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 cardiac 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 used 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 home based care, which drives the rise of cardiac telehealth system [3]. The cardiac telehealth systems are able to provide a high-quality, personalized, real-time and long-term ambulatory monitoring service for chronic CVD patients, while overcome barriers of time, cost, and distance. The cardiac telehealth system often consists 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 system prefers to transmit the ECG data to a remote server via wireless networks to provide high-quality interaction and real-time intervention for adverse cardiac events.

After the ECG data is collected, complex analysis will be performed to extract useful information. In this domain, clustering is a commonly applied approach to gain an overview of data, detect outliers or pre-process before further analysis, *i.e.,* clustering could help clinicians to gain an overview of the ECG data, and locate abnormal cardiac events quickly without reviewing all the data manually. 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 do not hold in cardiac telehealth systems.

As mentioned before, to provide high-quality interaction and real-time intervention for adverse cardiac events, some telehealth systems are designed to provide continuous monitoring services in a real-time manner. In these systems, the ECG data will be transmitted to remote server continuously, 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: 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 speciality of the ECG data into account, thus they can not be applied in the cardiac telehealth scenario.

In addition, the large number of patients and the high data update rate in cardiac telehealth system 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 build a system based on the MapReduce framework. In order to better utilize the power of MapReduce, the workflow has to be designed carefully according to the MapReduce

programming model so as 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 achieves the global optimum by merging and splitting dynamically. The evaluation on a Hadoop cluster with 32 computing cores shows that our system would 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

In our system, portable sensors are deployed to collect ECG data from patients. In order to provide high-quality monitoring service and real-time intervention for adverse cardiac events, the collected ECG data will be transmitted to a remote analysis server automatically and continuously by using wireless network technologies. In the perspective of the analysis server, the ECG data will arrive as a stream, and thus as soon as the data arrives, a stream clustering 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 heart beat, it can be divided into several intervals according to the heart beats, each of which is called as QRS interval in this paper and used as the basic processing unit for our cluster analysis. Figure 1 presents a standard QRS interval. As showed in Figure 1, the QRS interval actually consists of P wave, QRS complex, T wave and segments between them. Each wave has some characteristic points—-the beginning point, the peak and the end point. The QRS interval also has some special properties: Firstly, considering the nature of ECG, some regions of the QRS interval are more important than others. Second, since each QRS interval corresponds to a heart beat, the QRS intervals could be of various lengths. Finally, as a result of the biological diversity, the morphological characteristic of QRS 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 QRS intervals.

Stream clustering algorithm:

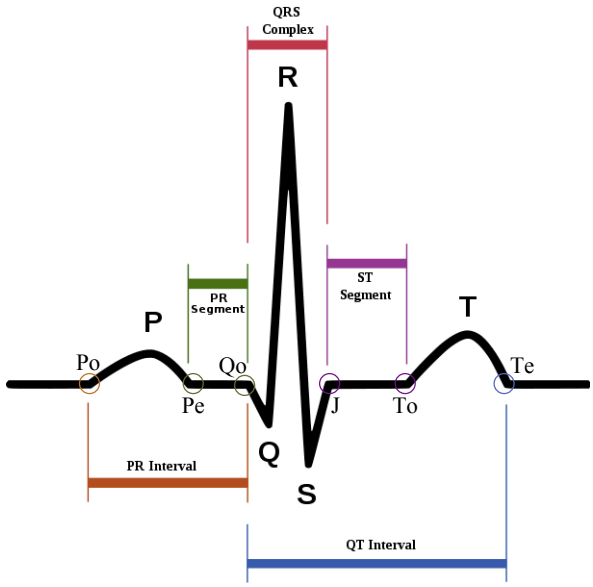

Fig. 1. Standard ECG with labels.

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.

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 local 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, while the clusters which have high distortion would be split 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 intermediate results from 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 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 execute the independent parts in parallel and run the dependent

parts in sequence.

III. ECG-Oriented Metric

The primary step of clustering is to define the basic processing unit. In our system, the QRS interval is used as the basic unit. As soon as a batch of ECG data arrives, a conventional wavelet transformation based algorithm [15] is applied to find the characteristic points. According to these characteristic points, the ECG data could easily be interpreted into QRS intervals. Since this algorithm has been well-discussed in [15], we will put more attention on the following steps.

The next step is to define the measurement of the dissimilarity between QRS intervals. The primary challenge deriving from the special properties of QRS interval is that they are time series data of unequal length. This makes some conventional metrics like Euclidean metric inapplicable since they can only handle data of equal length. So, to measure the dissimilarity between QRS intervals properly, we first align these QRS intervals by adopting the Dynamic Time Warping (DTW) algorithm [16], which could make two unequal-length QRS intervals align with the same length without losing the morphological characteristic: 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:

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

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 QRS interval is that considering the nature of ECG, some regions of the QRS interval are more important than others when measuring the similarity. Taking this into account, we divide the QRS interval into three segments and assign each segment with a heuristic weight, that is:

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

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 showed in Figure 1.

