マルチコア CPU 上における距離に基づく外れ値検出の並列処理
Parallel Processing of Distance-based Outlier Detection on Multi-core CPU

奥　淳基[†]　　田村　慶一[‡]　　北上　始[‡]
Junki Oku　　Keiichi Tamura　　Hajime Kitakami
広島市立大学大学院情報科学研究科
Email: [†]mx67003@wm.hiroshima-cu.ac.jp, [‡]{ktamura,kitakami}@hiroshima-cu.ac.jp

Abstract—**In the last few decades, outlier detection has attracted much attention of researchers, because it is widely used for many different application domains. Distance-based outlier detection, which is a non-parametric approach, identifiers unusual data objects, where their distance to neighbors is used as a measure of unusualness. Distance-based outlier detection is known for its huge computation time. One of the most successful algorithms for the improvement of the algorithm is Orca, which is based on nested loop with randomization and a simple pruning rule. This paper proposes the parallel processing of Orca on a multi-core CPU. The proposed parallelization model utilizes the data partition parallelism and the multi-thread model. In the processing of Orca, we need to share the outlier score table among threads. To reduce conflicts on the outlier score table, the proposed parallelization model makes the cache of the cutoff value for each thread. The experimental results show the proposed parallelization model outperforms the conventional model.**

I.　はじめに

データ集合において，他のデータとは異なる傾向のデータを検出することを外れ値検出という[1]．外れ値検出は，例えば，株価低下による不景気予測，工場における生産ラインでの不良品検出，Web サイト上での不正アクセス検出，バイオインフォマティックスおける変動遺伝子検出，医療分野おける悪性細胞の検出など様々な分野で応用されている．

データ集合における外れ値は古くから主に統計学的な手法を用いて検出が行われてきた．しかしながら，統計学的な手法を実世界のデータベースに適用しようとすると，実世界のデータは多変量であるため，パラメータ分布を求めることが困難である．また，データベース中のデータは時間ともに変化するため，モデルが古くなり，適切に外れ値が検出できないという問題点がある．

そこで，ノンパラメトリックな手法として，近傍にあるデータとの距離を外れ値の判定に使うアプローチが提案されている[2]．この手法を距離に基づく外れ値検出という．距離に基づく外れ値検出では，一般に 2 つのパラメータ t，k を与え，t 個の外れ値を外れ値スコアが大きい順に列挙する．外れ値スコアとして，データから k 番目に近いデータとの距離を用いる方法[3]が頻繁に用いられており，本研究においても，k 番目に近いデータとの距離を外れ値スコアとして用いる．

距離に基づく外れ値検出の最適な逐次アルゴリズムとして Orca アルゴリズム[4]が提案されている．Orca アルゴリズムでは，途中段階の暫定 t 番目の外れ値スコアをカットオフ値として枝刈りに使用する．データベース中の i 番目のデータについて外れ値スコアを求める場合，他の全てのデータとの距離を計算することになるが，k 近傍を記録しておき，暫定的な外れ値スコアがカットオフ値以下となれば，これ以上探索をしても i 番目のデータは t 番目に入ることはないので，次のデータへ移ることができる．

本研究では，Orca アルゴリズムに焦点を当て，Orca アルゴリズムのマルチコア CPU 上での並列化手法を提案する． 近年，CPU のマルチコア化が急速に進んでおり[5]，マルチコア CPU の資源を有効に活用するためにマルチコア CPU 上で効率的な並列化手法を考えることは，重要な課題のひとつとなっている．提案する並列化手法の特徴は次の通りである．

- データ分割並列化手法を用いて，並列化を行う．マルチコア CPU においてマルチスレッドで並列化を行い，各スレッドに対して別々のデータを割当て，処理の高速化を図る．データ分割方法として，ブロック分割とラウンドロビン分割を適用する．
- Orca アルゴリズムでは，外れ値スコアランキングテーブルを作成し，途中段階の t 番目の外れ値スコアをカットオフ値として，参照できるようにしている．しかしながら，複数のスレッドでこのテーブルを同時に参照・更新するとスレッド間の競合が大きい．そこで，グローバルに外れ値スコアランキングテーブルを作成するだけでなく，各スレッドでローカルの外れ値スコアランキングテーブルを作成し，競合の発生を小さくする．
- 正確な枝刈りには，グローバルの外れ値スコアランキングテーブルの t 番目の外れ値スコアをカットオフ値として参照する必要がある．この参照は頻繁に発生するが，この値を直接参照すると，テーブルの更新と参照が競合する．この問題を解決するため，各スレッドにカットオフ値のキャッシュを持たせる．

