A New Parallelization Model using Complex Grid Partitioning for Density-based Spatial Clustering Algorithm on a Multi-Core CPU

Tatsuhiro Sakai[1,2,a)] Keiichi Tamura[1,b)] Kohei Misaki[1] Hajime Kitakami[1,c)]

Abstract: Recently, the sizes and volumes of spatial databases have been increasing not only because of the popularity of geographical data, but also because of the popularity of geosocial media. A density-based spatial clustering algorithm is one of the simplest but most robust clustering techniques for geospatial data. Therefore, the speedup for the processing of density-based spatial clustering algorithms is one of the most important challenges. In this paper, we propose a new parallelization model using complex grid partitioning for density-based spatial clustering algorithm on a multi-core CPU. The main technique of the new parallelization model is that it forms complex spatial partition, n order to speed up the processing. The experimental results show that our new model outperforms a conventional data parallelization model.

1. Introduction

With the increasing interest in big data, the use of geospatial databases for ICT (information and communications technology) has received much attention in recent years. The clustering technique for geospatial data is one of the most well studied techniques because it allows us to reveal spatial relevance of geospatial data. Clustering techniques for geospatial data differ from traditional clustering techniques (e.g., k-means method) only in that clusters for geospatial data do not always form circles. For example, contaminated land sites form arbitrary shapes from a satellite observation.

A density-based spatial clustering algorithm is one of the simplest but most robust clustering techniques for geospatial data. The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm was first introduced by Ester et al. [1][2], and it applies a density-based concept of spatial clusters. Spatial clusters are recognized by analyzing the density of data points. Areas with a high density of data points are spatial clusters, whereas areas with a low density are not. DBSCAN can discover spatial clusters with arbitrary shapes. Therefore, many methods apply this algorithm to geospatial databases because spatial clusters in geospatial databases are not circular.

In this paper, we focus on the speedup of the DBSCAN algorithm. The goal of this study is to develop a new parallel-processing parallelization model for DBSCAN on a multi-core CPU. Currently, PCs and workstations have one or more multi-core CPUs. A multi-core CPU is a single microprocessor with two or more independent CPU cores on a die, which are the units that read and execute program instructions. It is necessary to develop an efficient parallelization model for spatial clustering techniques on a multi-core CPU.

To parallelize the DBSCAN algorithm, the proposed parallelization model is based on the master-worker model using data parallelism. In data parallelism, an entire geospatial database is divided into two or more sub-databases called partitions using grid partitioning. To extract a spatial cluster that is spread over several grids, each gird contains a replication of geospatial data beyond the borders of the grid. Moreover, to reduce the number of replications, the proposed parallelization model utilizes complex grid partitioning. In complex grid partitioning, a complex grid is composed of highly dense adjacent grids. Composing a complex grid reduces the number of grids; therefore, the number of replications decreases compared with simple grid partitioning. This improves the overall performance of the parallel processing.

The rest of this paper is organized as follows. In Section 2, related work is reviewed. In Section 3, a density-based spatial clustering algorithm and its algorithm are presented. In Section 4, we propose a new parallelization model for the parallel processing of DBSCAN. In Section 5, we report the experimental results. In Section 6, we conclude the paper.

2. Related Work

Recently, the parallelization model of DBSCAN for speedup of its algorithm has been proposed as the sizes and volumes of spatial databases have been increasing because of the popularity of geographical data. Xu et al. [3] proposed the parallelization model of DBSCAN on a cluster computer. The method divides an entire geospatial dataset using grid division of the space index,

[1] Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731-3194, Japan
[2] JSPS Research Fellow, Japan
[a)] da65003@e.hiroshima-cu.ac.jp
[b)] ktamura@hiroshima-cu.ac.jp
[c)] kitakami@hiroshima-cu.ac.jp

and each computer performs clustering for the divided geospatial data.

