PESC: A Parallel System for Clustering ECG Streams Based on MapReduce

Lin Yang, Jin Zhang, Qian Zhang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
Email: {lyangab, jinzh, qianzh}@ust.hk

***Abstract*—Nowadays, cardiovascular disease (CVD) has become a disease of the majority. As an important instrument for diagnosing CVD, electrocardiography (ECG) is used to extract useful information about the functioning status of the heart. In the domain of ECG analysis, cluster analysis is a commonly applied approach to gain an overview of the data, detect outliers or pre-process before further analysis. In recent years, to provide better medical care for CVD patients, the ECG-oriented telehealth system has been widely used. However, the extremely large volume and high update rate of data in the telehealth system has made cluster analysis challenging work. In this paper, we design and implement a novel parallel system for clustering massive ECG stream data based on the MapReduce framework. In our approach, a global optimum of clustering is achieved by merging and splitting clusters dynamically. Meanwhile, a good performance is gained by distributing computation over multiple computing nodes. According to the evaluation, our system not only provides good clustering results but also has an excellent performance on multiple computing nodes.**

I. Introduction

According to the latest research from the medical society, cardiovascular disease (CVD) has become a disease of the majority. Nowadays, more than one in three people are suffering this kind of disease in the United States [1]. As an important instrument for diagnosing CVD, the electrocardiography (ECG), a transthoracic interpretation of the electrical activity of the heart over a period of time, can be utilized to extract useful information about the functional status of the heart [2]. To help clinicians better utilize the ECG data, a variety of systems have been proposed, especially in recent years, as healthcare costs continue to increase, resources within the health sector are being redirected from hospital based care to provide increased primary and home based care, which contributes to the rise of cardiac telehealth systems[3]. These cardiac telehealth systems have proven to be able to provide a high-quality, personalized, real-time and long-term ambulatory monitoring of chronic CVD patients, while overcomes barriers of time, cost, and distance. Cardiac telehealth systems often consist in outfitting patients with portable, miniaturized and wireless sensors and devices that are capable to measure and report cardiac signals to telehealth providers. Owing to the fact that some chronic CVDs have low prevalence but high risk, the new-generation cardiac telehealth systems also tend to transmit the ECG data to a remote server which performs complex real-time analysis via wireless networks to provide high-quality interaction and real-time intervention for adverse cardiac events.

In the domain of ECG analysis, clustering is a commonly applied approach to gain an overview of data, detect outliers or pre-process before further analysis. As an important research topic, a proliferation of researches have been done in this area [4], [5], [6], [7], [8]. However, all of these works are built on a assumption that the ECG data is small enough to fit into the memory and will not be updated, But this assumption will not hold any more under the scenario of the cardiac telehealth systems.

As mentioned before, to provide high-quality interaction and real-time intervention for adverse cardiac events, the new-generation telehealth systems are designed to provide continuous monitoring services in a real-time manner. As a result, the ECG data will be transmitted to remote server continuously with a high update rate, which makes the ECG clustering in remote server a stream clustering problem. As an important topic in data mining, plenty of literature has been proposed in stream clustering: BIRCH [9], which uses the Cluster Feature as a compact representation of cluster, is regarded as one of the most primitive works in this area. M. Ester *et al.* [10] address this problem by designing an incremental clustering algorithm based on DBSCAN algorithm. D. Pham *et al.* [11] combines the idea of ”cluster jumping” with K-Means to cluster data incrementally. CluStream [12] is a framework for clustering evolving data streams, which could get a clustering result during a specific time. However, none of these works have taken the special properties of ECG data into account, thus they can not be applied in the cardiac telehealth systems.

In addition, the large number of user and the high data update rate in telehealth systems converge to the fact that the data volume in the system will be extremely large, therefore a framework is required to manage such massive data. Although there are many candidates, the MapReduce framework proposed by J. Dean *et al.* [13] has attracted great attention from both academia and the industry in recent years and its open-source implementation, namely Hadoop [14], becomes the first choice for managing big data. Although MapReduce has lots of excellent advantages, it is not easy work to design a system based on the MapReduce framework. In order to better utilize the power of MapReduce, the designer has to design the workflow carefully according to the MapReduce programming model to distribute as much computation work as possible.

