Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

空間分割による並列 k 近傍問合せの提案

横山拓也[†1]　石川佳治[†2,†1,†3]

指定された点に対して最も近い k 個の点を求める k 最近傍問合せは，空間データベースでは基本的な問合せの 1 つである．これを一般化して，データベース中の各点について，それぞれの k 最近傍を一度に求める問合せを全 k 最近傍問合せという．本研究では，この全 k 最近傍問合せを MapReduce フレームワーク上で行う手法を提案する．空間をセルに分割し，全 k 最近傍問合せの処理を MapReduce の並列分散処理方式に合った形で実行する．対象データの不均一な分布も考慮した改善策についても提案を行う．

Processing Parallel All k Nearest Neighbor Queries Based on Spatial Partitioning

Takuya Yokoyama[†1] and Yoshiharu Ishikawa[†2,†1,†3]

In spatial databases, a k nearest neighbor query, which finds the k nearest points in a database for the given query point, is one of the fundamental queries. Its generalized form, called an all k nearest neighbor query, tries to find k nearest neighbors for each point in the database at once. In this paper, we propose methods for processing all k nearest neighbor queries over the MapReduce framework. We partition the target space into cells, and then perform query processing based on a parallel and distributed execution manner. We also propose an improved approach which considers non-uniform spatial point distributions.

†1 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†2 名古屋大学情報基盤センター
Information Technology Center, Nagoya University
†3 国立情報学研究所
National Institute of Informatics

1. はじめに

Web から得られるデータとして，ジオタグなどの位置情報（座標データ）を持つデータが増加してきており，それらの情報を利用した新たなサービスが出現している．ここで例えば，位置情報が利用できるソーシャルネットワークサービスにおいて，各ユーザの地理的近くにいた 5 人の友人を定期的に求め，彼らの情報をユーザに提示するようなサービスを考える．これは **k 最近傍問合せ**（k nearest neighbor query）の機能により実現できるが，ソーシャルネットワークのデータが大規模であることと，すべてのユーザに対して k 最近傍を求める必要があることから，バッチ処理的に一度に計算することが望ましい．そのような問合せは，しばしば**全 k 最近傍問合せ**（all k nearest neighbor query; AkNN query）と呼ばれ，効率的な問合せ処理方式が提案されている[4),6),12)]．2 つの空間情報源を k 最近傍の条件で結合する **k 最近傍結合**（k nearest neighbor join）[3)] は，このバリエーションであるといえる．

一方，近年の爆発的な情報量の増加に対応するために，高度な並列分散処理に基づくクラウドコンピューティング技術の開発が急速に進んでいる．先に述べたソーシャルネットワークの例では，大規模なデータがデータセンタにおいて分散管理されていると考えられ，クラウド環境に親和性の高いアプローチで，並列分散処理による AkNN 問合せの効率的な問合せを行うことが求められる．並列分散コンピューティング技術としては，Google が自社の大量データを扱うために設計した手法である MapReduce が有名であり[8)]，また，その考え方を基に開発された Apache Hadoop[9),11)] がある．本研究では MapReduce フレームワークに基づき，具体的には Hadoop の使用を想定した AkNN 問合せ処理手法の提案を行う．空間を分割し，AkNN の問合せ処理を並列分散処理により繰り返し実行することがポイントとなる．基本的な手法に加え，空間上の点データの偏りを考慮した改善手法についても提案を行う．

2. 関連研究

2.1 MapReduce

本研究では Apache により開発されている並列分散コンピューティングフレームワーク **Hadoop**[9),11)] の使用を想定している．そこで基礎となっている **MapReduce**[5)] について簡単に説明する．MapReduce では，Map と Reduce という 2 つの関数が基本的な役割を果たす．利用者はそれらの関数を実装し組み合わせてデータ処理を行うことになる．分散コ

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

ンピューティングの際に重要となるデータの複製などは Hadoop のシステムが自動的に行ってくれるため，利用者はデータフローのみに集中できる．Map 関数ではデータの抽出を目的としており，入力されたデータに必要に応じて何らかの加工をした上で，キーと値のペアの形でデータを生成する．Reduce 関数はデータの集約が目的であり，入力となったキーと値のペアを集計し，その結果を出力する．Map と Reduce の間には Shuffle という処理があり，ここで，Map 関数の出力であるキーと値のペアが，キーごとにまとめられて Reduce 関数が実行される計算ノードに集まる．Map と Reduce の処理においては他の計算ノードとの通信が必要ないため，並列にデータ処理をすることができる．

