Parallel Processing for Density-based Spatial Clustering Algorithm using Complex Grid Partitioning and Its Performance Evaluation

Tatsuhiro Sakai\textsuperscript{1,2}, Keiichi Tamura\textsuperscript{1}, Kohei Misaki\textsuperscript{1}, and Hajime Kitakami\textsuperscript{1}
\textsuperscript{1}Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan
\textsuperscript{2}JSPS Research Fellow, Japan

Abstract—Density-based spatial clustering algorithms, which have been well studied in database domains, are based on densities of geospatial data. 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. Therefore, the speedup for the processing of density-based spatial clustering algorithms is one of the most important challenges in many different application domains. In this paper, we propose a new parallelization model on a multi-core CPU using the spatial partition method for DBSCAN, which is one of the most fundamental algorithms for density-based spatial clustering. The new parallelization model utilizes a data replication technique and complex grids for the parallel processing of DBSCAN, in order to improve the speedup performance of parallel processing. The experimental results show that our new model outperforms a conventional data parallelization model.

Keywords: density-based spatial clustering, spatial database, parallel processing, multi-core CPU, complex grid

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. To extract clusters for geospatial data, a huge number of spatial clustering techniques have been proposed. 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. \cite{1}\cite{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. The key concept of the DBSCAN algorithm is that for each data point in a spatial cluster, the neighborhood with a user-defined radius has to contain at least a minimum number of points; i.e., the density in the neighborhood must exceed some predefined threshold.

In this paper, we focus on the speedup of the DBSCAN algorithm. The goal of this study is to develop a novel 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.

The main contributions of this study are as follows:

- To parallelize the DBSCAN algorithm, the proposed parallelization model is based on the master-worker model using data parallelism. The DBSCAN algorithm has spatial independence at the data level, because a spatial cluster can be extracted independently of the extraction of other spatial clusters. In data parallelism, an entire geospatial database is divided into two or more sub-databases called partitions using grid partitioning. A partition is assigned to a worker thread on a CPU core, and it is executed on a worker thread.
- To extract a spatial cluster that is spread over several grids, we have to calculate the density of geospatial data near the boundary of the grids correctly. Each grid contains a replication of geospatial data beyond the borders of the grid. This replication allows us to calculate the density of geospatial data near the boundary of the grids. Moreover, several spatial clusters extracted from adjacent grids are merged if they are connected.
- To reduce the number of replications, the proposed parallelization model utilizes complex grid partitioning. One of the disadvantages of grid partitioning is the increase in the number of replications due to merging. 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 novel 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 [3][4] as the sizes and volumes of spatial databases have been increasing because of the popularity of geographical data [5]. 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, and each computer performs clustering for the divided geospatial data. It is possible to perform parallel processing of clustering by using multiple computers. Moreover, research on parallel processing of DBSCAN has also been conducted on the new computing platform, example, the parallelization model using graphics processing unit (GPU) [6][7] and MapReduce [8].

Misaki et al. [9] 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 of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) are briefly reviewed.

3.1 Definitions

In DBSCAN, the $\epsilon$-neighborhood of a geospatial data is defined as geospatial data in the neighborhood of a user-defined given radius $\epsilon$.

Definition 1 ($\epsilon$-neighborhood $GSN_\epsilon(gsd)$) The $\epsilon$-neighborhood of a geospatial data $gsd_p$, denoted by $GSN_\epsilon(gsd_p)$, is defined as

$$GSN_\epsilon(gsd_p) = \{gsd_q \in GSD | dist(gsd_p, gsd_q) \leq \epsilon\},$$  

where the function $dist$ returns the distance between geospatial data $gsd_p$ and geospatial data $gsd_q$.

An example of the $\epsilon$-neighborhood of $gsd_p$ is shown on Fig. 1. On the left side of the figure, there are four geospatial data in the $\epsilon$-neighborhood of $gsd_p$. Moreover, in the right side of Fig. 1, there are three geospatial data in the $\epsilon$-neighborhood of $gsd_p$.

Definition 2 (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, $\text{MinGSD}$, in the $\epsilon$-neighborhood $GSN_\epsilon(gsd_p)$ ($|GSN_\epsilon(gsd_p)| \geq \text{MinGSD}$). Otherwise, ($|GSN_\epsilon(gsd_p)| < \text{MinGSD}$), $gsd_p$ is called a border geospatial data.

Suppose that $\text{MinGSD}$ is set to four. A geospatial data $gsd_p$ on the left side of Fig. 1 is a core geospatial data, because there are four geospatial data in $GSN_\epsilon(gsd_p)$. A geospatial data $gsd_p$ on the right side of Fig. 1 is a border geospatial data because the number of geospatial data in $GSN_\epsilon(gsd_p)$ is less than $\text{MinGSD}$.