In this work, we designed a parallel system for clustering massive ECG stream data and implemented it on top of MapReduce. Our system performs the clustering by adopting an ECG-oriented metric and tries to achieves the global optimum by merging and splitting dynamically. The evaluation on a Hadoop cluster with 8 nodes shows that our system can not only provide a good clustering result but also has an excellent performance on multiple nodes.

The rest of this paper is organized as follows: In Section II, we present an overview of our system and indicate the challenges behind our work. In Section III, we introduce the ECG-oriented metric used throughout the whole system. In Section IV, we provide an in-depth discussion about the stream clustering algorithm. In Section V, we describe how to distribute our algorithm over Hadoop cluster. Section VI reports the evaluation result and Section VII concludes the paper.

II. System overview and challenges

Under the scenario of our work, portable sensors are deployed to collect ECG data from users and upload the data to our system automatically and continuously by using wireless network technologies. The ECG data then arrives in our system as a stream with a high update rate. As soon as the data arrives in our system, a stream clustering algorithm is performed immediately to cluster the newly arrived data properly with respect to the existing data. The clustering result can be provided to clinicians to help in diagnosis or delivered to another system for further analysis. To achieve this, several challenges need to be conquered.

ECG-oriented metric design:

The primary step of cluster analysis is to define the basic processing unit. For ECG data, as it is actually a combination of electrical deflection caused by the heartbeat, it can be divided into several intervals according to the heart rate, each of which is called as PT interval in this paper and used as the basic processing unit for our cluster analysis. Figure 1 presents a standard PT interval. As showed in Figure 1, the PT interval actually consists of P wave, QRS complex, T wave and segments between them. Each wave has some feature points—-the beginning point, the peak and the end point. The PT interval also has some special properties: Firstly, considering the nature of ECG, some regions of the PT interval are more important than others. Second, since each PT interval corresponds to a heart beat, the PT intervals could be of various lengths. Finally, as a result of the biological diversity, the morphological characteristic of PT intervals diversifies with different people and even with same person in different situations.

Considering these properties, we propose an ECG-oriented metric, which defines how to measure the dissimilarity between two unequal-lengthed PT intervals.

Stream clustering algorithm:

Different from the clustering of unchanged data, stream clustering poses several additional requirements: under the stream clustering scenario, data will arrive continuously, which means that the algorithm has to generate results in a very limited time. Moreover, the undergoing changes may cause existing clusters to emerge, merge and split, so the algorithm should be aware of these changes.

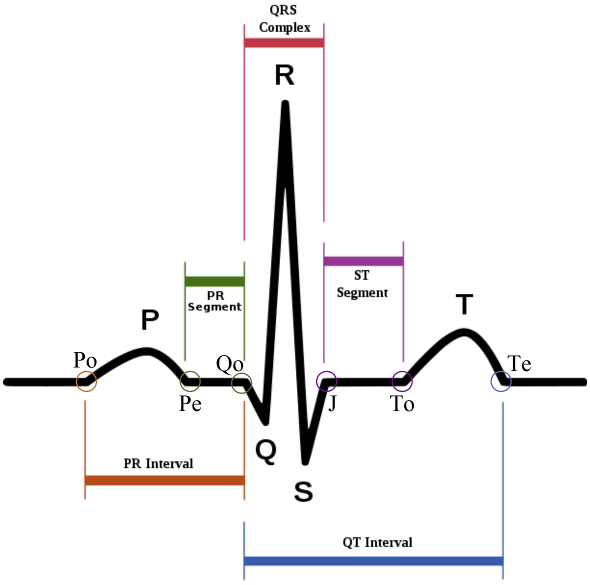

Fig. 1. Standard ECG with labels.

In our work, we introduce a new stream clustering algorithm for ECG stream data and empirically demonstrate its practical performance. The key idea of our algorithm is that as the new data arrives, it is clustered into several new clusters without initially taking the existing data into account. The connectivity and distortion of these new clusters and existing clusters will then be evaluated globally. Those clusters which have high connectivity would be merged into one, and the clusters which have high distortion would be divided into smaller clusters. This approach iterates as new batch of data comes in.

Speeding up with MapReduce:

To conquer the challenge derived from scaling our system over massive data, we choose to implement our system on top of MapReduce. MapReduce is actually a framework for processing sophisticated parallel problems, in which the Map is used to execute independent computation on a part of the data in parallel while the Reduce is responsible for ”summing up” the result of the Map. However, due to the inherent nature of our workflow, not all the computation could be distributed, thus deciding which part to distribute becomes a vital issue for our system performance.