AkNN 問合せは一種の結合処理であると考えられる．MapReduce における結合処理は，複雑かつ大規模な分析処理のために重要であり，その効率化のための研究が進められている[2),7)]．しかし MapReduce の結合処理は，一般に等結合しか利用できないという制限があり，等号以外の条件での結合の実行は直接的には行えない．しかし，結合演算はデータ処理において非常に重要であるため，MapReduce 環境で等結合以外の結合をするための研究も進められている[10)]．

2.2 全 k 最近傍問合せ

k 最近傍問合せとは，あるデータに対し距離的に最も近いデータ k 個を取得する問合せである．一方，**全 k 最近傍問合せ**（all k nearest neighbor query; AkNN query）とは，データベース中の各点に対する k 最近傍を 1 回の処理ですべて求める問合せである．1 章で提示したソーシャルネットワークの例では，各ユーザの近くにいたユーザの情報を求める例を挙げた．AkNN 問合せはこれ以外にも，データマイニングに先立つ前処理を一括して実現する場合などにも用いられる[3)]．

AkNN 問合せについては，すでに多くのアプローチが提案されているが[4),6),12)]，これらは集中化された従来のコンピューティング環境を前提としたものであり，MapReduce における使用には適用できない．そこで本研究では，空間分割を基礎とした，MapReduce フレームワークに適した AkNN 問合せの処理手法を提案する．

3. 空間分割とその性質

3.1 空間のセルへの分割

全 k 最近傍を求めるには，点同士の距離計算が必要となる．単純に考えるなら，データ集合に含まれるすべての点のペアについて距離計算をし，各点についての距離が小さい上位 k 件の点を取り出せば，それが AkNN 問合せの答えとなる．しかし，すべての組合せについて距離を求めるこの方法では，計算コストが膨大になってしまう．

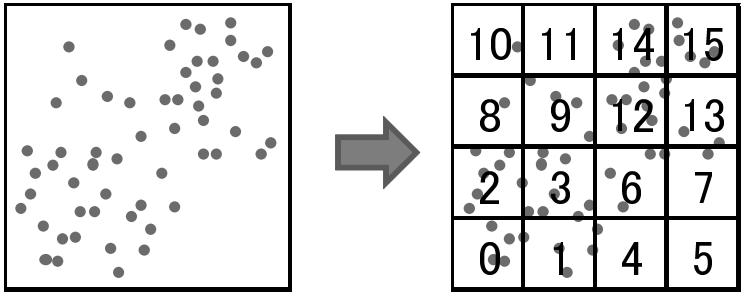

図 1 空間のセル分割
Fig. 1 Space Partitioning into Cells

そこで，k 最近傍は探索点から近い距離で見つかるという性質から，データの比較回数を抑えコスト削減を狙うため，まず狭い範囲に絞って k 最近傍点を探すことを考える．具体的には，図 1 のように空間をセルに分割することを考える．データの分布や k の値にもよるが，あるデータの k 最近傍の候補はそのデータが含まれるセルの中，もしくはその周囲のセルを順に見れば見つかる．そのため，極端に離れたセル同士のデータは比較する必要がなくなり，計算コストが抑えられる．例えば，図中のセル番号 1 のデータとセル番号 14 のデータはかなり離れているため k 最近傍に関与することはほとんどない．近くのセルのみに注目すればいいという局所性は，MapReduce のフレームワークと相性が良い．

3.2 セル内での局所的な最近傍処理

セルに分割した場合，まず，各セルの中に含まれる点集合のみを対象として k 最近傍処理を局所的に進めることが考えられる．各点の周辺のデータの分布に応じて，処理結果を以下のように 3 通りに分類することができる．