本論文の構成は次の通りである．第 2 章では，距離に基づく外れ値，その検出手法とその関連研究を述べる．第 3 章では，本研究で研究対象としている Orca アルゴリズムについて説明する．第 4 章では並列化手法について述べる．第 5 章で評価実験の結果を示し，第 6 章で本研究のまとめを行う．

II. 距離に基づく外れ値検出

本章では，距離に基づく外れ値検出と，その検出手法についての関連研究を示す．

A. *距離に基づく外れ値*

距離に基づく外れ値検出では，近傍との距離が大きいデータは，他のデータとは異なる特徴を持つデータといえるため，そのデータを外れ値として検出する．例えば，図 1 に示すデータ群があり，t=1，k=3 とする．外れ値スコアは k 近傍のデータとの距離を用いる方法で算出するので，図 1 の黒丸を見たとき，K との距離が外れ値スコア d となる．同様にすべてのデータに対してこの動作を行ったとき，このデータ群では黒丸が外れ値であるといえる．

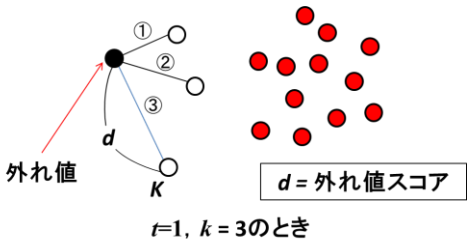


図 1　距離に基づく外れ値の例

B. *検出アルゴリズム*

距離に基づく外れ値検出の最もシンプルな検出アルゴリズムでは，データベース中の各データについて，他のすべてのデータ間との距離を計算することで各データの外れ値スコアを算出し，外れ値スコアの上位 t 番目を求める．これは，入れ子型ループを用いて簡単に実装することができる．

しかしながら，入れ子型ループではデータ数を n=|D|とすると，計算量が $O(n^2)$必要となり，大規模なデータベースでは計算量が爆発的に増えて，実用的な時間で外れ値を検出できないという問題がある．また，次元数 d が増えると距離計算の重みが増え，計算量は $O(n^2d)$となる．

そこで，高速化に関する様々な研究が行われている．データの k 近傍を高速に求めるために，空間インデックスである KD-tree や R-tree などを利用してデータの k 近傍を求める手法[6]が示されている．空間インデックスを利用した場合，計算量は $O(nlog\ n)$となるが，予め空間インデックスを作成しておく必要があり，大規模データベースだと空間インデックスの作成に時間がかかることや，次元の呪いの問題で低次元のデータでしか有効性がないことが問題となっている．

また，データをグリッドに分割し，データに近いグリッドに存在するデータから順に距離を求める方法や，グリッドを枝刈りする手法が提案されているが，グリッド分割に時間がかかることが課題となっている．一方，Orca アルゴリズムは，入れ子型アルゴリズムに対して，シンプルであるが効果的な枝刈りを導入したアルゴリズムであり，大規模かつ，高次元のデータにおいても，高速であることが示されている．

III. ORCA アルゴリズム

Orca アルゴリズムは途中段階の t 番目の外れ値スコアをカットオフ値とし，枝刈りに使用する．データセットの特徴により枝刈りの効果は変わってくるが，早い段階で外れ値が発見され，枝刈りが効果的に働いた場合は，シンプルなアルゴリズムにも関わらず，検出処理の高速化が期待できる．

