can be adapted to allow indexing under DTW [17]. FTW (Fast search method for Dynamic Time Warping) indexing which is introduced by Sakurai et al. can efficiently filter a significant number of search candidates, and leads to a direct reduction in the search cost [18]. A more recent approach, *i*SAX (indexable Symbolic Aggregate Approximation) gives a discrete representation to time series and changes time series into a sequence of symbols which can be indexed with a tree topology for simplicity [19]. However, one drawback of index-based techniques is that random access to the database will cause huge amount of I/O cost. It means that indexing will be burdened with so large I/O overheads that it is not suitable for massive databases.

MapReduce, which was proposed by Google, is a parallel programming model for processing massive data sets with a distributed algorithm in a cluster [5]. More detail about MapReduce will be given in section V. Many applications benefits from MapReduce, such as Hadoop [20] which is a distributed system infrastructure, and MongoDB [21] which is a database based on the distributed file storage. Accordingly, incorporating MapReduce with DTW will present a novel way to improve the mining efficiency.

III. The Measure of Time Series Similarity

3.1 The similarity of time series

Time series similarity comparison is highly demanded in data-based research and application. For example it is very important for aerospace improving fault detection and prediction though data mining. A large time series can be decomposed into thousands of data points, and each of which represents a value at a certain time.

Definition 1. *Time Series*: A time series $T=t_1,t_2...t_m$ is a sequence of numbers m collected at regular intervals over a period of time, and m called the length of T.

Definition 2. *Sub-sequence*: Given a time series T of length m, a sub-sequence C of T is a sampling of length n ($n \leq m$) of contiguous positions from T, that is, $C=t_p....t_{p+n-1}$, $1 \leq p \leq m-n+1$.

Definition 3. *Similarity of Time Series*: Consider two time series S_1 and S_2, their similarity function is $Sim(S_1,S_2)$, given a threshold ε, if:

$$Sim(S_1, S2) \leq \varepsilon \quad (1)$$

is established, it is say that S_1 and S_2 are similar to the case of the ε boundary.

A time series can be converted into a discrete representation by using the appropriate segmentation (e.g. a sliding window) and then clustering these sub-sequences using similarity measure. Euclidean Distance, Pattern Distance, Slope Distance and Dynamic Time Warping (DTW) are classical methods for time series similarity measure. This paper will pay attention to DTW's optimization in parallelization.

3.2 Review of dynamic time warping

The concept of DTW was first introduced by Bemdt and Clifford for analyzing the small-scale time series. In their works, DTW was examined from theoretical viewpoint and its time complexity was thought as a limitation. In the subsequent algorithm's introduction, a review about the classical DTW algorithm will be presented.

Suppose that there are two time series Q and C, with the length of number n and m respectively:

$$Q=q_1,q_2,...,q_i,...,q_n \quad (2)$$

$$C=c_1,c_2,...,c_j,...,c_m \quad (3)$$

Firstly, an n-by-m distance similarity matrix is constructed, where the (i, j) element of the matrix contains the distance $d(q_i,c_j)$ between the two points q_i and c_j(Here we use the Manhattan Distance, so $d(q_i,c_j)=|q_i-c_j|$). The matrix can be used to align the two sequences and each matrix element (i, j) corresponds to the alignment between the points q_i and c_j. The illustration is shown in Fig.1.

Definition 4. *Warping path*: A warping path, W, is a continuous sequence of distance similarity matrix elements that defines an alignment between Q and C. The k_{th} element of W is defined as $w_k=(i,j)_k$, so: $W=w_1,w_2,...,w_k,...w_K$, and the K satisfied: $\max(m,n)\leq K\leq m+n-1$.

A warping path must be met with several constrains: *boundary conditions*, *continuity*, *and monotonicity*. Obviously, there are a great number of potential warping paths that could meet the above conditions, however, the path which have the minimum warping cost is measured as:

$$DTW(Q,C)=\min\{(\sqrt{\sum_{k=1}^{K} w_k})/K\} \quad (4)$$

K in (4) is used to reduce the effect which is produced by warping paths with the different lengths. Before searching the warping path, the cumulative distance $\gamma(i,j)$ must be defined. $\gamma(i,j)$ is defined using the distance $d(i,j)$ which can be found in the current cell and the minimum of the cumulative distances of the adjacent elements. Using dynamic programming to compare the $\gamma(i,j)$ recursively, and the warping path can be found more efficiently.

$$\gamma(i,j)=d(q_i,c_j)+\min\{\gamma(i-1,j-1),\gamma(i-1,j),\gamma(i,j-1)\} \quad (5)$$

IV. The Optimization of DTW in Parallelization

4.1 Segmentation of time series

Theoretically, the brute-force search method can be adopted to find the paths which meet the conditions, but complete exhaustion is often impractical in large time series data. For aligning two time series with the length of M and N respectively, DTW usually takes $O(MN)$ time to compute a similarity. If the M and N are very large, (e.g., M, $N>10^7$) there will be considerable amount of computations. One solution is to split the large series into smaller units, which will reduce the time complexity to $O(\frac{M}{m}\frac{N}{n})$. The m and n can achieve more than one thousand in a huge, but distribute environment according to the number of nodes.