ケース A 対象の点を含むセルの中に k 個の最近傍点が存在し，k 番目の最近傍点までの距離を半径とする円を描いたときに他のセルの領域と重ならない場合にあたる．図 2 における点 A がこのケースに相当する[⋆1]．ここでは $k = 2$ の場合を考えるが，着目しているセル内を調べただけで点 A の k 最近傍が確定するため，他のセルを調べる必要がない．セルの中心に近い点がこのケースとなりやすい．問合せ処理を早期に終了できる

⋆1 ここでは点 B, C はないものとして考える．他のケースについても同様．

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

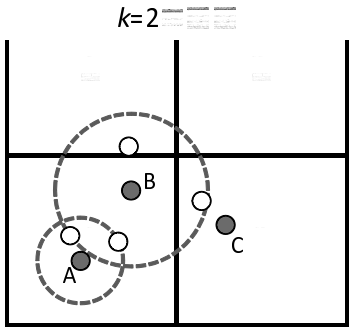

図 2 局所的な k 最近傍問合せ処理結果の分類
Fig. 2 Classification of Local kNN Query Processing Results

ため，望ましい状況である．

ケース B 対象の点を含むセルの中に k 個の最近傍点が存在するが，k 番目の最近傍点までの距離を半径とする円を描いたときに他のセルの領域と重なる場合にあたる．図の点 B がこのケースに相当し，円はセル 1, 2, 3 に重なっている．着目しているセル内に少なくとも k 個の最近傍点の候補が存在するが，隣接するセル内にまだ見ていない k 最近傍候補が存在する可能性がある．そのため，現在の k 最近傍点までの距離を半径とする円と重なるセルを探索する必要がある．実際，図においては，セル 1 内の点に B により近いものが存在する．一般にセルの境界付近の点がこの状況になりやすい．

ケース C 対象の点を含むセルの中に存在する最近傍点の個数が k 個未満である場合にあたる．この場合，現時点での k 番目の最近傍の候補が求められていないため，ケース A, B で用いた k 最近傍への最大距離が不明となる．図では点 C がこの状況に該当し，どのセルまで調べる必要があるのか確定できない．データ数の少ない疎な領域でこの状況が発生しやすい．k 最近傍を求めるためには，未探索の周囲のセルを近い順にアクセスすることになる．アクセスの結果，ケース A ないし B へと状況が変化すれば，先に示した方法で k 最近傍を確定することができる．

以下で提案する手法は，セル分割により局所的な問合せ処理を行い，その結果を効果的に活用するというアイデアに基づいている．そこでは上に述べた 3 つの場合分けが有効に活用される．まず，一様なセル分割に基づく基本的なアプローチについて述べる．

4. 手法 1：一様なセル分割に基づく手法

ここでは，セル分割が一様になされているとした場合の問合せ処理方式について述べる．データ集合のそれぞれのデータには一意にそれ特定できる ID（例：A001）が割り振られており，また，そのデータの位置を表す座標情報が含まれているとする．さらに，MapReduce 処理の前に，空間がどのようにセル分割されているかの情報が各ノードに配布されているものとする．

処理は 3 つのフェーズ（MapReduce ジョブ）で構成される．以下でそれぞれのフェーズでの処理を詳しく見ていく．

フェーズ 1：局所的な k 最近傍計算

このフェーズでは，セル単体で局所的に見たとき，データ集合の各データについて k 最近傍となるデータの ID とその距離を求める．図 3 にその処理の概要を示す．

Map	入力	データ集合
	処理	必要なデータの抽出
	出力	key: セル番号，value: ⟨ID, 座標 ⟩
Reduce	入力	Shuffle 処理によりセル番号ごとにまとめられた ⟨ID, 座標 ⟩
	処理	同じセルに割り当てられた点のすべてのペアに対する距離計算
	出力	key: ID, value: ⟨ 座標, セル番号, 現時点の k 最近傍の ID と距離のリスト ⟩

図 3 フェーズ 1 の処理内容
Fig. 3 Processing Phase 1

Map 関数ではデータ集合から今後の計算で必要となるデータの ID と座標を取り出し，セル番号をキーとして出力する．Shuffle 処理によりセル番号が同じデータが 1 つの計算ノードに集められ，Reduce 関数では集まった点集合のすべてのペアについて距離計算を行う．その出力は，キーとしてデータ集合のデータ ID が用いられ，値としてはそのデータ ID の座標およびセル番号だけでなく，計算結果として得られた現時点の k 最近傍点の情報（ID とその距離）が出力される．この MapReduce 処理の様子を図 4 に示す．なお，状況によっては最近傍点の候補が k 個未満しか得られていない場合もある．