例えば，1 番目から i-1 番目（t<i-1）のデータについて外れ値スコアが算出されており，t 番目に大きい外れ値スコアをカットオフ値（以下，この値を *cutoff* とする）とする．ここで，データ x_i について，他のデータと距離計算を行う途中段階において，k 番目に近いデータとの距離を *dist* とする．このとき，*dist* < *cutoff* であれば，データ x_i はこの後，他のどのデータとも距離計算を行ったとしても外れ値とはならない．よって，枝刈りができる．

Orca アルゴリズムは次の通りである．

(1) t 個(ユーザー指定)の要素(外れ値スコアとデータ番号のペア)からなる外れ値スコアランキングテーブルを作成する．

(2) i=1 から i=N まで以下の処理を繰り返す．

(a) k 個(ユーザー指定)の要素(各データ間の距離とデータ番号のペア)からなる k 近傍ランキングテーブルを作成する．

(b) j=1 から j=n まで以下の処理を繰り返す．

(i) x_i と x_j の距離 d_{ij} を計算する．

(ii) d_{ij} を用いて k 近傍ランキングテーブルを更新する．

(iii) k 近傍ランキングテーブルに k 個の要素が，また，外れ値スコアランキングテーブルに t 個の要素が登録されている場合，k 近傍ランキングテーブルの k 番目の値(*outscore*)と外れ値スコアランキングテーブルの t 番目の値（*cutoff* 値）とを比較し，*outscore* < *cutoff* ならば，次のデータに移る．

(c) k 近傍ランキングテーブルの k 番目の要素の距離が x_i の外れ値スコアなので，外れ値スコアランキングテーブルを更新する．

(3) 外れ値スコアランキングテーブルを出力する．

図 2 に Orca アルゴリズムの枝刈りの例を示す．3 個の要素が入る外れ値スコアランキングテーブルに図 2 のようにデータ番号と外れ値スコアが入っているとする．このときのカットオフ値は 30 となる．次のデータ x_i の外れ値スコアが 20 であったとき， 20<30 が成立するので，データ x_i はこの後，他のどのデータとも距離計算を行ったとしても外れ値とはならないため，枝刈りされる．

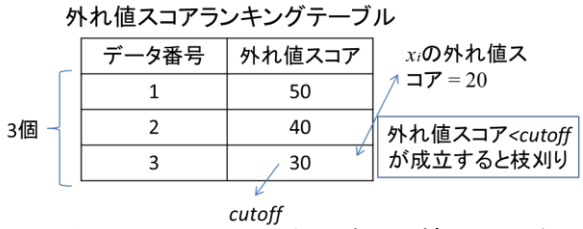


図 2　Orca アルゴリズムの枝刈りの例

IV.　提案手法

本章では提案手法であるデータ分割並列化手法に基づく Oraca アルゴリズムの並列化手法について説明する．

A.　データ分割並列化手法

データ分割並列化とは，扱うデータを分割して各スレッドに振り分けることで処理を並列化する手法である．マルチスレッドモデルを用いて，各コアにスレッドを割り当て，各スレッドが別々のデータを担当することで並列化を行う．データの分割方法として，ブロック分割，もしくは，ラウンドロビン分割で分割し，分割ごとに外れ値スコアを並列に算出していく．ブロック分割は範囲でデータを分割する方法で，ラウンドロビン分割は帯状に交互にデータを配分する方法である．本研究ではデータを 1 つずつ配分する．

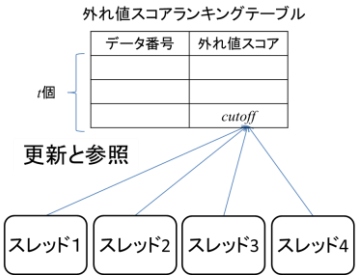


図 3　外れ値スコアランキングテーブル共有

B.　外れ値スコアキャッシュ方式

データ分割並列化手法を適用するにあたり，問題となるのが，外れ値スコアランキングテーブルの扱いである．Orca アルゴリズムで重要となるのが外れ値スコアランキングテーブルの t 番目の値であるカットオフ値である．カットオフ値により，枝刈りされかどうかが処理時間に大きく影響する．

