セルベースのDBSCANのマルチコアCPU上における並列化

酒井　達弘[†,††]　田村　慶一[†]　北上　始[†††]　竹澤　寿幸[†]

† 広島市立大学大学院情報科学研究科　〒 731–3194 広島県広島市安佐南区大塚東 3–4–1
†† 日本学術振興会特別研究員 DC1　〒 102–0083 東京都千代田区麹町 5–3–1
††† 広島工業大学情報学部　〒 731–5193 広島県広島市佐伯区三宅 2–1–1
E-mail: da65003@e.hiroshima-cu.ac.jp, {ktamura,takezawa}@hiroshima-cu.ac.jp,
h.kitakami.su@it-hiroshima.ac.jp

あらまし　密度に基づくクラスタリングはデータの密度をクラスタリングの基準とした，汎用的に使用されているクラスタリング手法である．近年，ビッグデータへの注目の高まりとともにデータベースの大規模化が進んでいる．そこで，大規模なデータに対する高速なクラスタリング手法の開発が求められている．密度に基づくクラスタリングの代表的なアルゴリズムである DBSCAN の高速化手法の一つとして，セルベースの DBSCAN が提案されている．セルベースの DBSCAN はデータセット全体を小さいセルに分割し，データの密度をセル単位で考え，セルを結合することでクラスタリングを行う．本論文では，セルベースの DBSCNA のさらなる高速化を目的とし，セルベースの DBSCAN のマルチコア CPU 上における並列化手法を提案する．提案手法はセルベースの DBSCAN の各処理について，マスタ・ワーカモデルによる動的負荷分散を用いて並列化し，処理の高速化を行う．評価実験の結果，提案手法は並列化によってセルベースの DBSCAN を高速化できることを示した．

キーワード　密度に基づくクラスタリング，DBSCAN，並列処理，マルチコア CPU，グリッド分割

1. はじめに

近年，ビッグデータへの注目の高まりとともにデータベースの大規模化が進んでおり，データマイニング分野においてデータクラスタリングの高速化に関する研究が注目を集めている．Ester ら [1] [2] によって提案された密度に基づくクラスタリングは，データの密度をクラスタリングの基準とした汎用的に使用されているクラスタリング手法である．密度に基づくクラスタリングの代表的な手法として DBSCAN（Density-based spatial clustering of applications with noise）がある．DBSCAN では密度の基準として，距離 ϵ 以内の近傍に含まれるデータ数を用いる．そして，データの距離 ϵ 以内に $MinPts$ 個以上のデータが存在するとき当該データをコアデータと呼び，コアデータを接続していくことでクラスタを抽出することができる．

DBSCAN には，各データの近傍を求めるための範囲検索とクラスタを形成するための到達可能データ探索という計算コストの大きい二つの処理がある．そこで，DBSCAN の高速化について数多くの研究が行われてきた．高速化の研究では，クラスタリング結果が厳密解となる手法と近似解を許す手法に分かれて研究が行われており，本論文では厳密解の得られる手法を研究対象とする．現在，高速かつ厳密解の得られる DBSCAN の高速化手法として，セルベースの DBSCAN [3] [4] が提案されている．

セルベースの DBSCAN は範囲検索の回数を大幅に削減し，小さく分割されたセルを基準にクラスタを形成することで到達可能データ探索の処理を必要としないため，処理の高速化が実現できている．セルベースの DBSCAN は 2 次元の場合において，データセット全体を一辺が長さ $\epsilon/\sqrt{2}$ の小さいセルに分割する．ここで，セル内のデータ数が $MinPts$ 以上である場合，セル内の全てのデータは自動的にコアデータであると判定できる．そして，コアデータを含むセルを結合していく．二つのセル間に距離 ϵ 以内の任意のコアデータのペアがある場合にのみ，その二つのセルを結合する．我々は先行研究 [5] において，このセルの結合判定に最小外接矩形とセルの再帰分割を用いることで処理の高速化を行った．

本論文では，セルベースの DBSCNA のさらなる高速化を目的とし，セルベースの DBSCNA のマルチコア CPU 上における並列化手法を提案する．本論文の貢献は以下の通りである．

- セルベースの DBSCAN の各処理についてマスタ・ワーカモデルによる動的負荷分散を用いた並列化手法を提案する．
- 最も処理時間を要するセル結合の処理について，一回のセルの結合判定を単位タスクとする手法，均等グリッド分割を用いる手法と再帰的なグリッド分割を用いる手法の三手法を提案する．
- 評価実験により，提案手法が並列化によってセルベースの DBSCAN を高速化できることを示す．

本論文の構成は以下の通りである．第 2 章では，関連研究を述べる．第 3 章では，DBSCAN とセルベースの DBSCAN を説明する．第 4 章では，提案手法について説明する．第 5 章では，評価実験の結果を示し，第 6 章で本論文をまとめる．