To improve the performance as much as possible, we carefully design our workflow according to the MapReduce model in order to distribute most of the computation among the cluster. For some parts of the workflow which are dependent on the others, we break them into smaller granularity and try to execute the independent parts in parallel and run the dependent parts in sequence.

III. ECG-Oriented Metric

As mentioned in section II, the primary step of cluster analysis is to define the basic processing unit. In our system, the PT interval is used as the basic unit of clustering. So, as soon as a batch of ECG data arrives, a conventional wavelet transformation based algorithm is applied to find the feature points. According to these feature points, the ECG data could easily be interpreted into PT intervals.

After that, we define how to measure the dissimilarity between these basic processing units. Considering the special properties of PT intervals, this is not easy work.

The primary challenge deriving from the special properties of PT interval is that they are time series data of unequal length. This makes some conventional metrics like Euclidean metrics inapplicable since they could only handle data of equal length. So, to measure the dissimilarity between PT intervals properly, we first align these PT intervals by adopting the Dynamic Time Warping (DTW) algorithm [15]: Given two sequences of time series data M and N, with length m and n respectively, the DTW algorithm would yield an optimal warping path between the two data by using the dynamic programming approach. The warping path WP is a set of tuples which is defined below:

$$WP = \{(i,j) \mid 0 \le i \le m,\ 0 \le j \le n\} \tag{1}$$

where each tuple (i,j) in the warping path implies that the i^{th} element in the data M should be aligned with the j^{th} element in the data N. So, with this warping path, these two time series data M and N could be aligned easily.

Another issue raised from the special properties of PT interval is that considering the nature of ECG, some regions of the PT interval are more important than others. Taking this into account, we divide the PT interval into three segments and assign each segment with a heuristic weight, that is:

$$\begin{cases} s_1 = [P_o, Q_o],\ w_1 = 0.15 \\ s_2 = [Q_o, J],\ w_2 = 0.5 \\ s_3 = [J, T_e],\ w_3 = 0.35 \end{cases} \tag{2}$$

where the P_o is the beginning point of P wave, Q_o is the beginning point of Q wave, J is the end point of S wave, the T_e is the end point of T wave, and the w_1, w_2 and w_3 is the heuristic weight of each segment respectively. All of these points are shown in Figure 1.

Combining these two ideas, we get the procedure of measuring the dissimilarity between PT intervals: Firstly, a standard PT interval would be synthesized, and all of the PT intervals, including the synthesized one, would be divided into three segments: s_1, s_2, s_3 according to the equation (2). Then, the DTW algorithm is applied between each PT interval and the standard PT interval on $s1$, $s2$ and $s3$ respectively to find the warping paths. By aligning the s_1, s_2 and s_3 respectively, all the PT intervals would be aligned with the standard PT interval. After that, the dissimilarity between two PT intervals can be defined as the sum of the weighted Euclidean distance of the aligned segment.

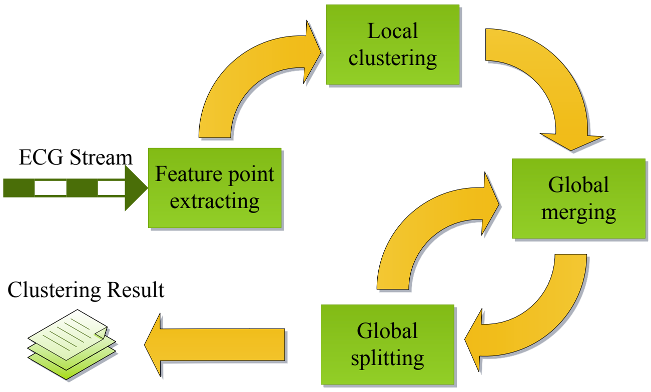

Fig. 2. Algorithm overview.

That is, given two PT intervals Q and R, which have been aligned with the standard PT interval, the dissimilarity $Dis(Q,R)$ between these two PT intervals is defined as:

$$Dis(Q,R) = \sum_{i=1,2,3} w_i * dis(s_{iq}, s_{ir}) \tag{3}$$

where s_{iq} and s_{ir} are the aligned segments i in Q and R respectively, and the $dis(X,Y)$ is the Euclidean distance between equal-length data X and Y.