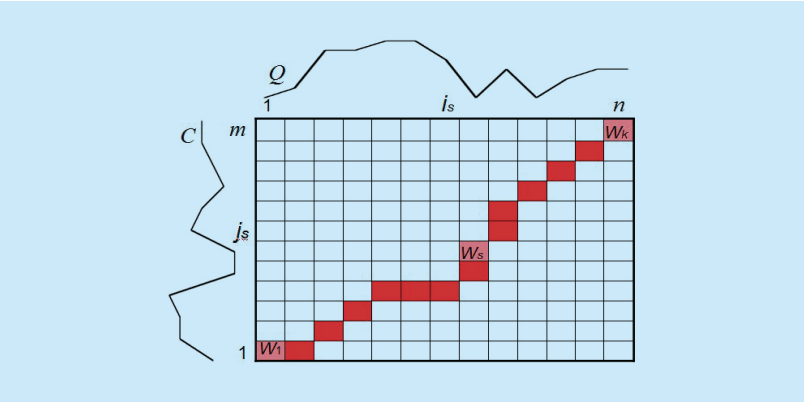

Fig.1 *The example of warping path (C is a time series with length of m, and j_s is one element of C; Q is a time series with of n, and i_s is one element of Q.)*

So the first step is to segment one of the time series C, and it is expected that the segmentation will result in sub-sequences with the form of $S_1,S_2,...,S_l$. Each piece of sequences is located on different nodes in a distributed cluster. Time series Q is not segmented in this paper, and is distributed on every node.

The Important Point Segmentation (IP) [22] method is considered here to split the time series, so the sub-sequences can remain most of the original time series modes, and each sub-sequence is relatively independent. Comparing the similarity between sub-sequences S_i and Q with DTW method respectively, the final results will be integrated to reflect the similarity between C and Q.

4.2 Parallelization of DTW

The following is an example to illustrate this process of parallel comparison. Supposing there are two time series:

$C=\{71,73,75,80,80,80,78,76,75\}$

C / Q	71	73	75	80	80	80	78	76	75
69	2	4	6	11	11	11	9	7	6
69	2	4	6	11	11	11	9	7	6
73	2	0	2	7	7	7	5	3	2
75	4	2	0	5	5	5	3	1	0
79	8	6	4	1	1	1	3	4	6
80	9	7	5	0	0	0	2	4	5
79	8	6	4	1	1	1	1	3	4
78	7	5	3	2	2	2	0	2	3
76	5	3	1	4	4	4	2	0	1

Fig.2 *The distance matrix of C and Q (The numbers in the blue region indicate the Manhattan Distance between c_i and q_i.)*

C / Q	71	73	75	80	80	80	78	76	75
69	2	6	12	33	44	55	64	71	77
69	4	6	12	33	44	55	64	71	77
73	6	4	6	13	20	27	32	35	37
75	10	6	4	9	14	19	22	23	23
79	18	12	8	5	6	7	8	11	15
80	27	19	13	5	5	5	7	11	16
79	35	25	17	6	6	6	6	9	13
78	42	30	20	8	8	8	6	8	11
76	47	33	21	12	12	12	8	6	7

Fig.3 *The warping path of C and Q (The numbers in the blue region indicate the cumulative distance, and the numbers with red mark represent the optimal path.)*

S_1 / Q	71	73	75	S_2 / Q	80	80	80	S_3 / Q	78	76	75
69	2	6	12	69	11	22	33	69	9	16	22
69	4	6	12	69	22	22	33	69	18	16	22
73	6	4	6	73	29	29	29	73	23	19	18
75	10	6	4	75	34	34	34	75	26	24	18
79	18	12	8	79	35	35	35	79	27	27	22
80	27	19	13	80	35	35	35	80	29	31	27
79	35	25	17	79	36	36	36	79	30	32	31
78	42	30	20	78	38	38	38	78	30	32	34
76	47	33	21	76	42	42	42	76	32	30	31
Node1				Node2				Node3			

Fig.4 *The result of parallel DTW (The paths with black arrow indicate all the possible paths of sub-sequence, and the paths with red mark are the optimal path of sub-sequence)*

Q={69,69,73,75,79,80,79,78,76}

According to the traditional DTW, the distance matrix is shown in Fig.2 (using Manhattan Distance).

Using the formula (5) to compute the cumulative distance $\gamma(i,j)$ of matrix D, and the obtained warping path W is shown in Fig.3.

That is W={(1,1), (1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,9)}.

Before the parallelization of DTW, the time series C is sorted into three sub-sequences, S_1 ={71,73,75}, S_2 ={80,80,80}, S_3 ={78,76,75}. The S_1 represents the mode of rising, S_2 is invariant and S_3 is the mode of falling, that is, each sub-sequence is part of its original sequence. There are three nodes in a cluster, Node1, Node2, Node3, and S_1, S_2, S_3 are located on the three nodes respectively, and Q is located on every node. The process of parallelization is to compare the similarity between S_1 and Q, S_2 and Q, S_3 and Q concurrently. The results of parallel DTW are shown in Fig.4.

On Node1, because the first point of warping path is fixed according to the *boundary conditions*, that is (1,1), obviously the path marked the red numbers is the shortest, that is W_1={(1,1),(1,2),(2,3),(3,4)}. On Node3, S_3 is the last sub-sequence, according to the definition of DTW, the last point of warping path is (n, m), so the path is also easy to be confirmed. In this case, the W_3={(7,8),(8,9),(9,9)}. On Node2, the choice of the path is relatively complex, because the first and the last point are all uncertain. We have listed all possible paths of Node2 as in Fig.4, that is: {(4,7),(5,8),(6,9)}, {(4,6),(5,7),(6,8)}, {(4,5),(5,6), (6,7)}, {(4,4),(5,5),(6,6)}, {(4,3),(5,4),(6,5)}, {(4,2), (5,3),(6,4)}, {(4,1),(5,2),(6,3)}, {(4,1),(5,1),(6,2)}, {(4,1),(5,1),(6,1)}.