Definition 3 (Density-based directly reachable) Suppose that a geospatial data $gsd_q$ is in the $\epsilon$-neighborhood of $gsd_p$. If the number of geospatial data in the $\epsilon$-neighborhood of $gsd_p$ is greater than or equal to $\text{MinGSD}$, i.e., if $|GSN_\epsilon(gsd_p)| \geq \text{MinGSD}$, $gsd_q$ is density-based directly reachable from $gsd_p$.

On the left side of Fig. 1, geospatial data $gsd_p$ is a core geospatial data, because $|GSN_\epsilon(gsd_p)| \geq \text{MinGSD}$. Geospatial data $gsd_1$, $gsd_2$, $gsd_3$, and $gsd_4$ are in the $\epsilon$-neighborhood of $gsd_p$. These four geospatial data are density-based directly reachable from $gsd_p$. On the other hand, on the right side of Fig. 1, geospatial data $gsd_p$ is a border geospatial data; i.e., it is not $|GSN_\epsilon(gsd_p)| \geq \text{MinGSD}$. These three geospatial data are not density-based
Definition 4 (Density-based reachable) Suppose that there is a geospatial data sequence \((gsd_1, gsd_2, \ldots, gsd_n)\), and the \((i + 1)\)-th geospatial data \(gsd_{i+1}\) is density-based directly reachable from the \(i\)-th geospatial data \(gsd_i\). The geospatial data \(gsd_n\) is then density-based reachable from \(gsd_1\).

Definition 5 (Density-based connected) Suppose that geospatial data \(gsd_p\) and \(gsd_q\) are density-based reachable from geospatial data \(gsd_o\). If \([GSN(gsd_o)] \geq \text{MinGSD}\), we denote that \(gsd_p\) is density-based connected to \(gsd_q\).

A density-based spatial cluster consists of two types of data: core geospatial data, which are mutually density-based reachable; and border geospatial data, which are density-based directly reachable from the core geospatial data. A density-based spatial cluster is defined as follows.

Definition 6 (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. \(\forall gsd_p, gsd_q \in GSC\), \(gsd_p\) is density-based connected to \(gsd_q\).

3.2 Algorithm

To extract density-based spatial clusters, approximate core geospatial data are appended recursively. A density-based spatial cluster is created using a core geospatial data first, and neighbors of the core geospatial data are then added to the cluster. For each geospatial data \(gsd_i\) in GSD, the following steps are executed. If \(gsd_i\) is a core geospatial data according to Definition 2, 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 \(\epsilon\)-neighborhood of the dequeued geospatial data is then queued to \(Q\), which puts input geospatial data into \(Q\) if they are not already in \(Q\).

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 Data Parallelism using Grid Partitioning

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. The extraction of spatial cluster can be performed in parallel using these 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. The whole space is divided into boundary boxes called grids. Each data in the geospatial database is assigned to a grid that includes the data. Fig. 2 shows an example of grid partitioning. In this example, the whole area is divided into four subareas called grids. A set of geospatial data is assigned to a partition.

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. 3, 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 of 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 \(\epsilon\). Therefore, adjoining grids overlap. In Fig. 3, Grid 1 contains
4.3 Complex Grid

To reduce the number of replications, the proposed parallelization model utilizes complex grid partitioning. Fig. 4 shows an example of complex grid partitioning. The left side of Fig. 4 shows simple grid partitioning. One of the disadvantages of simple grid partitioning is the increase in the number of replications due to merging. In complex grid partitioning on the right side of Fig. 4, 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.

Moreover, if data 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 all 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 Dynamic Load Balancing

The proposed model utilizes the task pool to distribute the loads. A processing of spatial clustering for a partition associated with a complex grid is referred to as a task. The master thread manages tasks using the task pool. Each worker performs 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.

4.5 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.

4.6 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 ∈ GSD, the master thread assigns gsd to a grid. For each grid, the number of geospatial data is calculated.
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(ksd) for geospatial data.
5. The master thread generates complex grids from a set of grids.
6. The master thread creates a task pool.
7. The complex grid is referred to as a partition. The master puts a partition in the task pool.
8. The master thread creates t worker threads.
9. The master thread receives a request for task assignment from a worker thread.
10. If the task pool is not empty, the master thread pops a task from the task pool and sends it to the worker thread. Otherwise, the master worker sends a wait message to the worker thread.
11. If the master thread has sent wait messages to all the worker threads, the processing step returns to (12). Otherwise, the processing step returns to (9).
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 \( \text{MinGSD} = 3000 \) with \( R15 \), \( \epsilon = 1.8 \) and \( \text{MinGSD} = 1550 \) with \( \text{Aggregation} \), and \( \epsilon = 1.6 \) and \( \text{MinGSD} = 2300 \) with \( \text{Pathbased} \) 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 DBSCAN using the proposed model, which utilizes complex grid partitioning (denoted by CGPM), and the previous model, which utilizes simple grid partitioning (denoted by SGPM). Fig. 6 shows the speedup for each dataset. The vertical axis represents the speedup ratios, while the horizontal axis shows the number of worker threads. In SGPM, the speedup ratio reaches up to approximately 3.7 times for each datasets. In CGPM, the speedup ratio reaches up to approximately 3 times using \( R15 \) and \( \text{Aggregation} \). However, the speedup ratio reaches up to approximately 2 times using \( \text{Pathbased} \). The previous model obtained a higher speedup compared to the proposed model.