IV. Stream Clustering

The design preference of providing continuous monitoring services in the ECG telehealth system has made the data arrive in the system as a stream with high update rate. This nature of this stream requires the clustering algorithm not only to be able to generate results in a limited time but also be aware of the undergoing changes. Under these concerns, we present our stream clustering algorithm, which could generate results in an incremental manner by dynamically merging and splitting clusters. Figure 2 gives an overview of the whole clustering algorithm. In this approach, as a batch of ECG stream data arrives, an iteration of clustering would be triggered and the whole iteration can be divided into three steps: local clustering, global merging and global splitting.

Step 1 is called local clustering. In this step, the ECG data has already been interpreted into PT intervals, and these PT intervals would be divided into several groups randomly and uniformly. Then, for each group, a K-Means algorithm is applied to generate k clusters respectively, where the parameter k is used to control the granularity of local clustering. After the clustering, the cluster features of each cluster would be computed, the cluster feature CF is actually the representation of a cluster, which is defined as below:

$$CF(C) = (c,\ e,\ r,\ lr) \tag{4}$$

where the c is the centre of cluster C, e is the centre of the Euclidean space, r is the average distance between centre c and the PT intervals in this cluster, and the lr is the maximal distance between centre c and the PT intervals in the cluster.

Some PT intervals that should belong to the same cluster, might be divided into different groups and thus be clustered

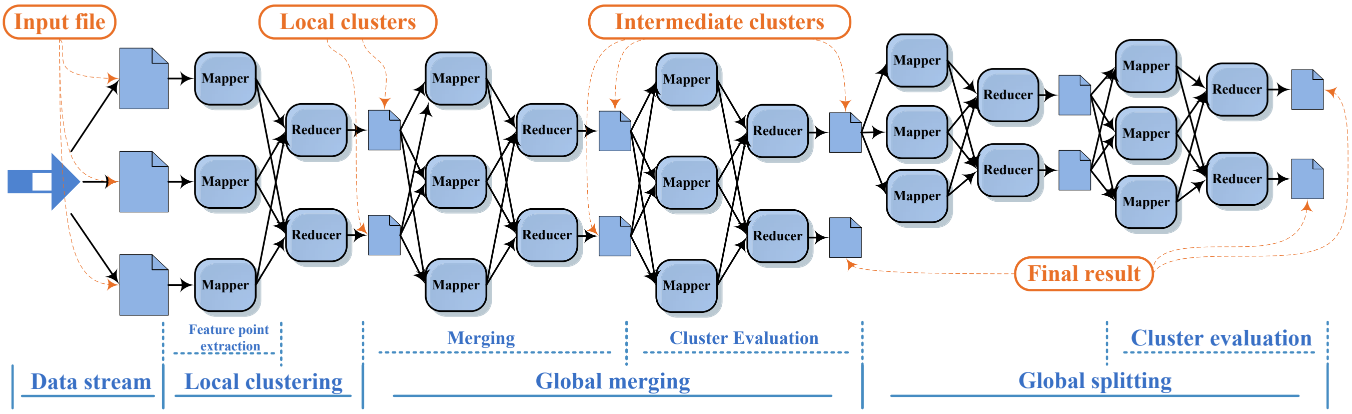

Fig. 3. The overview of system.

separately which eventually leads to their misplacement. A global merging and splitting is performed after the local clustering to address this problem.

Step 2 is the global merging. In this phase, we first define the connectivity con between two clusters Q and T as below:

$$con(Q,T) = \frac{(r_q + r_t)}{Dis(c_q, c_t)} \tag{5}$$

where the r_q and r_t is the average radius r of cluster Q and cluster T, the c_q and c_t is the centre of cluster Q and T, and $Dis(c_q, c_t)$ is the dissimilarity measurement between c_q and c_t

All the connectivity between each pair of clusters generated by the local clustering and existing clusters which are generated by the previous iterations would be calculated. After that, all the cluster pairs whose connectivity is satisfied the following condition would be merged into one cluster:

$$con(Q,R) \geq \lambda \tag{6}$$

where λ is a pre-defined parameter to control the granularity of global merging.

Although the global merging would redress the misplacement caused by local clustering, but it also might degrade the cluster quality by merging heterogeneous clusters together. So after the global merging, a cluster evaluation would be performed on each cluster generated from global merging to evaluate the quality of the clusters. The quality of a cluster is represented by its distortion error, and the evaluation function of distortion error is defined as below:

$$DST(C) = \frac{w_1 * r + w_2 * Dis(c,e)}{lr} \tag{7}$$

where the $DST(C)$ is the distortion error of cluster C, the w_1 and w_2 is the heuristic weight which empirically set to be 0.3, 0.4 respectively in our system, the r is the average distance of cluster C, the c is the centre of cluster C, the e is the centre of Euclidean space, and $Dis(c,e)$ is the dissimilarity measurement between c and e.