4.3 Pruning of warping paths

As illustrated in the above section, there are several possible paths in the middle Node, and the selection of optimal path will be generated through pruning the possible warping paths.

Definition 5. *Flexibility*: We define the flexibility F of warping path like this:

$$F = (\sum_{k=1}^{K} |i_k - j_k|)/K \quad (6)$$

In the process of searching warping paths, a threshold ε needs to be determined. The condition $F \geq \varepsilon$, suggests that the model of two time series is so different that it can't be dynamically warping matched. In this paper, we set the $\varepsilon=K$.

Definition 6. *Continuity*: The elements in warping path are continuous, and the alignment path does not jump in the time index, satisfying: $i_s - i_{s-1} \leq 1$ and $j_s - j_{s-1} \leq 1$.

According to definition 5, the paths {(4,7), (5,8), (6,9)}, {(4,1), (5,2), (6,3)}, {(4,1), (5,1), (6,2)}, {(4,1), (5,1), (6,1)} are considered too much different, and therefore they are discarded. For the remaining five paths, according to the descending of their distance value the paths are listed as: {(4,6), (5,7), (6,8)}, {(4,5), (5,6), (6,7)}, {(4,4), (5,5),(6,6)}, {(4,3), (5,4), (6,5)}, {(4,2), (5,3), (6,4)}. There are small differences in the distance of the above five path, but considering the *continuity*, the first point of path following W_1 would be (4,4) or (4,5), and the last point of path previous W_3 would be (6,7) or (6,8). From the remaining five paths, it can be seen that only the path {(4,5), (5,6), (6,7)} satisfies the *continuity*, so W_2={(4,5), (5,6), (6,7)} is the optimal path. Combining the paths W_1, W_2, W_3, gives the final warping path W={(1,1), (1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,9)}. It is apparent that the result is the same with the traditional DTW method.

4.4 Speeding up DTW in Cluster

Although using the *Flexibility* and *Continuity*, we can prune many possible warping paths to reduce the calculations. Most of the time, it is impractical to use the *Continuity* when a sub-sequence is also the middle sub-sequence and a large number of intermediate results is unavoidable. Not only will calculation time increase greatly, but also the intermediate results will be transmitted in the cluster and the efficiency will be reduced. The feature of the cluster causes the need to accelerate the DTW, and the methods may be fall into three categories [18]:

1) *Constrains* – Limiting the number of cells that are evaluated in the cost matrix.
2) *Data Abstraction* – Performing DTW on a reduced representation of the data.
3) *Indexing* – Use lower bounding functions to reduce the running time of DTW during time series classification or clustering.

This paper focuses on the first categories, and uses the method called *Slope Constraint* to restrict the number of warping paths.

Definition 7. *Slope Constraint*: $(j_{s_b} - j_{s_a})/(i_{s_b} - i_{s_a}) \leq p$ and $(i_{s_b} - i_{s_a})/(j_{s_b} - j_{s_a}) \leq q$, where $q \geq 0$ is the number of steps in the x-direction and $p \geq 0$ is the number of steps in the y-direction. After q steps in $\boldsymbol{x}$ one must step in $\boldsymbol{y}$ and vice versa. Because of the different values of p and q, the calculation formula of $\gamma(i, j)$ also needs to be changed.

This constraint can be used to reduce the warping paths that are nearly horizontal or vertical. The illustration is shown in Fig.5.

This constraint also prevents very short parts of the sequences from matching with very long ones, which is undesirable and needs to be discarded. As shown in Fig.6, the point p refers to the points s_1, s_2, s_3, s_4, s_5, s_6 of another sequences.

There are some other constraint methods to boost searching speed of DTW in cluster, e.g., *Global Constraints*, *Local Constraints*, *Step Size Condition* are not considered in this paper but will be analyzed for optimizing our algorithm in further work.

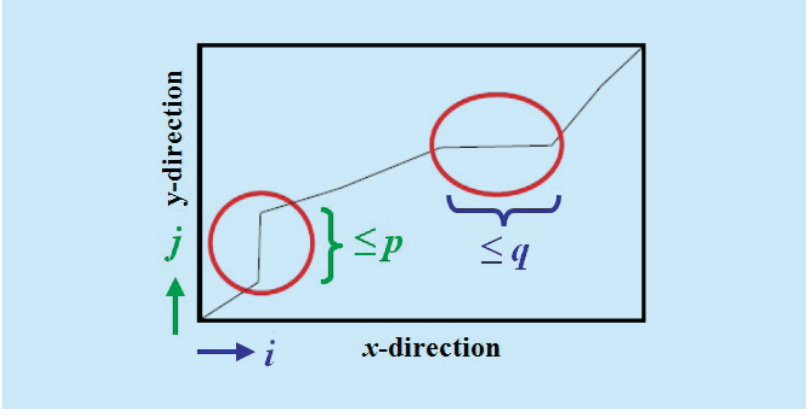

Fig.5 *The example of slope constraint (The time series do not appear in the figure.)*