Misaki et al. [4] proposed a parallelization model for the parallel processing of DBSCAN on a multi-core CPU. In previous model, a geospatial database is divided into two or more sub-databases called partitions using grid partitioning on the basis of data parallelism. Each CPU core performs the same processing on different partitions. In the experimental results, the previous model showed the effectiveness of parallel processing in terms of speedup; however, the process for each grid partitioning is time consuming because each grid is increased in the number of replications. The proposed new model reduce the processing time because decreasing the number of replications by using complex grid partitioning.

3. DBSCAN

In this section, the definitions and algorithm of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) are briefly reviewed.

3.1 Definitions

Let GSD be the entire geospatial data, and ϵ and $MinGSD$ be user parameters. The ϵ-neighborhood of gsd_p ($gsd_p \in GSD$) $GSN_\epsilon(gsd_p)$ is defined as geospatial data within a radius of ϵ.

Definition 1 (Core geospatial data, Border geospatial data) A geospatial data gsd_p is called a core geospatial data if there is at least the minimum number of geospatial data, $MinGSD$, in the ϵ-neighborhood $GSN_\epsilon(gsd_p)$ ($|GSN_\epsilon(gsd_p)| \geq MinGSD$). Otherwise, ($|GSN_\epsilon(gsd_p)| < MinGSD$), gsd_p is called a border geospatial data.

Definition 2 (Density-based reachable) A geospatial data sequence $(gsd_1, gsd_2, \cdots, gsd_n)$ is called density-based reachable in that satisfies the following restrictions:

(1) $gsd_1, gsd_2, \cdots, gsd_{n-1}$ are core geospatial data.
(2) $gsd_{i+1} \in GSN_\epsilon(gsd_i)$.

Definition 3 (Density-based spatial cluster) A density-based spatial cluster (GSC) in a geospatial data set GSD that satisfies the following restrictions:

(1) $\forall gsd_p, gsd_q \in GSD$, if and only if $gsd_p \in GSC$ and gsd_q is density-based reachable from gsd_p, and gsd_q is also in GSC.
(2) $\forall gsd_p, gsd_q \in GSC$, there is a $gsd_o \in GSC$, and gsd_p and gsd_q are density-based reachable from gsd_o.

3.2 Algorithm

To extract density-based spatial clusters, approximate core geospatial data are appended recursively. For each geospatial data gsd_i in GSD, the following steps are executed. If gsd_i is a core geospatial data according to Definition 1, it is assigned to a new spatial cluster GSC, and all the neighbors are queued to a candidate queue Q for further processing. The processing and assignment of geospatial data to the current spatial cluster continue until Q is empty. The next geospatial data is then dequeued from Q. If the dequeued geospatial data is not already assigned to the current spatial cluster, it is assigned to the current spatial cluster. The ϵ-neighborhood of the dequeued geospatial data is then queued to Q, which puts input geospatial data into Q if they are not already in Q.

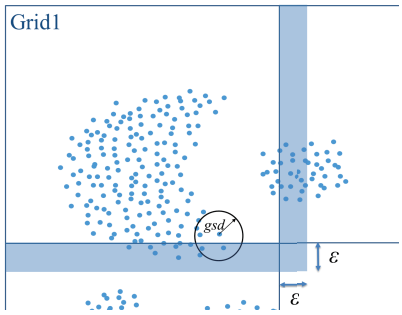


Fig. 1 Geospatial data around the border of a grid

4. Proposed Method

In this section, we propose a new parallelization model for the parallel processing of DBSCAN on a multi-core CPU.

4.1 Grid Partitioning and Dynamic Load Balancing

In this study, we focus only on the data-parallelism-based master-worker model on a multi-core CPU. In data parallelism, a geospatial database is divided into two or more sub-databases called partitions. In a multi-core CPU environment, each CPU core performs the same processing on different partitions. The proposed parallelization model utilizes grid partitioning to divide the whole database. Each data in the geospatial database is assigned to a grid that includes the data.

A processing of spatial clustering for a partition associated with a grid is referred to as a task. The master thread manages tasks using the task pool to distribute the loads dynamically. Each worker perform clustering after obtaining a task from the task pool. Finally, if the task pool is empty and each worker finishes task processing, the entire process is completed.