After the cluster evaluation, all the clusters which satisfy the following splitting condition would be applied with a K-Means algorithm to split into two smaller clusters:

$$DST(C) \geq \tau \tag{8}$$

where τ is a pre-defined parameter to control the granularity of global splitting.

After the global splitting, the whole clustering iteration ends. The final clustering result is saved in the file system in order to cluster newly arrived data incrementally in the next iteration.

V. Speedup with MapReduce

The data we are processing is stream data with high update rate, which implies that the data volume in our system would increase exponentially as time goes by. To handle such massive data properly, we implement our system on top of the MapReduce framework. For better utilization of the power of MapReduce, we designed our system workflow carefully in order to distribute as much computation as possible. Figure 3 shows the workflow of our system on top of MapReduce.

In this workflow, as the workload of feature point extraction is relatively lightweight, it has been combined into the local clustering step. It is obvious that the job of both feature point extraction and local clustering are independent of each others, so it would be easily distributed by performing the job in the MAP.

However, the situation becomes more complex when it comes to global merging. As it would cross check all the clusters to calculate the connectivity, global merging seems to be unable to distribute among clusters. After carefully investigating the workflow of global merging, we have found that this step could be divided into two smaller steps: generating merging plan and merging. The first step would calculate the pair-wise connectivity between clusters to generate a merging plan. Due to its serial nature, this step would only be able to run in sequence. Meanwhile, the second step conducts the actual merging according to the merging plan assigned to it and updates the cluster feature for new clusters. As it only concerns the data contains in the merging plan, this step could be performed in MAP and run in parallel

In the global merging, we actually have two MapReduce jobs: one for global merging and another one for cluster evaluation. Although the latter would only update the cluster feature for newly-generated clusters, it would not be combined with the previous MapReduce job. Recall that for new clusters, we have to calculate the centre and the average radius of cluster. This requires to scan over the whole data twice at least, the first one finds the centre, and the next one calculates the average radius with respect to the centre. However, this could not be done in a single REDUCE, because there is only one iteration allowed in the REDUCE as considering the input data, the volume of REDUCE would be extremely large. As a result, the latter MapReduce job for cluster evaluation is inevitable.

For the global splitting, as it only concerns a part of the whole data and is thus independent of the others, this step could be distributed easily. As mentioned before, an inevitable cluster evaluation job would be performed after the global splitting.

VI. Evaluation

We implemented our system on top of Hadoop 0.20.2. As described in the previous section, this system consists of multiple MapReduce jobs, and all the jobs are implemented in Java. To prepare the data for evaluation, we cooperate with New Element Medical Ltd.,Co, Shenzhen, China, which provides us a real ECG database collected from hundreds of users, including users of different age, gender and heart status. In the database, there are more than 1,500 ECG data files, each of which contains about 2,800 PT intervals. So, approximately, there are 4,356,800 PT intervals in total. With these data, we conduct our evaluation on a homogeneous Hadoop cluster, in which each node runs a CentOS release 5.5 and is equipped with an Intel Xeon CPU X3430 @ 2.4 GHz with 4 cores, 4 GB of RAM and 160 GB of hard disk. For all the experiments, we set the parameters in our system as below:

$$\{k = 4,\ \lambda = 1.0,\ \tau = 0.4\} \tag{9}$$

where k, λ and τ are the parameter which control the granularity of local clustering, global merging and global splitting respectively.

For the evaluation, we first conduct a baseline performance evaluation, in which we cluster an ECG stream consisting of 1500 ECG data files and report the clustering result. Then, the scalability characteristic of our system with respect to the number of computing cores is evaluated. Apart from this, we also evaluate the performance of our system with respect to different block size of HDFS.

Baseline performance evaluation:

In this evaluation, we use our system to cluster an ECG stream consisted of all the ECG data files from the real ECG database. The whole data is clustered into 4 clusters and some important statistics of each cluster are listed in table I, where the cluster size is the number of PT intervals in the cluster, the total distance is the sum of distances between the cluster centre and the PT intervals in the cluster, the average radius is the average distance between the cluster centre and the PT intervals in the cluster, and the maximal radius is the maximal distance between the cluster centre and the PT intervals in this cluster. To provide an intuitive view of the clustering results, Figure 4 plots all the PT intervals belong to the last cluster in the clustering result by overlapping them in a single figure.