2. 関連研究

近年，ビッグデータへの注目の高まりとともにデータベースの

大規模化が進んでおり，データクラスタリングの高速化に関する研究が注目を集めている．密度に基づくクラスタリング [1] [2] はデータの密度をクラスタリングの基準とした汎用的に使用されているクラスタリング手法であり，多くの研究者によってその代表的な手法である DBSCAN の高速化が行われてきた．DBSCAN の高速化の研究では，クラスタリング結果が厳密解となる手法 [6] [7] [8] と近似解を許す手法 [9] [10] [11] [12] [13] [14] に分かれている．近似解を許す手法はパラメータやデータ分布によっては得られるクラスタリング結果が厳密解とは異なる場合があり，精度の求められるアプリケーションでの利用には有効ではない．

厳密解を保ちつつ，高速化が可能な手法としてセルベースの DBSCAN [3] [4] が提案されている．セルベースの DBSCAN は範囲検索の回数を大幅に削減し，小さく分割されたセルを基準にクラスタを形成するため到達可能データ探索の処理を必要としない．我々は先行研究 [15] [5] において，セルベースの DBSCAN の最も処理時間を要するセルの結合判定に最小外接矩形とセルの再帰分割を用いることで処理の高速化を行った．

DBSCAN の並列化による高速化も行われている [16] [17] [18]．近年では，並列処理のプラットフォームとして様々な新しい計算機プラットフォームが提供されており，新しい計算機プラットフォーム上での並列化に関する研究が盛んに行われている．例えば，Graphics Processing Unit（GPU）を用いた手法 [19] [20] や，MapReduce を用いた手法 [21] [22] が提案されている．しかしながら，セルベースの DBSCAN の並列化は行われていない．本論文では，マイクロプロセッサのマルチコア化が進んでいるという点を踏まえ，マルチコア CPU 上における DBSCAN の並列化に着目する．厳密解の得られるセルベースの DBSCAN に焦点を当て，セルベースの DBSCAN の各処理についてマスタ・ワーカモデルによる動的負荷分散を用いて並列化することで，さらなる高速化を行う．

3. 事前準備

本章では，DBSCAN とセルベースの DBSCAN について説明する．

3.1 DBSCAN

本節では，DBSCAN の諸定義を説明する．d 次元のデータ集合を DT とし，ϵ と $MinPts$ をユーザが与えるパラメータとする．データ $dt_p \in DT$ について，dt_p から距離 ϵ 以内に存在するデータ集合を dt_p の ϵ-近傍と呼び，$DTN_\epsilon(dt_p)$ と表記する．本研究では，データ間の距離はユークリッド距離と定める．

定義 1 (コアデータ) データ dt_p の ϵ-近傍について，$|DTN_\epsilon(dt_p)| \geqq MinPts$ を満たすとき，dt_p をコアデータと呼ぶ．

定義 2 (ϵ-密度的に到達可能) データ列 $(dt_1, dt_2, \cdots, dt_n)$ について，以下の条件を満たすとき，dt_1 から dt_n へ ϵ-密度的に到達可能であると表現する．

（1） $dt_1, dt_2, \cdots, dt_{n-1}$ がコアデータである．

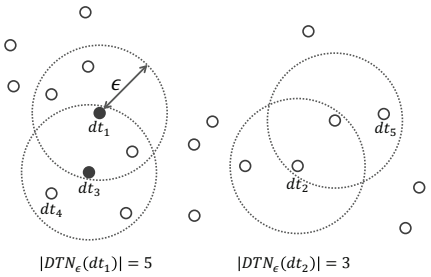


図 1　定義 1 と定義 2 の例

（2） dt_{i+1} が dt_i の ϵ-近傍に存在（$dt_{i+1} \in DTN_\epsilon(dt_i)$）する．

図 1 に定義 1 と定義 2 の例を示す．$MinPts = 5$ とする．データ dt_1 の ϵ-近傍は dt_1 を含めると $|DTN_\epsilon(dt_1)| = 5$ となるので，dt_1 はコアデータとなる．反対に，データ dt_2 の ϵ-近傍は $|DTN_\epsilon(dt_2)| = 3$ であるため，dt_2 はコアデータではない．また，dt_1 から dt_3 と dt_4 へは ϵ-密度的に到達可能であるが，dt_2 から dt_5 へは ϵ-密度的に到達可能ではない．

DBSCAN において，密度に基づくクラスタはコアデータから ϵ-密度的に到達可能なデータを再帰的に接続していくことで形成されていく．コアデータではないがコアデータから ϵ-密度的に到達可能なデータをボーダデータと呼ぶ．また，どのコアデータからも ϵ-密度的に到達可能ではないデータをノイズと呼ぶ．密度に基づくクラスタの定義を以下に示す．

定義 3 (密度に基づくクラスタ) データ集合 DT において，密度に基づくクラスタ DBC は以下の条件を満たす部分データ集合である．

（1） 任意のコアデータ $dt_p \in DBC$ について，DBC に属する全てのデータは dt_p から ϵ-密度的に到達可能である．

（2） 任意のデータ dt_p，$dt_q \in DBC$ について，dt_p と dt_q へ ϵ-密度的に到達可能な $dt_o \in DBC$ が存在する．

