[Supplementary File] Multi-Threaded Hierarchical Clustering by Parallel Nearest-Neighbor Chaining

Yongkweon Jeon and Sungroh Yoon, *Senior Member, IEEE*

S1 Survey of Clustering Methods

We can categorize existing clustering methods as follows [1]: (1) partitioning (2) hierarchical (3) density-based, and (4) grid-based methods.

Partitioning methods find (usually non-overlapping) partitions of the input data and assess them by certain criteria. The most widely used partitioning method is k-means clustering (also known as Lloyd's algorithm). Given k, the number of clusters to find, k-means clustering determines k centers of clusters and updates the membership of each cluster iteratively. It is a special case of the expectation-maximization algorithm, and its result varies depending on the choice of initial center locations. It is also often challenging to decide the value of k in advance. Nonetheless, k-means clustering is very popular in data analysis when the number of features is moderate. Further, k-means clustering is appropriate for parallelization, since there is no dependence among clusters on computing center-object distance. There have been various parallelization approaches for k-means clustering based on map-reduce [2], [3], message-passing interface [4], [5], and general-purpose GPU [6]. Variations of k-means clustering (such as k-medoids and k-centers) and related methodologies (such as self-organizing map [7] and CLARANS [8]) have also been parallelized [9], [10].

Hierarchical methods aim at discovering hierarchical structures from input data. The most widely used approach is the hierarchical agglomerative clustering (HAC) algorithm. It iteratively groups two closest clusters, eventually grouping input data into one. The outcome from HAC is a graphical record of clustering history called *dendrogram*. Other examples of hierarchical approaches include BIRCH [11], CURE [12], ROCK [13], Diana [14], and CHAMELEON [15]. It is known that the HAC algorithm is difficult to parallelize while maintaining its requirements: (1) exactness and (2) support of various linkage methods. This challenge originates largely from the need of maintaining a distance matrix globally. Consequently, existing parallel HAC approaches [9], [16], [17] are heuristics or limited to a specific linkage method.

Density-based clustering methods find dense regions appearing in the input data space based on connectivity and density functions. Examples include DBSCAN [18], OPTICS [19], DenClue [20], and recent large-scale pre-clustering methods such as canopy clustering [21] are related. In contrast to partitioning methods (such as k-means) that usually tend to find spherical regions, density-based methodologies are not limited to specific cluster shapes. Typically, density-based methods are scalable to large-scale data, since they usually employ a local search technique to discover dense regions. In addition, multiple dense regions can be sought simultaneously, giving ample opportunities for parallelization. For instance, PDBSCAN [22] is a message-passing-based parallelization version of DBSCAN targeted for distributed-memory environments. In PDBSCAN, the input data set is split into multiple partitions, each of which is assigned to a node. Each node performs DBSACN on its assigned partition, and in the end, the results from individual nodes are collected and merged into a global clustering result. Most of the existing parallelization techniques for density-based clustering work similarly.

A grid-based approach creates a finite number of grid cells in the input space and then carries out clustering on the grid structure. Existing methods include STING [23], WaveCluster [24], and CLIQUE [25]. Grid-based methods typically have advantages in terms of time and space efficiency, given that the efficiency of grid-based clustering algorithms depends not on the input quantity but on the grid quantity. This type of clustering is particularly useful for mining spatial data. Parallel grid-based clustering algorithms divide data grids into a number of subgrids, process each subgrid, and merge individual results to get the final clustering result [26], [27], [28].

• *Y. Jeon and S. Yoon are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-744, Republic of Korea.*
E-mail: sryoon@snu.ac.kr

S2 Additional Details of generalPar

The GENERALSEQ and GENRALPAR algorithms [29] presented in Section 4.8 of the main text are one of the the best hierarchical agglomerative clustering (HAC) methodologies to date in terms of exactness and generality and are widely used as a standard benchmark. Although there have been many parallelization methods proposed, GENERALPAR is the most widely applicable 'exact' technique to the best of the authors' knowledge. This is why we included GENERALSEQ and GENRALPAR in our experiments.

In order to understand why our approach outperforms GENERALPAR, we first explain GENERALSEQ and GENERALPAR briefly. GENERALSEQ allocates a separate priority queue for each cluster. The priority queue of a cluster stores the pairwise distances between that cluster and all the other clusters. The reason for maintaining a priority queue per cluster is to reduce the search space for the closest pair. The front entry in the priority queue of a cluster contains the information about the cluster that is the closest to this cluster.

Recall that the core algorithm of hierarchical agglomerative clustering consists of two main steps that are iteratively executed in turn: (A) cluster merge and (B) distance matrix update. GENERALSEQ implements the HAC algorithm using the notion of per-cluster priority queue as follows. In step A, GENERALSEQ examines the top entry of each priority queue, finds the closest pair among all the top entries, and merges it. In step B, GENERALSEQ updates the distance matrix and constructs the priority queue for the newly merged cluster. The worst-case time complexity of GENERALSEQ is $O(n^2 \log n)$, where n is the number of objects, and $O(\log n)$ originates from the cost for priority queue operations. Note that this complexity is inferior to the worst-case time complexity of the NN-chain-based HAC: $O(n^2)$.