Combining these two ideas together, we get the procedure of measuring the dissimilarity between QRS intervals: Firstly, a standard QRS interval would be synthesized, and all of the QRS 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 QRS interval and the standard QRS 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 QRS intervals would be aligned with the standard QRS interval. After that, the dissimilarity between two QRS intervals can be defined as the sum of the weighted Euclidean distances of the aligned segments.

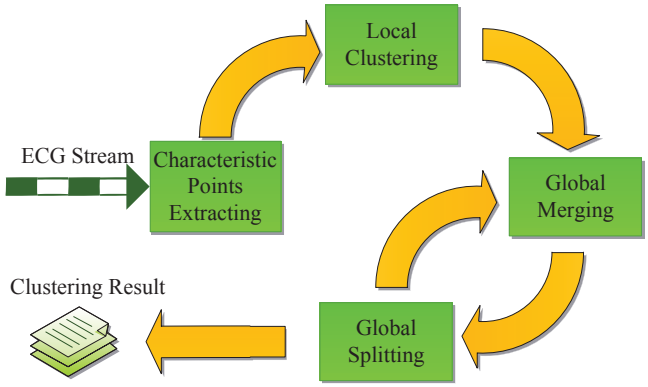

Fig. 2. Algorithm overview.

That is, given two QRS intervals Q and R, which have been aligned with the standard QRS interval, the dissimilarity $Dis(Q, R)$ between these two QRS 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 new-generation cardiac telehealth system has made the data arrive as a stream. This 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 this algorithm. In our approach, as soon as a batch of new ECG stream data arrives, an iteration which consists of four steps would be triggered:

Step 1 is the characteristic points extraction. In this step, a conventional wavelet transformation based algorithm [15] is performed to find the characteristic points, and according to these characteristic points, the ECG data could be easily interpreted as QRS intervals, which is the basic processing unit for following steps.

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

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

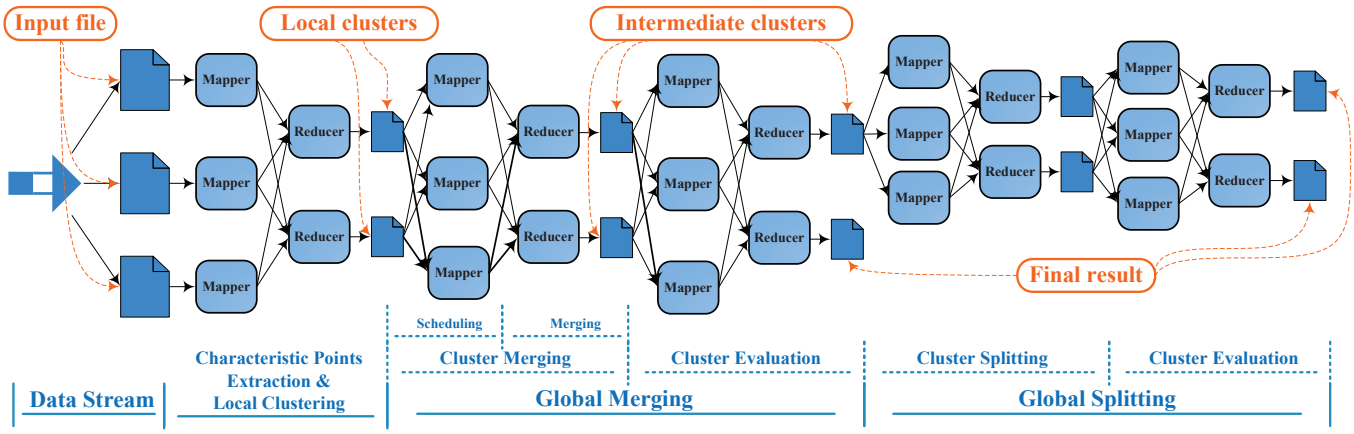

Fig. 3. The overview of system.

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 QRS intervals in this cluster, and the lr is the maximal distance between centre c and the QRS intervals in the cluster.

In this step, some QRS intervals that should belong to the same cluster, might be divided into different groups and thus be clustered separately, which eventually leads to their misplacement. To address this problem, the global merging and splitting is performed after the local clustering.

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 connectivities between each pair of clusters generated from the local clustering and existing clusters generated from the previous iterations will be calculated. After that, all the clusters whose connectivities satisfy 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 address 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 to evaluate the quality of each cluster. The quality of cluster is represented by its distortion error, which is defined as below:

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

where $DST(C)$ is the distortion error of cluster C, the w_1 and w_2 are the heuristic weights, r is the average distance of cluster C, c is the centre of cluster C, 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 will be saved in the file system for clustering next batch of data incrementally.