3.2 セルベースの DBSCAN

本節では，セルベースの DBSCAN について説明する．セルベースの DBSCAN は主にセル分割，近傍セルリストの作成，コアデータの判定，セル結合とボーダデータとノイズの判定の五つの処理から構成される．五つの処理が完了すると，各データはどのクラスタに属するか，またクラスタに属さないノイズとなるかが決定する．

3.2.1 セル分割

最初に，データセット全体を小さいセルに分割し，各データを一つのセルに割り当てる．セルは一辺の長さが同じ超立方体とする．d をデータの次元数とすると，セル一辺の長さは $\epsilon/\sqrt{d}$ となる．セル一辺の長さを $\epsilon/\sqrt{d}$ とすることでセルの対角線の長さは ϵ となる．図 2 にセルベースの DBSCAN の例を示す．図 2 は $d = 2$ であり，$\epsilon/\sqrt{2} \times \epsilon/\sqrt{2}$ のセルに分割されている．

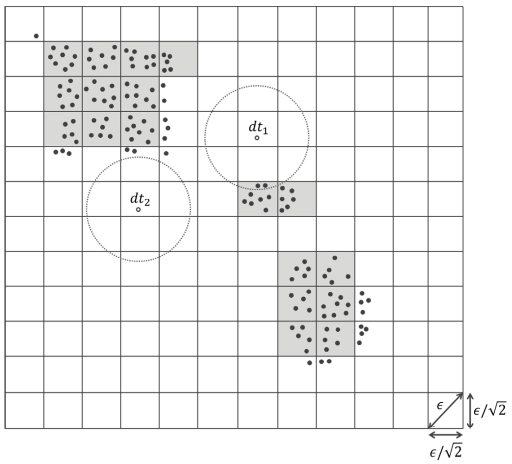


図 2　セルベースの DBSCAN の例

3.2.2　近傍セルリストの作成

次に，各セルの近傍セルリストを作成する．あるセルから最短距離が ϵ 以下のセルを，そのセルの近傍セルと呼び，各セルの近傍セルリストを作成する．近傍セルリストを作成することで，あるデータの ϵ-近傍を求める際に，そのデータが属するセルの近傍セルリストの内のデータとの距離を計算するのみで，求めることができる．

3.2.3　コアデータ判定

次に，各データがコアデータであるかそうでないか判定する．セル内のデータ間の距離は最大でも高々ϵ であり，セル内にデータが $MinPts$ 以上存在する場合，そのセル内のデータは全てコアデータと判定できる．反対に，セル内のデータ数が $MinPts$ 未満の場合，そのセル内のデータについては近傍セルに存在するデータとの距離計算を行い，コアデータであるか判定する．図 2 の例を見ると，$MinPts = 5$ とすると，グレーで示しているセルにはデータが 5 以上あり，これらのセルに属するデータは全てコアデータとなる．

3.2.4　セ ル 結 合

次に，コアデータを含むセルを結合しクラスタを形成していく．二つのセル間に距離 ϵ 以内の任意のコアデータのペアがある場合，二つのセルに含まれる全データは ϵ-密度的に到達可能となり，二つのセルを結合しセル内のデータ集合でクラスタを形成する．セルの結合判定には，先行研究 [5] にて提案された最小外接矩形とセルの再帰分割を用いる．図 2 の例を見ると，左上に密集しているセルに属するデータ集合と，右下に密集しているセルに属するデータ集合とで二つのクラスタが形成される．

3.2.5　ボーダデータとノイズの判定

最後に，コアデータではないデータに対してクラスタに属するボーダデータ，もしくはノイズとなるか判定する．判定するデータについて，もし ϵ 以内にコアデータが存在する場合，そのデータはボーダデータとなり，最も近いコアデータが属するクラスタに属する．反対に，ϵ 以内にコアデータが存在しない場合，そのデータはクラスタに属さないノイズとなる．図 2 の例を見ると，データ dt_1 から距離 ϵ 以内にコアデータが存在しており，dt_1 はボーダデータとなる．反対に，データ dt_2 から距離 ϵ 以内にはコアデータが存在していないので，dt_2 はノイズとなる．

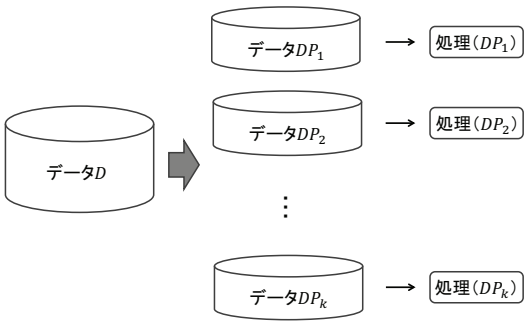


図 3　データ分割並列化の例

4. 提 案 手 法

本章では，提案手法であるセルベースの DBSCNA のマルチコア CPU 上における並列化手法を説明する．

4.1　負荷分散モデル

提案手法では，3.2 節にて説明したセルベースの DBSCAN の各処理について，マスタ・ワーカモデルによる動的負荷分散を用いて並列化を行う．マスタワーカモデルとは，並列処理や分散処理の実装で用いられるモデルの一つで，処理を分割して単位タスク（一つの作業単位）を生成するスレッドのマスタと，単位タスクを取得して処理を行うワーカスレッドの二種類のスレッドによって構成される．マスタスレッドはタスクプールを用いてタスクを管理する．各ワーカスレッドは一つのタスクを終了した後，タスクプールからタスクを取得し，取得したタスクを行う．最終的に，タスクプールが空になり，各ワーカが処理を終えると全体の処理が終了したこととなる．