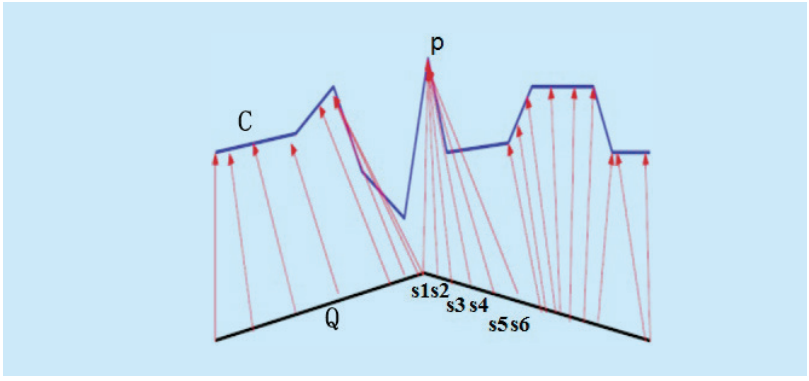

Fig.6 *Short parts match to long*

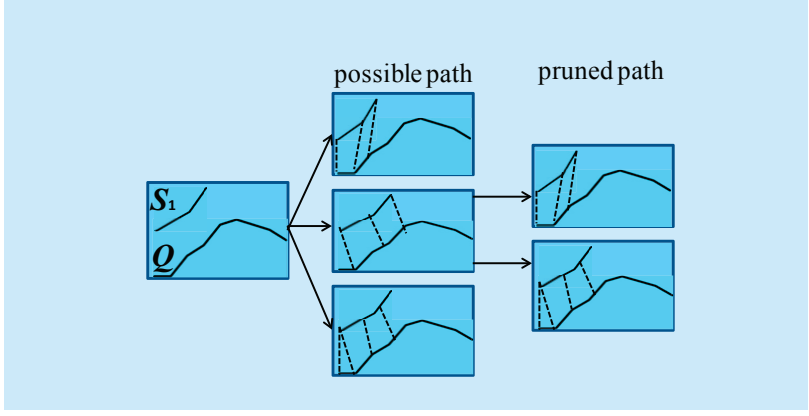

Fig.7 *The process of Map for DTW (Before Map, S is split into three subsequences, and in the figure, only the Map of s_1 is illustrated.)*

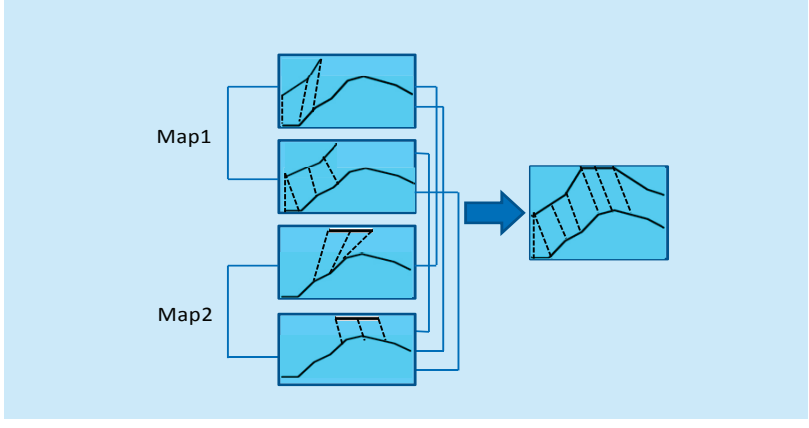

Fig.8 *The process of Reduce for DTW (Only the Reduce between s_1 and s_2 is illustrated in the example.)*

V. Algorithm Implementation of MRDTW

The algorithm implementation of parallel DTW is based on Map-Reduce which is an effective and useful framework for processing massive data. MapReduce is a parallel programming framework that allows developers to write programs and process massive data. The framework is divided into two parts:

- Map, a function that transforms the input data into a set of key/value pairs and performs filtering and sorting. All the operations will be executed individually in different nodes. For DTW algorithm, the map function is designed to find the possible warping paths between sub-sequence and another time series, and then prune these paths according to some limits.
- Reduce, a function that merges together the results from the Map and integrates the results into a single value. For DTW algorithm, the reduce function is to merge the possible warping paths into an optimal path, and the process is shown in Fig.8.

The MapReduce framework can coordinate and manager the distributed nodes efficiently, running the subtasks in parallel and managing all communication data transferred in the cluster. More detailed introductions about Map-Reduce are explained in [5]. To initialize MRDTW, one of the time series has to be split into multiple sub-sequences which will be compared with another time series in parallel. It is better to segment longer time series into pieces. The whole process of implementation is shown in Fig.9.