4.2 Data Replication

In the grid partitioning framework, we cannot determine whether a geospatial data near the border of a grid is core geospatial data or not using only the data set of the grid that contains geospatial data. In Fig. 1, even though a geospatial data gsd near the border of Grid 1 is a core geospatial data the geospatial data gsd is not identified as a core geospatial data, because some its neighbors are located in Grid 3. To determine whether a geospatial data near the border of a grid is core geospatial data or not, all grids extend only ϵ. Therefore, adjoining grids overlap. In Fig. 1, Grid 1 contains not only a set of geospatial data located in its area but also a set of geospatial data located in the area shown with a transmission color. The set of geospatial data located in the area is a replication.

4.3 Complex Grid

To reduce the number of replications, the proposed parallelization model utilizes complex grid partitioning. Fig. 2 shows an

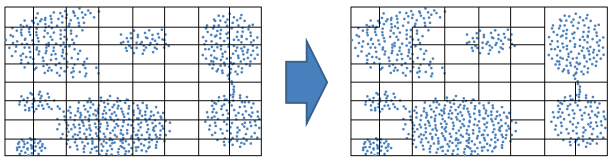


Fig. 2 Example of complex grid partitioning

example of complex grid partitioning. One of the disadvantages of simple grid partitioning is the increase in the number of replications due to merging, as shown in the left side of Fig. 2 In complex grid partitioning on the right side of Fig. 2, a complex grid is composed of highly dense adjacent grids. Composing a complex grid reduces the number of grids; therefore, the number of replications decreases compared with simple grid partitioning. This improve the overall performance of the parallel processing. Moreover, if data there is concentrated in one of the grids, the loads are not distributed. Then, if the number of geospatial data in a grid is larger than the number of the entire geospatial data divided by the number of workers, the grid is further divided.

The steps are the processing steps of creating complex grids.

(1) For each grid, the number of geospatial data in the grid is counted.
(2) For each grid, if the number of geospatial data is larger than the number of all geospatial data divided by the number of workers, the grid is further divided.
(3) For each grid, if the number of geospatial data is larger than twice the average of the number of geospatial data, the grid is labeled a dense grid. Otherwise the grid is labeled a non-dense grid.
(4) Each dense grid combines with the adjoining dense grids up to the number of all geospatial data divided by the number of workers. A set of dense grids then forms a complex grid.
(5) Each non-dense grid forms a complex grid.

4.4 Merging Clusters

To extract a spatial cluster that spread over several grids, the proposed model merges extracted spatial clusters from each partition. First, the proposed model obtains adjacent grids information from the area number of each partition and the division points for each dimension. On the basis of the information from the adjacent grids, the proposed model extracts overlapping clusters from spatial clusters in a grid and spatial clusters in grids adjacent to its grid. It is possible to extract overlapping spatial clusters because of data replication. The extracted overlapping spatial clusters are merged, and those spatial clusters become one spatial cluster. The proposed model can obtain the same as clustering results using no parallel method by the merging clusters.

4.5 Algorithm

The processing steps of the master thread and the worker threads are as follows.

A) Master Thread

(1) The master thread received a geospatial database GSD, and parameters p, ϵ, and $MinGSD$.
(2) The whole space is divided into p subspaces for each dimension. A separated space is a grid. For each geospatial data $gsd \in GSD$, the master thread assigns gsd to a grid.
(3) If the number of geospatial data in a grid is larger than the number of the entire geospatial data divided by the number of workers, the grid is further divided.
(4) The master thread calculates $GSN_\epsilon(gsd)$ for geospatial data.
(5) The master thread generates complex grids and a tack pool.
(6) The complex grid is referred to as a partition. The master puts a partition in the task pool.
(7) The master thread creates t worker threads.
(8) The master thread receives a request for task assignment from a worker thread.
(9) If the task pool is not empty, the master thread pops a task from the task pool and sends it the task to the worker thread. Otherwise, the master worker sends a wait message to the worker thread.
(10) If the master thread has sent wait messages to all the worker threads, the processing step goes to (11). Otherwise, the processing step returns to (8).
(11) The master thread sends an end message to each worker thread.
(12) The master thread receives clustering results from all the worker threads and merges clusters that spread over several grids.
(13) The master thread returns a set of spatial clusters.