4.2　各処理の並列化

本節では，セル分割，近傍セルリストの作成，コアデータ判定，ボーダデータとノイズの判定の並列化について説明する．これらの処理はデータ分割並列化手法を用いて並列化を行う．データ分割並列化とは，扱うデータを複数のパーティションに分割し，各パーティションの処理を並列に実行することで全体の処理を並列化する手法である．図 3 にデータ分割並列化の例を示す．マルチコア CPU 環境においては，各 CPU は各パーティションに分割されたデータ集合に対する処理を行う．

セル分割の並列化では，各データの所属セルを求める処理について，並列化を行う．データセットを均等に k 分割し，分割されたデータ集合に対する所属セルを求める処理を単位タスクとする．

近傍セルリストの作成の並列化では，各セルの近傍セルリストを作成する処理の並列化を行う．セル集合を均等に k 分割し，分割されたセル集合に対する近傍セルリストを求める処理を単位タスクとする．

コアデータ判定については，3.2.3 項に示すように，セル内

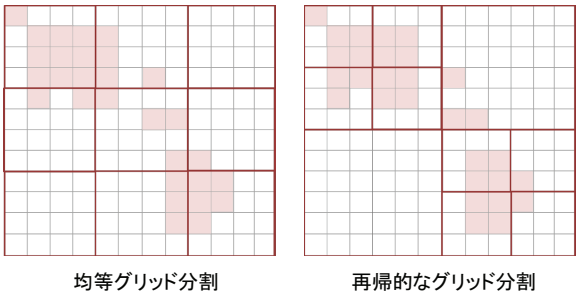


図 4　グリッド分割の例

にデータが $MinPts$ 以上存在する場合，そのセル内のデータは全てコアデータと判定できる．このコアデータ判定については高速に行えるため，並列化を行わない．セル内のデータ数が $MinPts$ 未満のセルに属するデータのリストを作成し，均等に k 分割を行う．分割されたデータ集合に対するコアデータ判定の処理を単位タスクとする．

ボーダデータとノイズの判定の並列化もコアデータ判定の並列化と同様に，判定の対象となるコアデータではないデータのリストを作成する．そして，データ集合を均等に k 分割し，分割されたデータ集合に対するボーダデータとノイズの判定の処理を単位タスクとする．

4.3　セル結合の並列化

セル結合の並列化では，セルの結合判定の処理を並列化する．本論文では，一回のセルの結合判定を単位タスクとする手法，均等グリッド分割を用いる手法と再帰的なグリッド分割を用いる手法の三手法を提案する．

一回のセルの結合判定を単位タスクとする手法では，タスク分割並列化手法によって並列化する．タスク分割並列化手法とは，行う処理を同時に実行できる複数のタスクに分割し，各タスクの処理を並列に実行することで全体の処理を並列化する手法である．本手法では，セルの結合判定を行うセルのペアのリストを作成し，各結合判定の処理を単位タスクとする．

均等グリッド分割を用いる手法では，データ空間全体を各次元について均等に l 分割する．そして，各セルをグリッドに割り当て，一つのグリッドのセル集合に対する結合判定の処理を単位タスクとする．図 4 の左に均等グリッド分割の例を示す．図 4 の均等グリッド分割では，各次元について 3 分割し，合計 9 つのグリッドを作成している．しかしながら，均等グリッド分割を用いた手法では，各グリッドの結合判定を行うセルの数に偏りがあり，負荷分散が行えず，処理に多くの時間を要してしまうグリッドが出てくる．図 4 において，赤で示しているセルをコアデータが含まれており結合判定を行うセルとすると，左上のグリッドが 10 個のセルについて結合判定を行う必要があり，他のグリッドに比べて多くの処理時間を要する．

均等グリッド分割よりも負荷分散を効率よく行うために，再帰的なグリッド分割を用いる手法を考える．再帰的なグリッド分割を用いる手法では，最初に各次元について，2 分割したグリッドを作成する．そして，グリッド内のコアデータが含まれるセルの数が，（コアデータが含まれるセルの総数 / k）よりも多い場合，さらに分割を行う．図 4 の右に再帰的なグリッド分割の例を示す．図 4 の再帰的なグリッド分割の例では，一つのグリッド内の結合判定を行うセル数は最大でも 5 個となっており，負荷分散を効率よく行うことができる．