In terms of implementing of the proposed parallel DTW, the Map function is expounded as follows: Comparing the similarity of the sub-sequence and the sequence in parallel on each distributed node, and obtaining the possible warping paths respectively. The paths of the first and the last sub-sequence can be determined immediately, the others can be pruned by the setting of *flexibility* and the *Slope Constraint*. The specific algorithm is as follows:

MRDTW Map()

```
Input: C_i, Q (i is the number of segment)
Output: W_i[r] (r is the number of possible paths)
BEGIN
    r=1; (initialize the number of paths)
    if (i==1)
        W_i[r] = getTheFirstW(); return;
    if (i==k)
        W_i[r] = getTheLastW(); return;
    for j=1 to m do { (m is the length of Q)
        W_i[r] = getTheMidW();
        if (countFlexibility(W_i[r])<ε))
            { r++; }
    }
END
```

The implementation of Reduce function: According to the principle of *continuity*, merging the end-to-end warping paths, which satisfied $i_s - i_{s-1} \leq 1$ and $j_s - j_{s-1} \leq 1$. Calculating the total length of warping path, the path with the minimum length is the optimal solution. The specific algorithm is as follows:

MRDTW Reduce()

```
Input: W_i[r] (i is the number of segment)
Output: W
BEGIN
  for i=1 to k do { (k is the total number of segment)
    if (i==1) {
      (m_i,n_i)=getLastCoordinateOfW(W_i[1]);
      for l=1 to length(W_2[r]) do {
      (m_{i+1}−m_{i+1})=getFirstCoordinateOfW(W_2[l]);
      if ((m_{i+1}−m_i) ≤ 1 and (n_{i+1}−n_i) ≤ 1)
        W=mergeW(W_1[1], W_2[l]);
      }
    }
    for s=1 to length(W_i[r]) do {
      (m_i,n_i)=getLastCoordinateOfW(W_i[s]);
      for l=1 to length(W_{i+1}[r]) do {
      (m_{i+1}−m_{i+1})=getFirstCoordinateOfW(W_{i+1}[l]);
      if ((m_{i+1}−m_i) ≤ 1 and (n_{i+1}−n_i) ≤ 1)
        W=mergeW(W_i[s], W_{i+1}[l]);
      }
    }
  }
END
```

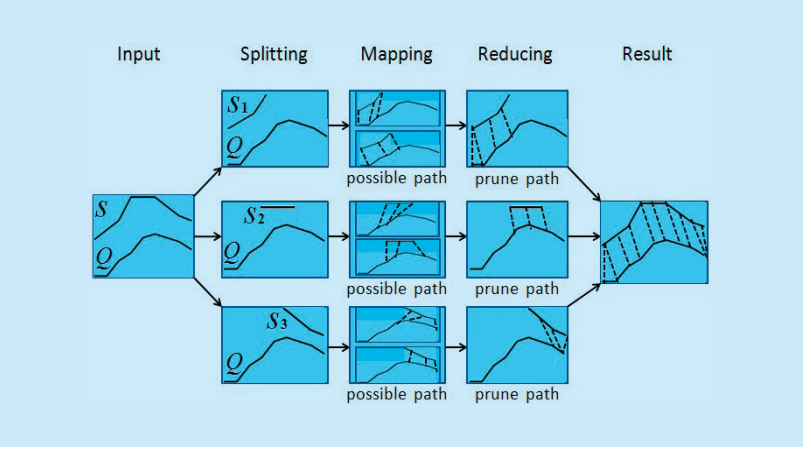

Fig.9 *The process of MapReduce for DTW (Obviously, there are more possible paths in the process, and for conveniently description, we only show two possible paths in every sub-sequence.)*

VI. Experimental Results

The goal of this experiment is to demonstrate the efficiency and accuracy of the MRDTW algorithm on very large time series data sets compared with other algorithms.

6.1 Efficiency of MRDTW

The data that is chosen to measure the efficiency of MRDTW is the parametric data from the satellite collected at a rate of 2Hz within one day. The characteristics of the data are:

1) *Large* – The size of the original data is more than 4G for one day.
2) *Multi-parameter* – The number of parameters is more than two thousand.

With respect to the length of the input time series, the efficiency of the MRDTW algorithm will be measured using time unit (second). The compared methods are the standard DTW algorithm and Euclidean distance. The MRDTW algorithm will be tested with the *node* parameter which set as 5 and 10. The lengths of the time series varied from 100 to 168578.

In this experiment the MRDTW algorithm was drastically faster than the standard DTW and Euclidean when the time series become larger than 10000. MRDTW is 270 or 430 times faster than standard DTW (using *node*

Table I *A comparison of different methods in efficiency*

Algorithm	Length of Time Series			
	100	1,000	10,000	100,000
DTW	0.02s	1.12s	50.12s	3852.73s
Euclidean	0.01s	0.57s	23.89s	247.58s
MRDTW (*node*=5)	2.59s	3.21s	4.49s	14.25s
MRDTW(*node*=10)	2.68s	3.11s	3.73s	8.78s