V. Speedup with MapReduce

As 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, thus we implement our system on top of the MapReduce framework in order to handle such massive data properly. Figure 3 shows the workflow of our system in the perspective of MapReduce. The four steps of our stream clustering algorithm: characteristic points extraction, local clustering, global merging and global splitting, are implemented as five MapReduce jobs, and the intermediate results of each jobs are stored in HDFS as input for next job. To speed up the clustering, we designed our system carefully to distribute as much computation as possible. The following paragraphs introduce more details about our implementation.

In the workflow of our algorithm, since the workload of characteristic points extraction is relatively lightweight, it has been combined with the local clustering and implemented as a single MapReduce job. In this MapReduce job, since the characteristic points extraction and local clustering only concern about the local data stored in current machine, the computation could easily perform independently in the MAP phase, and the intermediate results are aggregated and pushed to HDFS for further use in the REDUCE phase.

However, situation becomes more complex when it comes to global merging. As shown in Figure 3, the global merging is

actually implemented as two MapReduce jobs: one for cluster merging and another for cluster evaluation.

In the cluster merging, we merge clusters generated from previous local clustering step with existing clusters. Since we use the cluster feature as representation in this operation, there is not much heavy work. However, since this operation has to cross-check all the clusters to calculate the connectivity, it seems to be unable to run in parallel. After carefully investigating the workflow of cluster merging, we have found that this step could be divided into two smaller steps: scheduling and merging. The scheduling step computes the pair-wise connectivities between clusters to schedule a merging plan and the latter step conducts the actual merging according to the merging plan. Due to its serial nature, the scheduling step will only be able to run in sequence, but the merging step only involves the data containing in its assigned merging plan, so it could be performed in a single MAP phase and thus run in parallel.

After the cluster merging, the quality of each cluster has to be evaluated in order to check whether this cluster needs to be split. Although it only requires update the cluster feature for newly-generated clusters, it could not be combined with the previous MapReduce job. Recall that for a cluster feature, we have to compute the centre and the average radius of cluster. This requires to scan the whole data twice at least: the first scan finds the center, the second scan calculates the average radius with respect to this centre. However, multiple scan is forbidden in a single MapReduce job. The reason for this is that multiple scan implies the caching of whole data, but this is impossible under the scenario of massive data. As a result, we have to perform an inevitable MapReduce job for cluster evaluation after merging.

For the global splitting, like the local clustering, it only need local data stored in current machine. Therefore, this step could be easily performed in single MapReduce job. However, the quality of cluster need to be updated after splitting, thus an inevitable cluster evaluation job would be performed after the cluster splitting job.

VI. Evaluation

To conduct the evaluation, we implemented our system on Hadoop 0.20.2 and conduct a series of experiments 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. To get a convincing result, we cooperate with New Element Medical Ltd.,Co, Shenzhen, China [17], which provides us a real ECG database collected from hundreds of patients, including patients of different age, gender and heart status. In this database, there are more than 1,500 ECG data files, each of which contains about 2,800 QRS intervals. So, approximately, there are 4,356,800 QRS intervals in total. For all the experiments, we set the parameters in our system as below:

$$\begin{aligned} &\{w_1 = 0.15,\ w_2 = 0.5,\ w_3 = 0.35\ \} \\ &\{k = 4,\ \lambda = 1.0,\ \tau = 0.4\ \} \end{aligned} \tag{9}$$

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

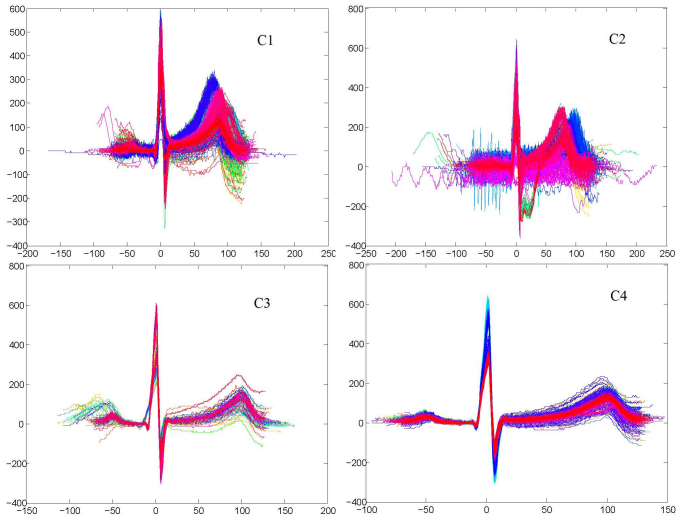

Fig. 4. One of the clustering results.

where w_1, w_2, and w_3 are the heuristic weights used in the similarity measurement, and k, λ and τ are the parameter which control the granularity of local clustering, global merging and global splitting respectively.