表 1　データセットの詳細

データセット名	次元数	データ数
SSD3	3	60,000,000
SSD5	5	60,000,000
SSD7	7	60,000,000

4.4　処 理 手 順

提案手法の処理手順は以下の通りである．

（1）　データセット全体を一辺の長さが $\epsilon/\sqrt{d}$ の超立方体に分割し，セルリストを作成する．

（2）　各セルの近傍セルリストの作成を並列化によって行う．

（3）　各データのセルへの割り当てを並列化によって行う．

（4）　各データのコアデータ判定を並列化によって行う．

（5）　各セル間の結合判定を並列化によって行う．

（6）　結合されたセル内のデータ集合をクラスタとし，クラスタリストを作成する．

（7）　クラスタに属していないデータのボーダデータとノイズの判定を並列化によって行い，ノイズリストを作成する．

（8）　クラスタリストとノイズリストを出力する．

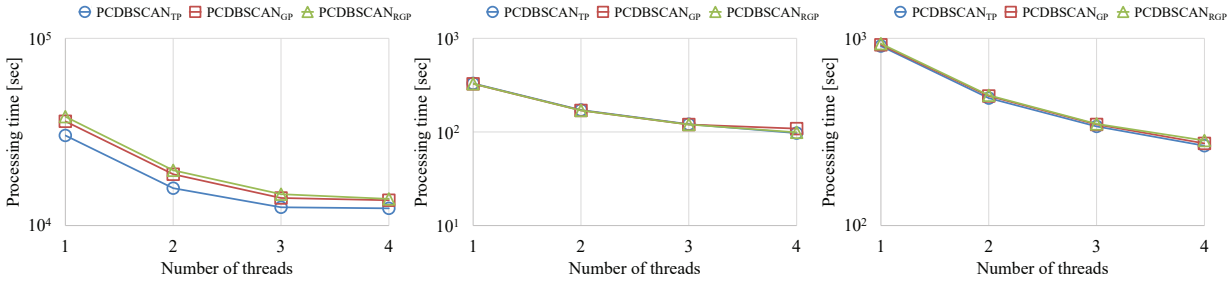
5.　評 価 実 験

本章では，評価実験の結果を示す．

5.1　実験内容とデータセット

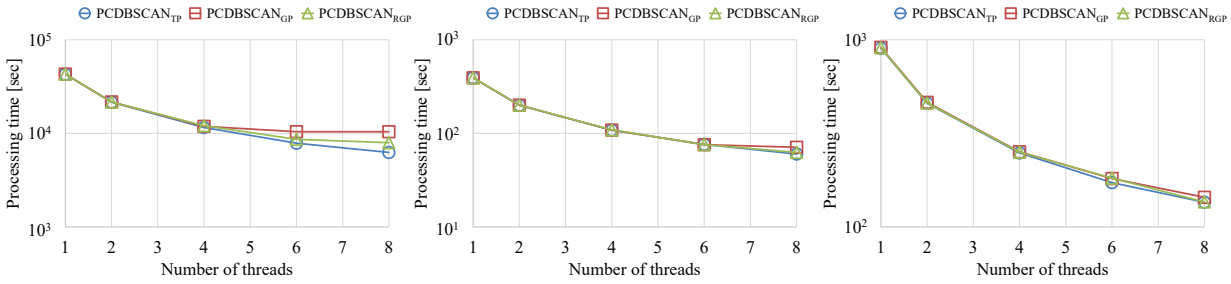
評価実験では，提案手法の処理時間とスピードアップの評価を行う．セル結合の並列化については，一回のセルの結合判定を単位タスクとする手法（$\text{PCDBSCAN}_{\text{TP}}$ と表記する），均等グリッド分割を用いる手法（$\text{PCDBSCAN}_{\text{GP}}$ と表記する）と再帰的なグリッド分割を用いる手法（$\text{PCDBSCAN}_{\text{RGP}}$ と表記する）を比較する．実験環境としては，4 コアの CPU を搭載した計算機（CPU : Intel Xeon E5-1270 V2 @ 3.50 GHz，メモリ : 32GB），8 コアの CPU を搭載した計算機（CPU : Intel Core i7-5960X @ 3.00GHz，メモリ : 64GB）と 48 コアの CPU を搭載した計算機（CPU : Intel Xeon E7-4850 V2 @ 2.3 GHz ×4，メモリ : 640GB）をそれぞれ用いて実験を行った．プログラミング言語は C++を用いて実装し，コンパイラとして g++ 4.9.4，オプションとして-O3 を使用した．

データセットは文献 [4] にて開発された Seed Spreader を使用して作成した 3，5 と 7 次元の人工データ（*SSD3*，*SSD5*，*SSD7*）を使用した．各データセットのデータ数を表 1 に示す．Seed Spreader は空間内に密集したデータ集合をいくつかランダムに配置し，少量のノイズをランダムに配置することで，クラスタリングのベンチマークに適した分布のデータセットを作成