フェーズ 2：周辺のセルのチェック

フェーズ 1 では各セルの中だけに着目して k 最近傍の候補を求めた．このフェーズでは，そこで得られた k 番目の最近傍への距離の情報から，他に探すべきセルがあるかどうかを

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

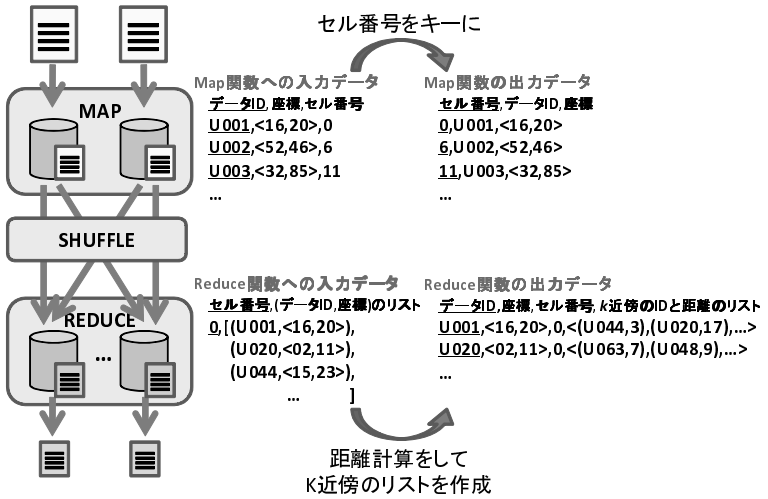

図 4 フェーズ 1 の MapReduce
Fig. 4 MapReduce of Phase 1

Map	入力	フェーズ 1 の出力

	処理
	＋
	出力
		if 着目しているデータ x に k 個の最近傍候補が存在 **then**
		x の座標を中心に最大距離の円を描き，他に見るべきセルがあるかを計算
		if 他に見るべきセルがない場合 **then**
		key: FINISH, value: ⟨ID, 現状 k 最近傍 ⟩ を出力
		else
		key: 見るべきセルの番号, value: ⟨ID, 現状の k 最近傍 ⟩ を出力
		// 見るべきセルが複数存在する場合，その数だけ key-value ペアを出力
		end
		else
		見るべき周辺のセルを 1 ないし複数個選定
		key: 見るべきセルの番号, value: ⟨ID, 現状の k 最近傍 ⟩ を出力
		end

Reduce	入力	Shuffle 処理によりセル番号ごとにまとめられた ⟨ID, 座標 ⟩ の集合
	処理	同じセルに割り当てられた点のすべてのペアに対する距離計算 既存の k 最近傍より近いものがあれば入れ替える
	出力	key: ID, value: ⟨ 座標, 候補となるセル番号のリスト, k 最近傍の ID と距離のリスト ⟩

図 5 フェーズ 2 の処理内容
Fig. 5 Processing Phase 2

確かめ，必要ならそれらのセル中の点との距離計算を行う．図 5 にその処理の概要を示す．具体的には，Map 関数では，現状で最も遠い k 番目の最近傍データまでの距離を半径とした円が未探索のセルの領域に重なっているかどうかの判定を行う．最大距離の円が図 2 の点 A のように他のセルと重ならない場合は，この時点でそのデータについての k 最近傍は確定する．その場合，終了したことを示す FINISH というキーを与え，値に ID と k 最近傍のデータを格納して出力とする．

最大距離の円が図 2 の点 B のように他のセルと重なる場合は，重なったセルそれぞれについて，そのセル番号をキー，値は元々の入力データ（データ ID，座標，k 最近傍情報）に，重なったセルのセル番号の情報を追加して出力する．点 B の例では，円がセル番号 1，2，3 の領域に重なっているので，キーだけが 1，2，3 と異なる 3 つのデータが出力される．図 2 の点 C のように，セルの中に k 個の点がなく最大距離の円が描けない場合は，周辺の 1 ないし複数（具体的には何らかの手法で設定）のセルを調べるために，点 B の時と同様，キーだけが異なる複数のデータを出力する．この MapReduce 処理の様子を図 4 に示す．下線があるものがキー，残りが値となっている．