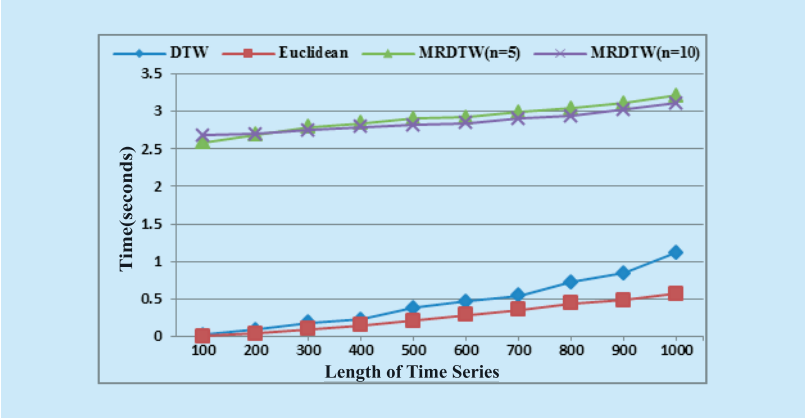

Fig.10 *The efficiency of four methods on small time series*

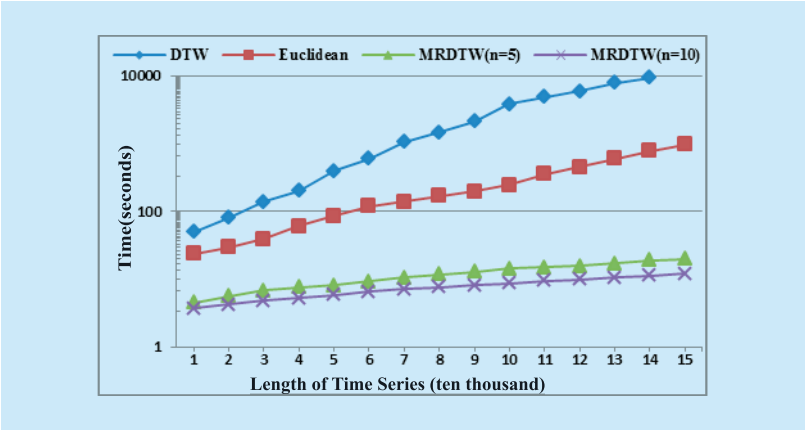

Fig.11 *The efficiency of four methods on large time series (The y-axis has log-log scaling.)*

values of 5 and 10 respectively) when the length of time series achieved 168,578. A sample of the experiment results can be seen in Table I.

In table I, DTW and Euclidean have similar execution times for the 100 points time series and Euclidean is more quickly for its simple calculation. Due to the allocation of Map tasks need to consume a certain time in cluster, MRDTW is more slow than DTW and Euclidean when the length of time series is less than 1000. The MRDTW is very suitable to deal with large time series and with the rapid increase in the length of time series, the growth of execution time is very gentle. Also with the increase of nodes, the efficiency of the MRDTW algorithm can be further raised.

It is important to note that Fig.10 is scaled normally, but for convenience of illustration the *y*-axis of Fig.11 has log-log scaling.

6.2 Experiment for long time series

From the above experiment, it is confirmed that MRDTW is not suitable for short time series. So the datasets have been changed, and only long time series have been selected. Some examples for the dataset are shown in Fig.12, and all data series are Z-normalized. Random Walk dataset [24]: To provide the scalability of the experiment, a large number of series are generated by the equation: $t_{i+1}=t_i+random(N(0,1))$, where $random(N(0,1))$ is a random value drawn from a normal distribution. The length of each series is about 100,000.

The efficiency and accuracy will be compared between MRDTW, SDTW [16] and FastDTW [14] which are all optimized methods for DTW. The MRDTW used a distributed parallel mechanism to optimize the similarity comparison. Unlike MRDTW, SDTW used linear piecewise to represent original firstly, and then computed the similarity between compressed series. FastDTW used a multilevel approach that recursively projects a solution from a coarse resolution. The experimental results are shown in Fig.13.

It is can be seen from Fig.13 that when radius is 0 the FastDTW get the quick response time, and with the radius increased, the response time become longer. For SDTW, the larger the compression ratio is, the faster execution time will be got. But as the compression ratio grows, the accuracy of SDTW will be declined rapidly because of its approximate to original series. When the number of nodes is set at 5, MRDTW is not superior to other algorithms. But when the number achieve to 10, it is obviously that with the length of time

series become longer, the execution efficiency is the best between the three methods.

6.3 Accuracy of MRDTW

The accuracy of the improved DTW algorithm can be measured by evaluating how much the improved algorithm's warping path distance differs from the standard DTW's warping path distance. For example, the error of an approximate DTW algorithm can be calculated by the following equation [18]:

$$Error\ of\ warp\ path = \frac{approxDist - s\tan dDist}{s\tan dDist} \quad (6)$$

In this experiment the different numbers of nodes are set, and through comparing the classification results to evaluate the accuracy of MRDTW algorithm. In this experiment, the Electrocardiogram (ECG) [25] dataset is considered. The dataset is about electrocardiogram data recorded from human subjects which contains 250 samples per second. The length of each series is about 20,000.

From the Fig.15, it can be seen that the accuracy of MRDTW is determined by the number (n) of nodes. With the n increased, the accuracy of classification results will be declined. The reason is that the Reduce operations will become complex when the n is become larger. Choosing bad path in the process of Reduce will lead to wrong classification results. Still, the results of the experiment are also can be acceptable. Take SDTW to compare, when its efficiency equal to the MRDTW with n=16, the accuracy of SDTW will drop to 50%. From the results of the experiment, it can be concluded that the number of nodes is the bottleneck to enhance the accuracy of MRDTW. The solution to this problem is to improve the Reduce algorithm to find the optimal path from a great many intermediate results.