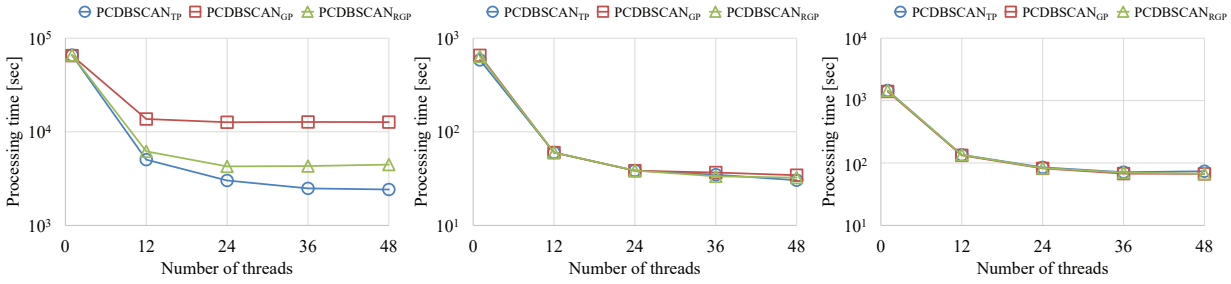
SSD3 *SSD5* *SSD7*

図 5　4 コアの計算機を使用した各データセットの処理時間



SSD3 *SSD5* *SSD7*

図 6　8 コアの計算機を使用した各データセットの処理時間



SSD3 *SSD5* *SSD7*

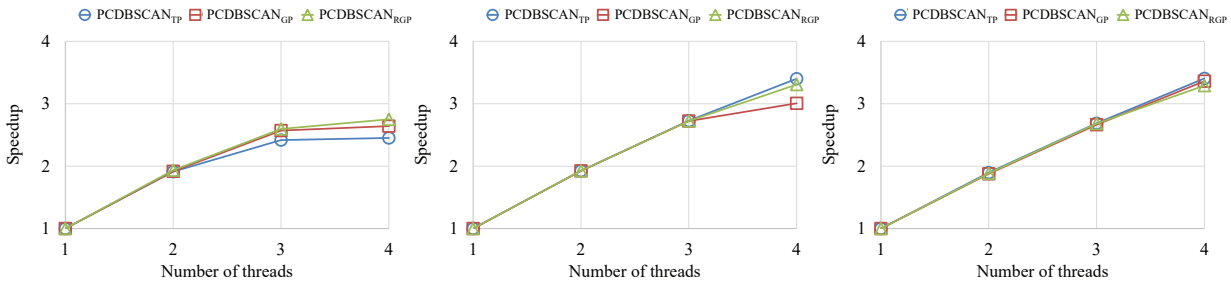
図 7　48 コアの計算機を使用した各データセットの処理時間

できる．パラメータは，$MinPts = 100$，$\epsilon = 5,000$，$k = 128$，$l = 8$ を用いた．ワーカスレッド数は，4 コアの計算機では 1 から 4 まで，8 コアの計算機では 1 から 8 まで，48 コアの PC では 1 から 48 まで用いて比較を行った．

5.2　実験結果

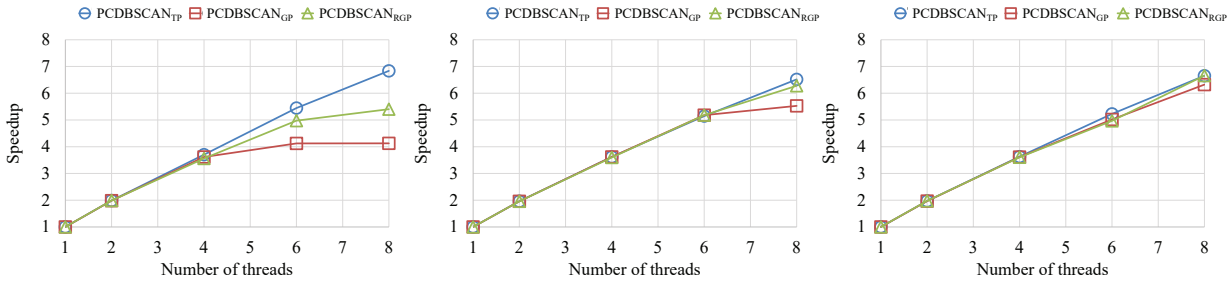
図 5，6 と 7 に 4 コア，8 コアと 48 コアの計算機を用いた各データセットにおける各手法の処理時間を示す．図 5，6 と 7 より，*SSD5* と *SSD7* の処理時間は，三手法に大きな差はなかった．三手法に大きな差がなかった理由としては，*SSD3* の処理時間に比べて高速であり，三手法とも十分に負荷分散できているためである．*SSD3* の処理時間では，各計算機において PCDBSCAN$_{TP}$ がどのスレッド数を用いた場合でも，最も高速となった．PCDBSCAN$_{TP}$ が最も高速であった理由としては，単位タスクが一回のセルの結合判定でありタスクの処理が非常に小さく，十分に負荷分散できているためである．48 コアの計算機において，*SSD3* の 48 スレッド使用時の処理時間は，PCDBSCAN$_{TP}$ は 2,418 秒，PCDBSCAN$_{GP}$ は 14,103 秒，PCDBSCAN$_{RGP}$は 4,680 秒 となった．この結果について，PCDBSCAN$_{GP}$ と PCDBSCAN$_{RGP}$ を比較すると，PCDBSCAN$_{RGP}$ の方が高速となった．図 7 より，PCDBSCAN$_{GP}$ は 12，24，36 と 48 スレッドを用いた結果に大きな差はない．これはセル結合において，セルの結合判定に 10,000 秒以上要しているグリッドがあり，そのグリッドのセルの結合判定に負荷が偏っていたため，全体の処理時間に影響を与えていた．