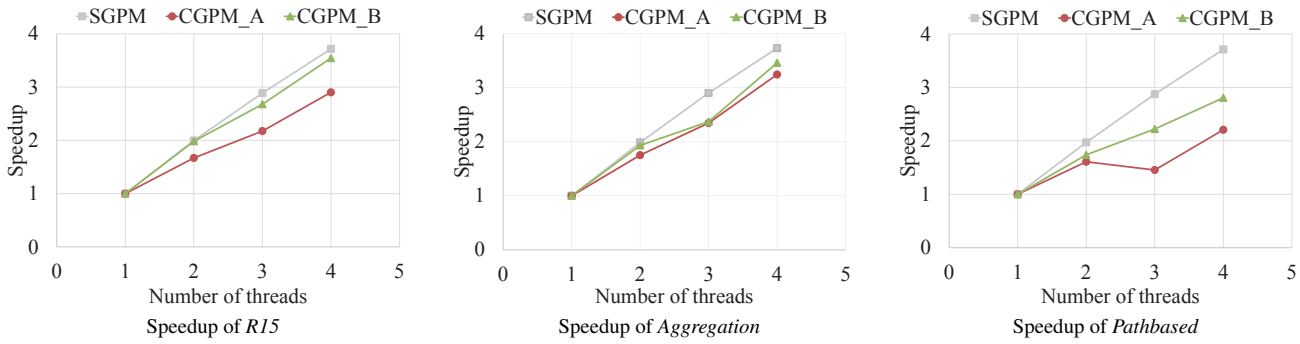
B) Worker Thread

(1) The worker thread sends a task assignment request to the master thread.
(2) If the worker thread receives a wait message, the processing step goes to (4). Otherwise, the worker thread receives a task from the master thread.
(3) The worker thread extracts spatial clusters from a partition associated with the task using the DBSCAN algorithm. The worker thread puts the clustering results of the assigned task in a result pool. The processing step returns to (1).
(4) The worker thread waits for an end message from the master thread.
(5) The worker thread sends a set of spatial clusters to the result pool.

5. Performance Evaluation

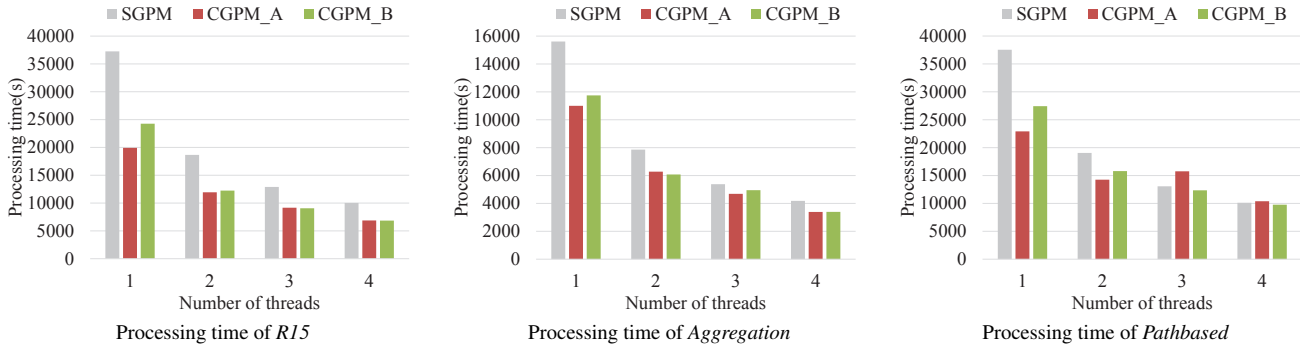
To evaluate the proposed model for parallel processing for DBSCAN on a multi-core CPU, we implemented the proposed model. We conducted an experiment with a PC with the following specifications: CPU INTEL XEON E5-1270 V2 (number of core = 4) @3.5 GHz, memory:32GB. In the experiment, we used three types of datasets: *R15*, *Aggregation* and *Pathbased*. For, each dataset, we expanded the number of geospatial data to approximately 100,000 by increasing geospatial data around each geospatial data artificially. The parameters were set to the number of initial grid divisions of each dimension $p = 8$. Moreover, the parameters were set to $\epsilon = 0.5$ and $MinGSD = 3000$ with *R15*, $\epsilon = 1.8$ and $MinGSD = 1550$ with *Aggregation*, and $\epsilon = 1.6$ and $MinGSD = 2300$ with *Pathbased* so as to be correct clustering results. We compared the results of changing the number of worker threads t from 1 to 4.