枝刈りの正確性を守るには，外れ値スコアランキングテーブルを複数のスレッドで共有するのが一番であるが（図 3），このテーブルは頻繁に参照・更新されるためスレッド間で競合が大きくなる．そこで，各スレッドにもローカルの外れ値スコアランキングテーブルを作成し，各スレッドはローカルの外れ値スコアテーブルを参照・更新をする．ローカルの外れ値テーブルが更新されたときのみグローバルの外れ値スコアランキングテーブルを更新する（図 4）．

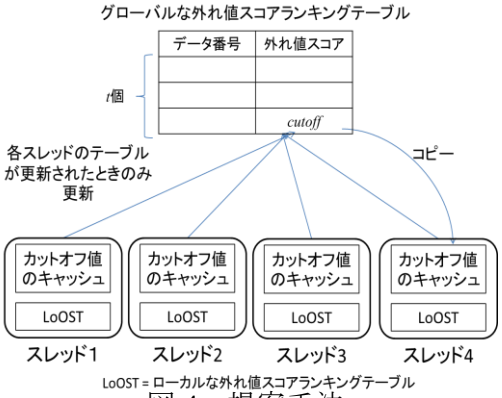


図 4　提案手法

また，グローバルの外れ値スコアランキングテーブルの t 番目の外れ値スコアをカットオフ値（*cutoff*）として参照する必要がある．この参照は頻繁に発生するが，この値を直接参照すると，テーブルの更新と参照が競合する．この問題を解決するため，各スレッドにカットオフ値のキャッシュを持たせる．

C.　アルゴリズム

提案手法における処理手順を次に示す．

(1) t 個(ユーザー指定)の要素(外れ値スコアとデータ番号のペア)からなる外れ値スコアランキングテーブルを作成する．

(2) m 個のスレッドを起動する．

(3) もし，ブロック分割なら各スレッドは i= N/スレッド数×スレッド *ID* から i= N/スレッド数×(スレッド *ID*+1)まで，以下の処理を繰り返す．もし，ラウンドロビン分割なら i=スレッド ID から i=N まで，i=i+スレッド数として，以下の処理を繰り返す．

(a) k 個(ユーザー指定)の要素(各データ間の距離とデータ番号のペア)からなる k 近傍ランキングテーブルを作成する．

(b) グローバルな外れ値スコアランキングテーブルの t 番目の値（*cutoff* 値）をローカル変数 *local_cutoff* にコピーする．

(c) j=1 から j=n まで以下の処理を繰り返す．

(i) x_i と x_j の距離 d_{ij} を計算する．

(ii) d_{ij} を用いて k 近傍ランキングテーブルを更新する．

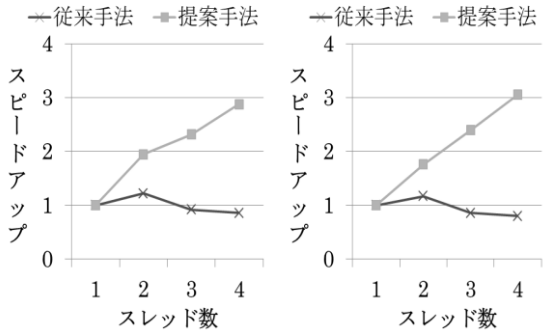
(iii) k 近傍ランキングテーブルに k 個の要素が，また，外れ値ランキングテーブルに t 個の要素が登録されている場合，k 近傍ランキングテーブルの k 番目の値 (*outscore*) と *local_cutoff* を比較し，*outscor*<*local_cutoff* ならば，次のデータに移る．

(d) k 近傍ランキングテーブルの k 番目の要素の距離が x_i の外れ値スコアなのでローカルな外れ値スコアランキングテーブルを更新する．ここで，ローカルな外れ値スコアランキングテーブルのランキングが更新されたときのみ当該データを使って，グローバル外れ値スコアランキングテーブルを更新する．

(5) すべてのスレッドの終了を待つ.
(6) グローバルな外れ値スコアランキングテーブルを出力する.