図 8，9 と 10 に 4 コア，8 コアと 48 コアの計算機を用いた各データセットにおける各手法のスピードアップを示す．最も高速であった PCDBSCAN$_{TP}$ について，4 コアの計算機では，*SSD3*，*SSD5* と *SSD7* について，4 スレッド使用時に 2.5 倍，3.4 倍と 3.4 倍のスピードアップがそれぞれ得られた．同様に 8 コアの計算機では，8 スレッド使用時に 6.8 倍，6.5 倍と 6.7 倍，48 コアの計算機では，48 スレッド使用時に 27 倍，19 倍



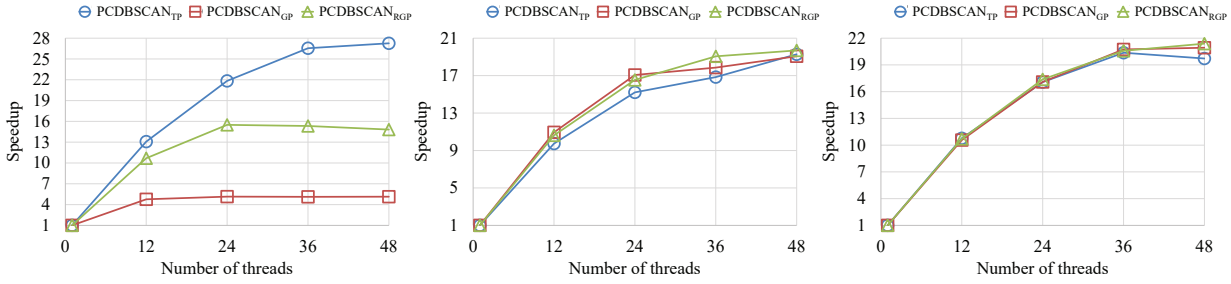
SSD3 *SSD5* *SSD7*

図 8　4 コアの計算機を使用した各データセットのスピードアップ



SSD3 *SSD5* *SSD7*

図 9　8 コアの計算機を使用した各データセットのスピードアップ



SSD3 *SSD5* *SSD7*

図 10　48 コアの計算機を使用した各データセットのスピードアップ

表 2　パラメータを変更して行った PCDBSCAN$_{TP}$ の実験結果

データセット	スレッド数	処理時間（秒）	スピードアップ
SSD3	1	66283.9	1
	12	5072.6	13.1
	24	2934.2	22.6
	36	2487.8	26.6
	48	2297.5	28.9
SSD5	1	4485.4	1
	12	377.1	11.9
	24	232.0	19.3
	36	186.5	24.1
	48	166.8	26.9
SSD7	1	13325.7	1
	12	1122.8	11.7
	24	658.1	20.2
	36	539.6	24.7
	48	460.9	28.9

と 19 倍のスピードアップが得られた．PCDBSCAN$_{TP}$ の 48 コアの計算機を用いた結果は，十分なスピードアップを得られなかった．

PCDBSCAN$_{TP}$ が十分なスピードアップを得られなかった理由としては，48 スレッドでの並列化に対するクラスタリングの処理数が非常に少なく，タスク生成などの逐次処理が処理時間の大部分を占めていたためである．そこで，PCDBSCAN$_{TP}$ の 48 コアの計算機を用いた実験について，処理数を増やすために $MinPts = 1,000$ に設定し，再実験を行った．表 2 にパラメータを変更して行った実験結果を示す．表 2 より，48 コアの計算機を用いた結果，PCDBSCAN$_{TP}$ は 26 から 28 倍のスピードアップが得られた．以上の結果より，提案手法は並列化によってセルベースの DBSCAN を高速化でき，特に，PCDBSCAN$_{TP}$ が最も高速となった．

6. ま と め

本論文では，セルベースの DBSCNA のさらなる高速化を目的とし，セルベースの DBSCNA のマルチコア CPU 上における並列化手法を提案した．提案手法はセルベースの DBSCAN

の各処理についてマスタ・ワーカモデルによる動的負荷分散を用いて並列化することで，処理の高速化を行う．また，セル結合の並列化については，一回のセルの結合判定を単位タスクとする手法，均等グリッド分割を用いる手法と再帰的なグリッド分割を用いる手法の三手法を提案した．評価実験の結果，提案手法は並列化によってセルベースの DBSCAN を高速化できることを示した．