Reduce 関数ではフェーズ 1 と同様にセル番号が同じデータが集められるので，集められた点のペアについて距離計算を行う．フェーズ 1 と異なるのは，既に別のセル領域で求めた k 最近傍の情報を持っている点である．新規に計算したデータとの距離が既存の k 最近傍のものより近いならば，データを入れ替えることになる．Reduce 関数の出力の形式はフェーズ 1 の場合と同じになる．

フェーズ 3

フェーズ 3 では，フェーズ 2 で得られた，それぞれの領域で見つけた k 最近傍候補を合わせて，最終的な k 最近傍ををを求めることが目的となる．図 5 にその処理の概要を示す．

Map 関数の出力データについてだが，これまで見てきたフェーズ 1, 2 では Map 関数の出力はセル番号をキーとしていたが，ここではデータの ID をキーとしている．データの ID をキーにすることにより，そのデータについて各セルで計算された k 最近傍の結果が，Shuffle 処理により集まることになる．Reduce 関数では，Shuffle 処理で集まった各セルでの計算結果からより距離が近い k 個の点を選び，新たな k 最近傍とする．出力の形式は，他のフェーズと同じである．この MapReduce 処理の様子を図 8 に示す．

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

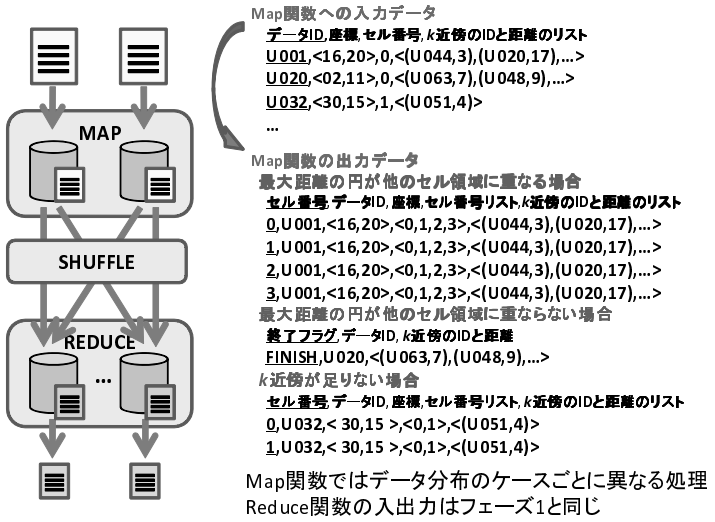

図 6 フェーズ 2 の MapReduce
Fig. 6 MapReduce of Phase 2

Map	入力	フェーズ 2 の出力
	処理	必要なデータの抽出
	出力	key: ID, value: ID に対応するデータ
Reduce	入力	Shuffle 処理により ID ごとにまとめられた，フェーズ 2 において各セルで計算された K 近傍のデータ
	処理	各セルでの k 最近傍を統合し k 最近傍を更新する
	出力	key: ID, value: 〈 座標, 候補となるセル番号のリスト, k 最近傍の ID と距離の情報 〉

図 7 フェーズ 3 の処理内容
Fig. 7 Processing Phase 3

議　　論

以上が本研究で提案する第一の手法である．なお，1 点だけ注意が必要である．図 2 の点 B のように調べるべき最大距離が分かっている場合は，この 3 つのフェーズを順に適用することで，k 最近傍が求まる．しかし，点 C のように k 最近傍候補が少なく最大距離が不明な場合は，3 つのフェーズを順に適用しただけでは k 最近傍が確定しない可能性がある．