VII. Conclusions

In this paper we introduced the MRDTW algorithm, a linear approximation of dynamic time warping (DTW). Unlike the previous

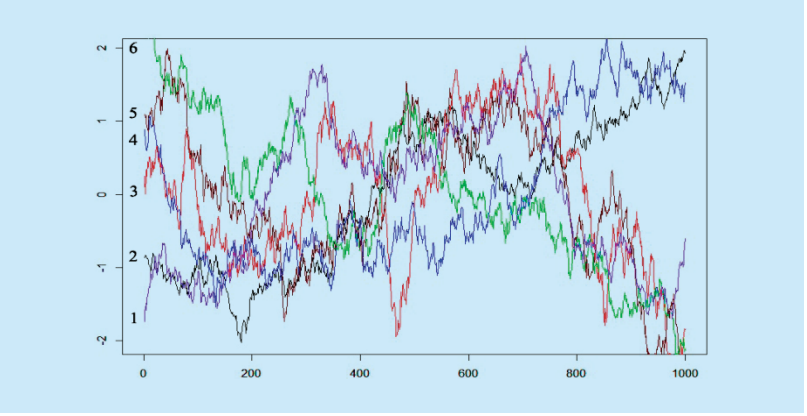

Fig.12 *Example series for Random Walk datasets (In order to illustrate conveniently, only parts of the length were plotted)*

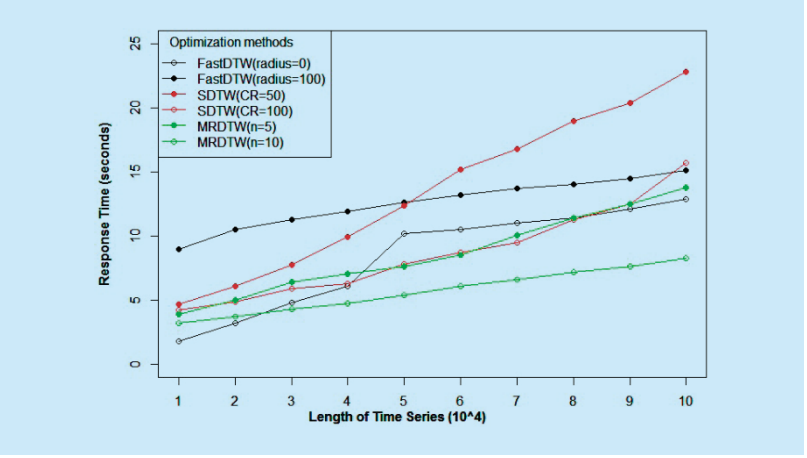

Fig.13 *The experimental results of long time series (Radius is used to increase the chances of finding the optimal solution. CR is the abbreviation of compression ratio, and n is the number of nodes)*

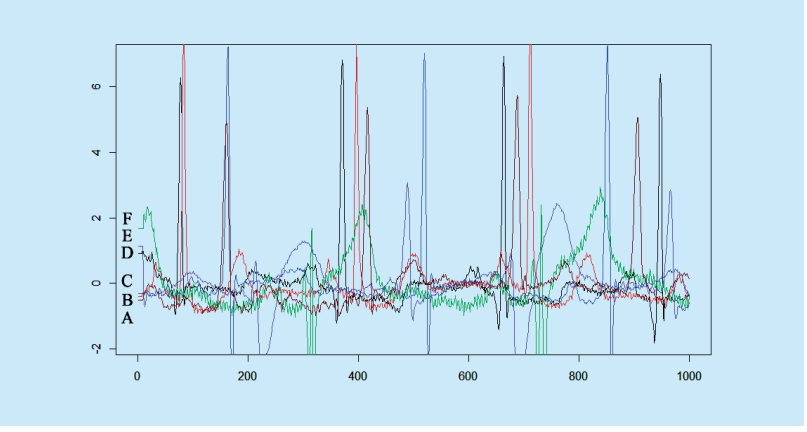

Fig.14 *Example series for ECG datasets (In order to illustrate conveniently, only parts of the length were plotted)*

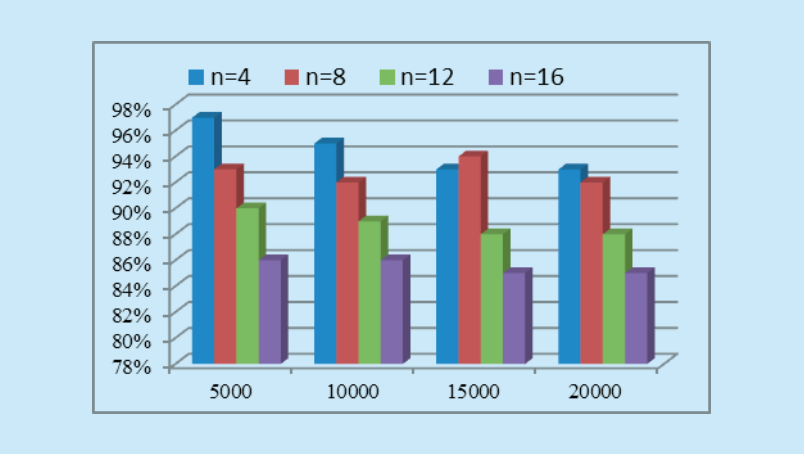

Fig.15 *The classification results of MRDTW (The numbers of x-axis is the length of series. Y-axis represents the accuracy of classification, and n is the number of nodes)*

algorithm made DTW faster, our algorithm focus on parallelization optimization, and use the MapReduce computing framework to implement in a cluster with 4 nodes or more. Experiments show that the MRDTW is very suitable for analyzing large time series. With the increase of the points of time series, the execution time of most of algorithms is exponential growth, but the MRDTW is only linear growth and retained high accuracy compared to other methods.

The primary constraint of the MRDTW algorithm is that the sub-sequences are calculated separately on each node and it will produce a large number of intermediate results. The results will be transmitted between the network, so it will reduce the efficiency and consume the bandwidth. Therefore, the improvement of the algorithm to reduce the possible warping paths will be discussed in future work.