TABLE I
Clustering Result

ID	Cluster size	Total distance	Average radius	Maximal radius
C1	2753836	130889825.08	47.53	186.50
C2	1922412	107558951.40	55.95	202.10
C3	2003	55743.49	27.83	88.04
C4	11771	289684.31	24.61	110.13

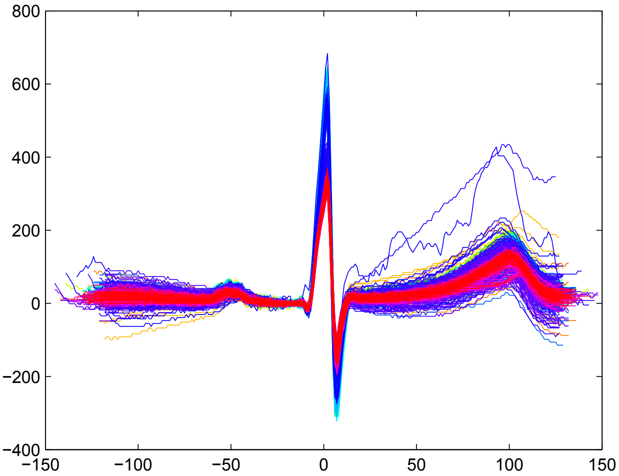

Fig. 4. One of the clustering results.

Scalability with computing cores:

To evaluate the scalability of our system, we conduct an experiments which processes 300 ECG data files in each iteration of clustering and run this experiment on a cluster scale from 8 cores to 32 cores (each node in our cluster has 4 cores, and there are 8 nodes in total). Figure 5 shows the average running time of our system with different number of computing cores. It is obvious that the total running time is decreasing significantly as the number of computing cores grows, which shows that our system scales well on multiple computing cores.

The average running of each step of our stream clustering algorithm is also evaluated. As showed in Figure 6, we have found that the global merging is the most time consuming job in the whole process. This is because the workload is quite heavy since it has to cross check all the existing clusters in a serial manner and when it executes the actual merging work, it causes a lot of IO operations. We also notice that the global merging benefits significantly when increasing the number of cores. This could be explained by observing the fact that the most work in the global merging are done in the reduce stage. Since the number of reducers is relative to

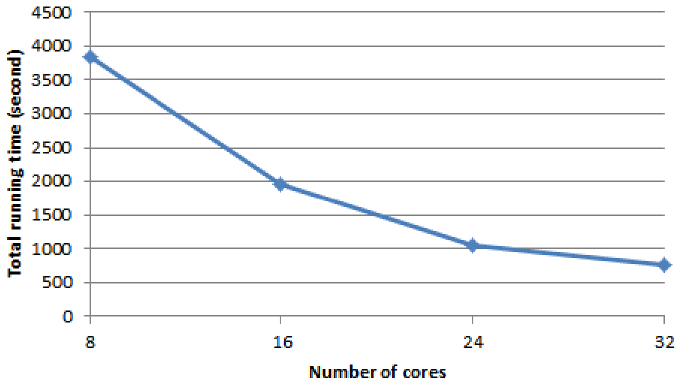

Fig. 5. Scalability of PESC system.

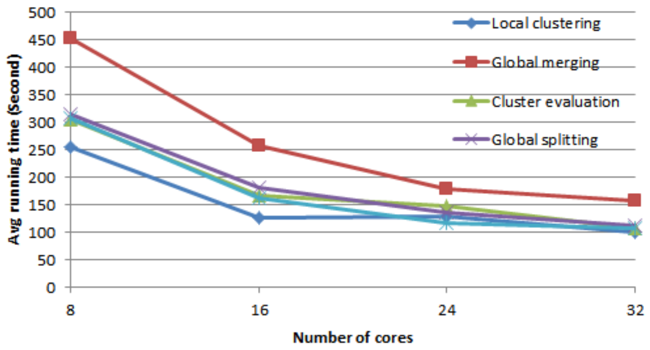

Fig. 6. Scalability of steps in stream clustering algorithm.

the number of computing cores, there would be more reducer running in parallel as the number of computing cores grow, which could speed up the global merging significantly.