In addition, Fig. 7 shows the processing time for each dataset. The processing time of CGPM are faster than that of SGPM using \( R15 \) and \( \text{Aggregation} \), as shown in Fig. 7. This is because the number of replication data is reduced by the complex grid partition. The processing time with \( t = 8 \) and 4 of CGPM is worse than that of SGPM using \( \text{Pathbased} \). It is assumed that deviation of the loads occurred by combining of dense grid.

The previous model obtained a higher speedup compared to the proposed model, as shown in Fig 6. 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 conducted an experiment with changing the condition of combining of the dense grids. The condition is combining up to the number of all geospatial data divided by four. That is, we set the same condition for each the number of worker thread. Fig 8 and Fig 9 show the speedup and processing time, respectively.
The speedup ratio of CGPM is much the same as the speedup ratio of SGPM using R15 and Aggregation. The processing time of CGPM is faster than the processing time of SGPM, as shown in Fig 9. However, the processing time of CGPM in Fig 9 is slower than the processing time of CGPM in Fig 7. We are necessary to develop a method for automatic setting of the combining condition for each the number of worker thread.

Datasets whose data distribution exhibits a small deviation on the coordinate space. We then created datasets whose data distributions exhibited large deviations. We artificially expanded the number of geospatial data of each dataset to about 100,000. A_R15, A_Aggregation and A_Pathbased are half of the geospatial data distributed in the lower left. The parameters were set as \( p = 8 \) and \( t = 4 \). The parameters were set as \( \epsilon = 0.2 \) and \( \text{MinGSD} = 3000 \) with A_R15, \( \epsilon = 1.0 \) and \( \text{MinGSD} = 4650 \) with A_Aggregation, and \( \epsilon = 1.4 \) and \( \text{MinGSD} = 6900 \) with A_Pathbased.

Table 1 shows the processing time for each process using A_R15, A_Aggregation, and A_Pathbased. The processing time of CGPM is faster than that of SGPM using A_R15, A_Aggregation and A_Pathbased, as shown in Table 1. These results indicate that CGPM is effective in terms of the processing time, using a data distribution with a large deviation on the coordinate space.

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 improve the speedup performance of parallel processing. The data replication technique is utilized to determine whether a geospatial data near the border of a grid is core geospatial data or not. 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 partition. In our future work, we intend to discuss combining condition of the dense grids. Moreover, we intend to conduct experiments by increasing the number of workers.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 16J05403 and 26330139, and Hiroshima City University Grant for Special Academic Research (General Studies).
Fig. 8: Speedup for each dataset with changing the condition of combining of the dense grids

Fig. 9: Processing time for each dataset with changing the condition of combining of the dense grids

Table 1: Processing time for each process using data distribution with a large deviation

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Model</th>
<th>Calculating neighbors(s)</th>
<th>Creating tasks(s)</th>
<th>Tasks process(s)</th>
<th>Merging clusters(s)</th>
<th>Total time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_R15</td>
<td>SGPM</td>
<td>26.05</td>
<td>0.08</td>
<td>21296.92</td>
<td>0.02</td>
<td>21323.07</td>
</tr>
<tr>
<td></td>
<td>CGPM</td>
<td>30.15</td>
<td>16.42</td>
<td>12767.14</td>
<td>0.06</td>
<td>12813.78</td>
</tr>
<tr>
<td>A_Aggregation</td>
<td>SGPM</td>
<td>38.48</td>
<td>0.09</td>
<td>27733.29</td>
<td>0.11</td>
<td>27771.98</td>
</tr>
<tr>
<td></td>
<td>CGPM</td>
<td>49.12</td>
<td>31.56</td>
<td>22728.25</td>
<td>0.14</td>
<td>22809.17</td>
</tr>
<tr>
<td>A_Pathbased</td>
<td>SGPM</td>
<td>71.69</td>
<td>0.12</td>
<td>31777.16</td>
<td>0.14</td>
<td>31849.11</td>
</tr>
<tr>
<td></td>
<td>CGPM</td>
<td>97.29</td>
<td>88.39</td>
<td>30876.74</td>
<td>0.20</td>
<td>31062.62</td>
</tr>
</tbody>
</table>

References