Acknowledgements

Many people have made invaluable contributions, especially, I greatly appreciate my wife and my newborn baby Yin Zhihan. This work was supported in part by National High-tech R&D Program of China under Grants No. 2012AA012600, 2011AA010702, 2012AA01A401, 2012AA01A402; National Natural Science Foundation of China under Grant No.60933005; National Science and Technology Ministry of China under Grant No.2012BAH38B04; National 242 Information Security of China under Grant No. 2011A010.

References

[1] Das G, Lin K-I, Mannila H, Renganathan G, Smyth P: Rule Discovery from Time Series. In: KDD: 1998. 16-22.

[2] Catalano J, Armstrong T, Oates T: Discovering patterns in real-valued time series: Springer; 2006.

[3] Keogh EJ, Pazzani MJ: An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In: KDD: 1998. 239-243.

[4] Debrégeas A, Hébrail G: Interactive Interpretation of Kohonen Maps Applied to Curves. In: KDD: 1998. 179-183.

[5] Dean J, Ghemawat S: MapReduce: simplified data processing on large clusters. Communications of the ACM 2008, 51(1):107-113.

[6] Zhang Y, Glass JR: An inner-product lower-bound estimate for dynamic time warping. In: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on: 2011. IEEE: 5660-5663.

[7] Alon J, Athitsos V, Yuan Q, Sclaroff S: A unified framework for gesture recognition and spatio-temporal gesture segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 2009, 31(9):1685-1699.

[8] Keogh E, Wei L, Xi X, Vlachos M, Lee S-H, Protopapas P: Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. The VLDB Journal—The International Journal on Very Large Data Bases 2009, 18(3):611-630.

[9] Huber-Mörk R, Zambanini S, Zaharieva M, Kampel M: Identification of ancient coins based on fusion of shape and local features. Machine vision and applications 2011, 22(6):983-994.

[10] Chadwick NA, McMeekin DA, Tan T: Classifying eye and head movement artifacts in EEG Signals. In: Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on: 2011. IEEE: 285-291.

[11] Berndt DJ, Clifford J: Using Dynamic Time Warping to Find Patterns in Time Series. In: KDD workshop: 1994. Seattle, WA: 359-370.

[12] Adams N, Marquez D, Wakefield G: Iterative deepening for melody alignment and retrieval. 2005.

[13] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S: Locally adaptive dimensionality reduction for

indexing large time series databases. In: ACM SIGMOD Record: 2001. ACM: 151-162.
[14] Salvador S, Chan P: Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis 2007, 11(5):561-580.
[15] Chu S, Keogh EJ, Hart D, Pazzani MJ: Iterative Deepening Dynamic Time Warping for Time Series. In: SDM: 2002.
[16] Keogh EJ, Pazzani MJ: Scaling up dynamic time warping to massive datasets. In: Principles of Data Mining and Knowledge Discovery. Springer; 1999: 1-11.
[17] Keogh E, Ratanamahatana CA: Exact indexing of dynamic time warping. Knowledge and information systems 2005, 7(3):358-386.
[18] Sakurai Y, Yoshikawa M, Faloutsos C: FTW: fast similarity search under the time warping distance. In: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems: 2005. ACM: 326-337.
[19] Shieh J, Keogh E: iSAX: disk-aware mining and indexing of massive time series datasets. Data Mining and Knowledge Discovery 2009, 19(1):24-57.
[20] White T: Hadoop: The definitive guide: O'Reilly Media, Inc.; 2010.
[21] Chodorow C: Introduction to MongoDB. In: Free and Open Source Software Developers' European Meeting (FOSDEM): 2010.
[22] Fink E, Pratt KB, Gandhi HS: Indexing of time series by major minima and maxima. In: Systems, Man and Cybernetics, 2003 IEEE International Conference on: 2003. IEEE: 2332-2335.
[23] Kadous MW: Learning Comprehensible Descriptions of Multivariate Time Series. In: ICML: 1999. 454-463.
[24] Assent I, Krieger R, Afschari F, et al. The TS-tree: efficient time series search and retrieval[C]// Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM, 2008: 252-263.
[25] Moody G B, Mark R G. A new method for detecting atrial fibrillation using RR intervals[J]. Computers in Cardiology, 1983, 10: 227-230.

Biographies

YIN Hong, received the M.S. in School of Computer in National University of Defense Technology in 2011 and is currently a Ph.D. candidate at this University, Changsha, China. He was a teacher at Military College of Economics in 2006-2008.His research interests include satellite fault diagnosis, cloud computing, massive data mining.Email: yinhonggfkd@aliyun.com.

YANG Shuqiang, he was born in 1969. Ph. D. He is a PhD supervisor and professor of Computer Science University of Defense Technology. He also is the senior member of CCF. His research interests include database and distributed computing. Email: sqyang9999@126.com.

MA Shaodong, received the B.Eng. degree in Mobile Telecommunication Technology and the Ph.D. degree in Electronic Engineering both from the University of Hull, Hull, UK.Shaodong became an IEEE member since 2008. His primary research interestsinclude the applications of non-linear control techniques in acoustic communication signal processing, integrated healthcare monitoring and adaptive resonance tuning for wave energy harvesting. Currently he is doing post-doctorate research at the University of Hull, UK. Email: shaodongma@outlook.com.