We first conduct a baseline performance evaluation, in which we cluster an ECG stream data which consists 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. After that, we 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 QRS intervals in the cluster, the total distance is the sum of distances between the cluster centre and the QRS intervals in the cluster, the average radius is the average distance between the cluster centre and the QRS intervals in the cluster, and the maximal radius is the maximal distance between the cluster centre and the QRS intervals in this cluster. To provide an intuitive view of the clustering results, Figure 4 plots all the clusters in the Table I in an overlapped way. We could see that the quality of $C1$, $C3$ and $C4$ are good since the morphological characteristic of each QRS interval in each cluster is well-matched, even some of them have obvious drift on different segments. However, $C2$ has introduced some outliers. After investigating the data

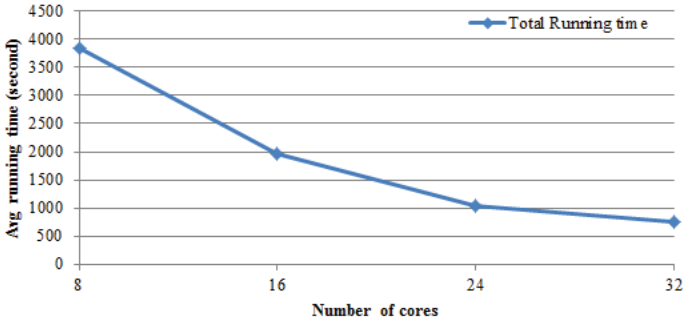

Fig. 5. Scalability of PESC system.

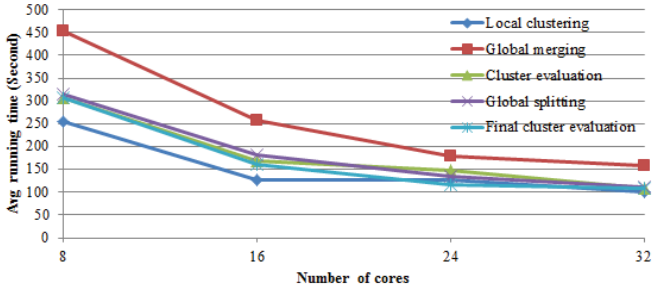

Fig. 6. Scalability of steps in stream clustering algorithm.

carefully, we find that this is caused by interference and human factor, *i.e.* incorrect usage of cardiac sensors. Since we are using a real ECG database, this kind of issues are inevitable.

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. This shows that as the workload has been distributed and can be done in parallel, thus the performance gains a significant improvement when using more computing resource.

Besides, the average running time of each step in our stream clustering algorithm is also evaluated. As showed in Figure 6, we notice that the global merging is the most time-consuming job during the whole process. This is because the workload is relatively heavy since this step has to cross-check all the clusters in a serial manner and would causes a lot of IO operations when performing the cluster merging operation. We also find that the global merging benefits significantly when increasing the number of computing cores. This could be explained by observing the fact that the most work in the global merging is done in the reduce phase, and since the number of reducers is depend on the number of computing cores, thus if we use more computing cores, there will be more reducer running in parallel, which eventually could speed up the global merging significantly.

Optimization with block size:

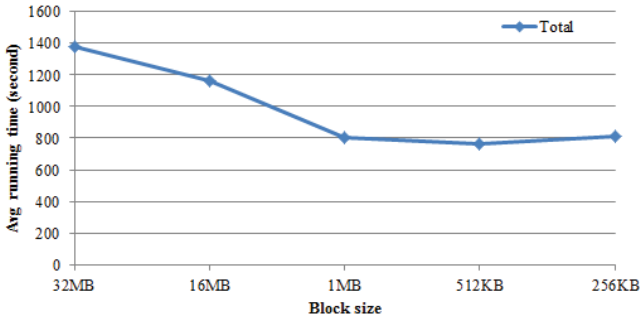

Fig. 7. PESC performance V.S. HDFS block size.

As mentioned at the beginning of this section, the database we are using consists of thousands of ECG data files. However, processing large volumes of small files in Hadoop could be a quite frustrating work due to its design principle [14]. To evaluate one of the most important issue—-block size of HDFS, we conduct an evaluation with different block sizes and report the result in Figure 7. We notice that the performance gains a exponentially improvement when the block size decreases from $32MB$ to $1MB$, but it drops when the block size continues decreasing to $256KB$. 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 will result in a lot of useless IO operation and networks overhead and thus eventually degrade the performance significantly. 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 would also degrades the performance.

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] C. Li, C. Zheng, and C. Tai, "Detection of ecg characteristic points using wavelet transforms," *Biomedical Engineering, IEEE Transactions on*, vol. 42, no. 1, pp. 21–28, 1995.
[16] 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.
[17] "New element medical," http://www.newelementmedical.com/.