In the experiments, we measured the processing time of DB-



Speedup of *R15* Speedup of *Aggregation* Speedup of *Pathbased*

Fig. 3 Speedup for each dataset



Processing time of *R15* Processing time of *Aggregation* Processing time of *Pathbased*

Fig. 4 Processing time for each dataset

SCAN using the proposed model, which utilizes complex grid partitioning, and the previous model, which utilizes simple grid partitioning (denoted by SGPM). Moreover, we compare CGPM with chenging the condition of combining of the dense grids. One is to combine up to the number of all geospatial data divided by the number of workers (denoted by CGPM_A). The other is to combine up to the number of all geospatial data divided by four (denoted by CGPM_B). Fig. 3 shows the speedup for each dataset. The previous model obtained a higher speedup compared to the proposed model, as shown in Fig 3.

In addition, Fig. 4 shows the processing time for each dataset. The processing time of CGPM_A are faster than that of SGPM using *R15* and *Aggregation*. This is because the number of replication data is reduced by the complex grid partition. The processing time with $t = 3$ and 4 of CGPM_A is worse then that of SGPM using *Pathbased*. It is assumed that deviation of the loads occurred by conbining of dense grid.

The previous model obtained a higher speedup compared to the proposed model. It is assumed that the number of replication data with worker threads $t = 4$ is more than the number of replication data with worker threads $t = 1$, because the less the number of threads, the more combining of dense grids increase. We then considered the experimental result of CGPM_B. The speedup ratio of CGPM_B is much the same as the speedup ratio of SGPM using *R15* and *Aggregation*. The processing time of CGPM_B is faster than the processing time of SGPM. However, the processing time of CGPM_B is slower than the processing time of CGPM_A. We are necessary to develop a method for automatic setting of the combining condition for each the number of worker thread.

6. Conclusion

This paper proposed a new parallelization model on a multi-core CPU for the parallel processing of DBSCAN. The proposed parallelization model utilizes the data replication technique and complex grids in order to speed up processing time. Moreover, the proposed model reduces the number of replications owing to the complex partition grid partition. The experimental results showed that the proposed parallelization model outperforms the conventional parallelization model, which utilizes the simple grid partitioning. In our future work, we intend to discuss combining condition of the dense grids.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 16J05403 and 26330139, and Hiroshima City University Grant for Special Academic Research (General Studies).

References

[1] Ester, M., peter Kriegel, H., Sander, J. and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD-96, pp. 226–231 (1996).

[2] Sander, J., Ester, M., Kriegel, H.-P. and Xu, X.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications, *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 169–194 (1998).

[3] Xu, X., Jäger, J. and Kriegel, H.-P.: A Fast Parallel Clustering Algorithm for Large Spatial Databases, *Data Mining and Knowledge Discovery*, Vol. 3, No. 3, pp. 263–290 (1999).

[4] Misaki, K., Tamura, K. and Kitakami, H.: Parallel Processing for Density-based Clustering Algorithm on a Multi-core CPU, *Proceedings 2014 IEEE SMC Hiroshima Chapter Young Researchers' Workshop*, pp. 33–36 (2014).