Optimization with block size:

As mentioned at the beginning of this section, the database we are using consists of thousands of small files. However, processing large volumes of small files in Hadoop could be quite frustrating. One of the most significant reasons is that for each file in the HDFS which is smaller than a block would still occupy a block. That is, as the default block size of HDFS is 64 MB, if there are 10,000 files in HDFS, each of which is smaller than 64 MB, then these files will occupy $10,000 * 64MB = 625GB$ in the HDFS, which would degrade the performance significantly. To evaluate the influence of block size to our system performance, we run our system with different block size and the result is shown in Figure 7. It is found that when the block size of HDFS is significantly larger than the size of the files stored in it, the total running time is quite high since there are lots of useless IO and network overheads. However, due to the fact that each map task usually processes a block of input at a time, if the block size are noticeable smaller than the input files size, then there will be plenty of map tasks, each of which imposes extra bookkeeping overhead, which eventually degrades the performance.

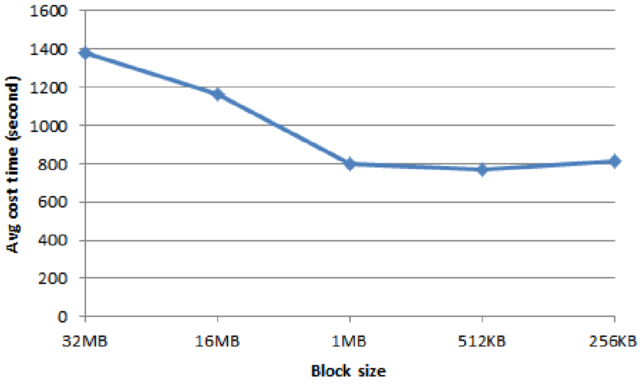

Fig. 7. PESC performance V.S. HDFS block size.

VII. Conclusion

We have designed and implemented the PESC, a parallel system for massive ECG stream clustering. Our system would perform the clustering by adopting an ECG-oriented metric and try to achieve the global optimum by merging and splitting dynamically. The evaluation result shows that our system would not only provide a good clustering result but also produces an excellent performance on multiple computing nodes.

References

[1] V. L. Roger, A. S. Go, and et al., "Heart disease and stroke statistics2012 update," *Circulation*, 2011.

[2] M. Bashir, D. Lee, M. Akasha, G. Yi, E.-j. Cha, J.-w. Bae, M. Cho, and K. Ryu, "Highlighting the current issues with pride suggestions for improving the performance of real time cardiac health monitoring," in *Information Technology in Bio- and Medical Informatics, ITBAM 2010*.

[3] G. Demiris, N. Charness, E. Krupinski, D. Ben-Arieh, K. Washington, J. Wu, and B. Farberow, "The role of human factors in telehealth," *Telemedicine and e-health*, vol. 16, no. 4, pp. 446–453, 2010.

[4] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo, "Clustering ecg complexes using hermite functions and self-organizing maps," *Biomedical Engineering, IEEE Transactions on*, vol. 47, no. 7, pp. 838–848, 2000.

[5] F. Sufi, I. Khalil, and A. Mahmood, "A clustering based system for instant detection of cardiac abnormalities from compressed ecg," *Expert Systems with Applications*, vol. 38, no. 5, pp. 4705–4713, 2011.

[6] R. Ceylan, Y. Özbay, and B. Karlik, "A novel approach for classification of ecg arrhythmias: Type-2 fuzzy clustering neural network," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6721–6726, 2009.

[7] G. Bortolan, R. Degani, and J. Willems, "Ecg classification with neural networks and cluster analysis," in *Computers in Cardiology 1991, Proceedings*. IEEE, 1991, pp. 177–180.

[8] G. Bortolan and J. Willems, "Diagnostic ecg classification based on neural networks." *Journal of Electrocardiology*, vol. 26, p. 75, 1993.

[9] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *SIGMOD Rec.*, vol. 25, no. 2, pp. 103–114, Jun. 1996.

[10] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, ser. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 323–333.

[11] D. Pham, S. Dimov, and C. Nguyen, "An incremental k-means algorithm," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 218, no. 7, pp. 783–795, 2004.

[12] C. Aggarwal, J. Han, J. Wang, and P. Yu, “A framework for clustering evolving data streams,” in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[13] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] T. White, *Hadoop: The definitive guide*. Yahoo Press, 2010.

[15] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.