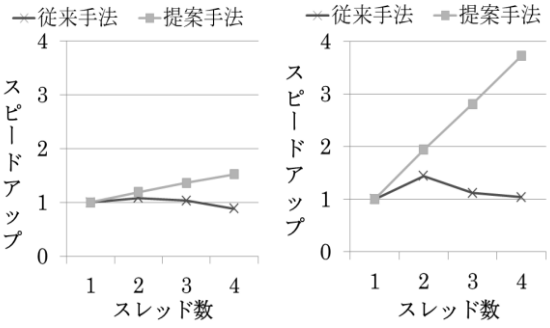
V. 評価実験

提案する並列化手法を，マルチスレッドを用いて実装し，2 種類のデータベースを使用してスピードアップ値を測定した．実験は，CPU:Intel（R）Core (TM)2 Quad CPU Q6700 @ 2.66GHz （コア数 4），Memory:2GB，HDD:250GB を搭載した PC 上で行った．それぞれの手法でのスピードアップの結果を図 5 と図 6 とに示す.



（a）ブロック分割　（b）ラウンドロビン分割

図 5　データセット ColorHistgram の実験結果



（a）ブロック分割　（b）ラウンドロビン分割

図 6　データセット Covertype の実験結果

データセット ColorHistgram（次元数 32，データ数 68040）の実験結果では，提案手法で約 3 倍のスピードアップが得られた．データ分割ではスピードアップに差は出ていない．それに対し，データセット Covertype（次元数 54，データ数 581012）の実験結果では，ラウンドロビン分割で提案手法を用いたものが約 4 倍のスピードアップが得られた.

従来手法でスピードアップが得られなかった原因として，外れ値スコアランキングテーブルを各スレッドが共有しているため，各スレッドが同時に外れ値スコアランキングテーブルを更新する時，競合が起きることがあげられる．また，データ分割の方法では，ブロック分割手法，ラウンドロビン分割手法の両方にメリットとデメリットがあるが，もし，ブロック分割でスレッド 1 に割り振ったデータの処理に時間がかかった場合，他のスレッドの処理が早く終了していても，実行時間はスレッド 1 の処理が終了するまでかかる．よってデータ分割は，扱うデータセットのデータの並びに依存するが，ラウンドロビン分割手法の方がよいと考えられる.

VI. まとめ

本論文では，距離に基づく外れ値検出に焦点を当て，Orca アルゴリズムのマルチコア CPU 上における並列化手法を提案した．提案手法は，ローカルに外れ値ランキングテーブルを作成すること，カットオフ値をキャシュすることでスレッド間の競合を防ぐ手法となっている．今後の課題として，CPU 上でキャッシュミスが起こっているか調べること，キャッシュの更新頻度の最適な調整方法の検討があげられる.

謝辞

本研究の一部は，広島市立大学・特定研究費（一般研究，研究課題名「時空間文書ストリーム上におけるバースト領域の抽出手法」） の支援により行われた.

参考文献

[1] Vic Barnett and Toby Lewis, Outliers in Statistical Data, John Wiley & Sons., 3rd edition, 1994.

[2] Edwin M. Knorr and Raymond T . Ng, and Vladimir Tucakov. ‘‘Distance-based outliers: algorithms and applications,’’ The VLDB Journal, Vol. 8,No. 3-4, pp.237-253, 2000.

[3] Fabrizio Angiulli and Clara Pizzuti. ‘‘Fast outlier detection in high dimensional spaces,’’ In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD ’02, pp15 26, 2002.

[4] Stephen D. Bay and Mark Schwabacher. ‘‘Mining Distance Based Outliers in Near Linear Time with randomization and a simple pruning rule,’’ In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD ’03, pp. 29-38, 2003.

[5] Mark D. Hill , Michael R. Marty, ‘‘Amdahl's Law in the Multicore Era,’’ Computer, Vol.41,No.7, pp.33-38, July 2008.

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. ‘‘Lof: identifying density-based local outliers,’’ In Proceedings of the 2000 AMC SIGMOD international conference on Management of data, SIGMOD ’00, pp. 93-104, 2000.

問い合わせ先
〒731-3194
広島市安佐南区大塚東 3 丁目 4 番 1 号
広島市立大学大学院情報科学研究科
知能工学専攻
奥　淳基