今後の課題としては，実データセットを用いて評価を行うことや計算機クラスタ上での実装と評価を行うことがあげられる．5.2 節の実験結果において，一回のセルの結合判定を単位タスクとする手法が最も高速であったが，タスクの受け渡しに通信を行う必要のある計算機クラスタ上での実行を考えると，グリッド分割を用いる手法に比べてタスク数が非常に多いため，タスクの受け渡しに多くの処理時間を要する可能性がある．そのため，計算機クラスタ上で実装し，タスクの受け渡しに伴う通信を含めた処理時間を計測し，評価を行う必要がある．

謝　　辞

本研究の一部は，JSPS 科研費 JP16J05403，総務省 SCOPE（受付番号:162308002）と広島市立大学・特定研究費の支援により行われた．

文　　献

[1] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD 1996*, pp. 226–231, 1996.

[2] Jörgand Sander, Martinand Ester, Hans-Peterand Kriegel, and Xiaowei Xu. 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] Ade Gunawan. A faster algorithm for DBSCAN. Master's thesis, Technische University Eindhoven, 40 pages, 2013.

[4] Junhao Gan and Yufei Tao. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *Proc. SIGMOD 2015*, pp. 519–530, 2015.

[5] 酒井達弘, 田村慶一, 北上始, 竹澤寿幸. 最小外接矩形とセルの再帰分割を用いたセルベースの DBSCAN の高速化. 電子情報通信学会論文誌, Vol. J101-D, No. 4, 2018 年 4 月出版予定.

[6] Y. El-Sonbaty, M. A. Ismail, and M. Farouk. An efficient density based clustering algorithm for large databases. In *Proc. ICTAI 2004*, pp. 673–677, 2004.

[7] S. Mahran and K. Mahar. Using grid for accelerating density-based clustering. In *Proc. CIT 2008*, pp. 35–40, 2008.

[8] Son T. Mai, Ira Assent, and Martin Storgaard. AnyDBC: An efficient anytime density-based clustering algorithm for very large complex datasets. In *Proc. KDD 2016*, pp. 1025–1034, 2016.

[9] Shuigeng Zhou, Aoying Zhou, Jing Cao, Jin Wen, Ye Fan, and Yunfa Hu. Combining sampling technique with DBSCAN algorithm for clustering large spatial databases. In *Proc. PAKDD 2000*, pp. 169–172, 2000.

[10] M. Dash, H. Liu, and Xiaowei Xu. '1+1>2': merging distance and density based clustering. In *Proc. DASFAA 2001*, pp. 32–39, 2001.

[11] Xin Wang and Howard J. Hamilton. DBRS: A density-based spatial clustering method with random sampling. In *Proc. PAKDD 2003*, pp. 563–575, 2003.

[12] B. Borah and D. K. Bhattacharyya. An improved sampling-based DBSCAN for large spatial databases. In *Proc. ICISIP 2004*, pp. 92–96, 2004.

[13] B. Liu. A fast density-based clustering algorithm for large databases. In *Proc. ICMLC 2006*, pp. 996–1000, 2006.

[14] Cheng-Fa Tsai and Chien-Tsung Wu. GF-DBSCAN: A new efficient and effective data clustering technique for large databases. In *Proc. MUSP 2009*, pp. 231–236, 2009.

[15] Tatsuhiro Sakai, Keiichi Tamura, and Hajime Kitakami. Cell-based DBSCAN algorithm using minimum bounding rectangle criteria. In *Proc. BDMS 2017*, pp. 133–144, 2017.

[16] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. *Data Mining and Knowledge Discovery*, Vol. 3, No. 3, pp. 263–290, 1999.

[17] Domenicaand Arlia and Massimo Coppola. Experiments in parallel clustering with DBSCAN. In *Proc.of Euro-Par 2001*, pp. 326–331, 2001.

[18] Tatsuhiro Sakai, Keiichi Tamura, Kohei Misaki, and Hajime Kitakami. Parallel processing for density-based spatial clustering algorithm using complex grid partitioning and its performance evaluation. In *Proc. PDPTA 2016*, pp. 337–343, 2016.

[19] Christian Böhm, Robert Noll, Claudia Plant, and Bianca Wackersreuther. Density-based clustering using graphics processors. In *Proc. CIKM 2009*, pp. 661–670, 2009.

[20] Benjamin Welton, Evan Samanas, and Barton P. Miller. Mr. Scan: Extreme scale density-based clustering using a tree-based network of GPGPU nodes. In *Proc. SC 2013*, pp. 84:1–84:11, 2013.

[21] Wei Zhong Zhao, Hui Fang Ma, and Yan Xiang Fu. Research on parallel DBSCAN algorithm design based on MapReduce. Vol. 301, pp. 1133–1138, 2011.

[22] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. MR-DBSCAN: An efficient parallel density-based clustering algorithm using MapReduce. In *Proc. ICPADS 2011*, pp. 473–480, 2011.