GENERALPAR parallelizes GENERALSEQ as follows. In step A, GENERALPAR utilizes multiple threads in order to examine the front of multiple priority queues simultaneously. More specifically, each thread gets assigned a priority queue to process and finds the closest pair therein (*i.e.* 'local' closest pair). Then, GENERALPAR relies on the parallel reduction to find the 'global' closest pair: it keeps comparing two local closest pairs at a time to find the global closest pair in the end. In step B, there is no dependence between operations, and the update operations are distributed to multiple threads and processed independently. To construct the priority queue for the newly merged cluster, GENERALPAR uses a concurrent priority queue to support multi-threading.

We now explain why GENERALPAR is slower than the proposed method. In essence, GENERALPAR suffers from the multithreading overhead more significantly than the proposed method. More specifically,

1) For GENERALPAR, there can be load imbalance among threads during step A (cluster merge). Given that the number of priority queues (*i.e.* clusters) are usually greater than that of threads, different threads can get assigned different quantities of priority queues. In addition, the spawn timing of threads may differ from thread to thread, causing load imbalance. Further, during parallel reduction, threads that do not participate in the reduction may remain idle. In contrast, the proposed method does not suffer from the load imbalance issue in step A, because there is no common resource that needs to be distributed to threads.
2) There are overheads incurred by associating a priority queue for every cluster. The matrix update step itself in GENERALPAR is identical to that in the proposed method, but there is extra cost for constructing and operating priority queues in case of GENERALPAR. Additionally, the concurrent priority queue management used by GENERALPAR inevitably needs serialization.
3) GENERALPAR must have a barrier between steps A and B, increasing the possibility of incurring idle threads. In contrast, for the proposed method, steps A and B are executed asynchronously.

References

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.

[2] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*. Springer, 2009, pp. 674–679.

[3] P. Zhou, J. Lei, and W. Ye, "Large-scale data sets clustering based on mapreduce and hadoop," *Journal of Computational Information Systems*, vol. 7, no. 16, pp. 5956–5963, 2011.

[4] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-Scale Parallel Data Mining*. Springer, 2000, pp. 245–260.

[5] S. Kantabutra and A. L. Couch, "Parallel k-means clustering algorithm on nows," *NECTEC Technical journal*, vol. 1, no. 6, pp. 243–247, 2000.

[6] R. Farivar, D. Rebolledo, E. Chan, and R. H. Campbell, "A parallel implementation of k-means clustering on gpus." in *PDPTA*, 2008, pp. 340–345.

[7] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[8] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 5, pp. 1003–1016, 2002.

[9] K. J. Kohlhoff, M. H. Sosnick, W. T. Hsu, V. S. Pande, and R. B. Altman, "Campaign: an open-source library of gpu-accelerated data clustering algorithms," *Bioinformatics*, vol. 27, no. 16, pp. 2322–2323, 2011.

[10] A. Ene, S. Im, and B. Moseley, "Fast clustering using mapreduce," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 681–689.

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

[12] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[13] ——, "Rock: A robust clustering algorithm for categorical attributes," in *Data Engineering, 1999. Proceedings., 15th International Conference on*. IEEE, 1999, pp. 512–521.
[14] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.
[15] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.
[16] D.-J. Chang, M. M. Kantardzic, and M. Ouyang, "Hierarchical clustering with cuda/gpu." in *ISCA PDCCS*, 2009, pp. 7–12.
[17] Z. Du and F. Lin, "A novel parallelization approach for hierarchical clustering," *Parallel Computing*, vol. 31, no. 5, pp. 523–527, 2005.
[18] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, vol. 96, 1996, pp. 226–231.
[19] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *ACM SIGMOD Record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.
[20] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, vol. 98, 1998, pp. 58–65.
[21] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 169–178.
[22] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," in *High Performance Data Mining*. Springer, 2002, pp. 263–290.
[23] W. Wang, J. Yang, and R. Muntz, "Sting: A statistical information grid approach to spatial data mining," in *VLDB*, vol. 97, 1997, pp. 186–195.
[24] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *VLDB*, vol. 98, 1998, pp. 428–439.
[25] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data," *Data Mining and Knowledge Discovery*, vol. 11, no. 1, pp. 5–33, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10618-005-1396-1
[26] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 443–452.
[27] C. Xiaoyun, C. Yi, Q. Xiaoli, Y. Min, and H. Yanshan, "Pgmclu: A novel parallel grid-based clustering algorithm for multi-density datasets," in *Web Society, 2009. SWS'09. 1st IEEE Symposium on*. IEEE, 2009, pp. 166–171.
[28] H. Zhang, Y. Zhou, J. Li, X. Wang, and B. Yan, "Analyze the wild birds migration tracks by mpi-based parallel clustering algorithm," in *Advanced Data Mining and Applications*. Springer, 2010, pp. 383–393.
[29] C. Olson, "Parallel algorithms for hierarchical clustering," *Parallel computing*, vol. 21, no. 8, pp. 1313–1325, 1995.