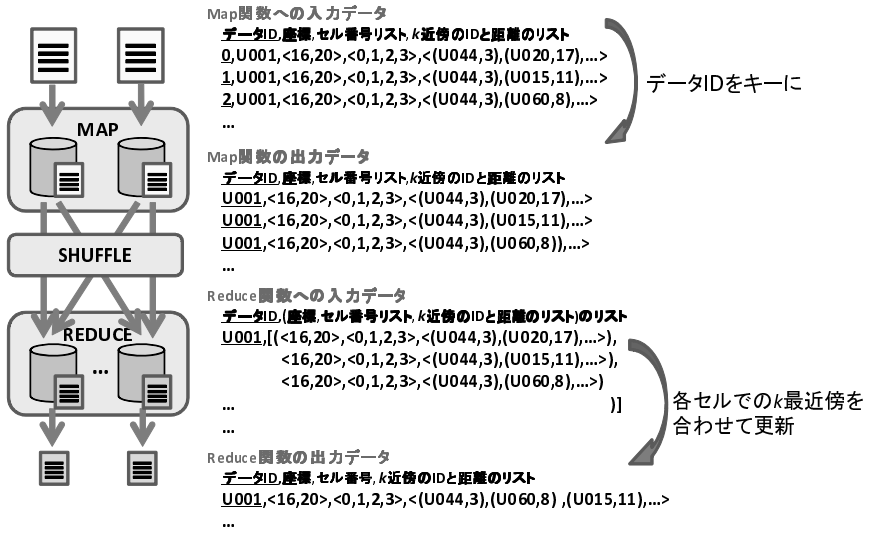

図 8 フェーズ 3 の MapReduce
Fig. 8 MapReduce of Phase 3

このときは，フェーズ 2 とフェーズ 3 を繰り返して処理することで対処する．繰り返しの度に探索範囲を広げることで k 最近傍候補のデータが増えていくため，その数が k 以上となった時に最大距離の円を描くことができ，その重なりを調べることで k 最近傍が確定する．与えられたデータ集合のすべての要素について k 最近傍が確定した時点で AkNN 問合せが終了できる．

5. 手法 2：分布情報を用いた非一様なセル分割に基づく手法

4 章の手法は，データが空間に均一に分布している場合を想定したアルゴリズムである．データが均一な場合は，各セルに含まれるデータ数もほぼ同じになり，MapReduce 処理での各計算ノードのデータ処理量も平均化されやすい．しかし，データの分布に極端な偏りがあると，密な部分と疎な部分でのデータ処理量に大きな差が生じ，分散環境を十分に生かせなくなってしまう．この問題を解決するために，データ分布の統計情報をもとにセル分割を調整する手法を提案する．セル分割を行って各データの位置するセルのセル番号が決まれば，後のデータ処理は 4 章で解説した 3 つのフェーズからなる MapReduce 処理をそのま

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

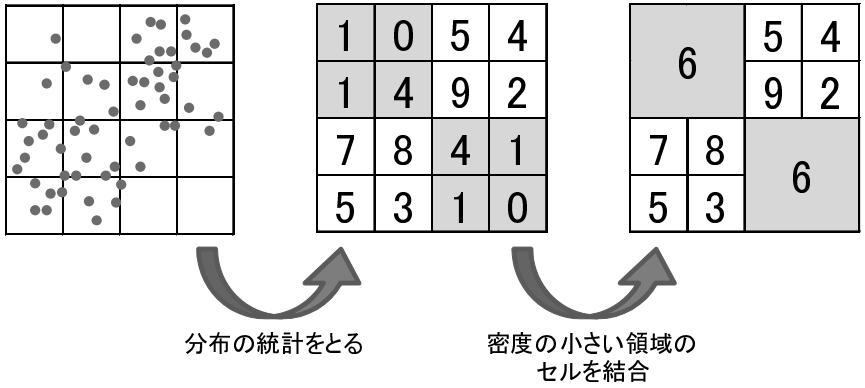

図 9 分布の統計によるセル分割の調整
Fig. 9 Adjustment of Cells Using Distribution Statistics

ま適用できる．そのためこの章では，データの分布に応じたセル分割のアプローチについてのみ説明する．

5.1 セル分割の調整方式

空間をセルに分割したときにデータ処理量に大きな差を生じないようにするためには，各セルに含まれるデータの量をなるべく均等にする必要がある．そのために，最初に細かいセル分割をした上で，それぞれのセルにどれだけのデータが存在するかの統計を採る．得られた統計情報から，特に密度が低い領域のセルを結合させることで，セル内のデータの量を一定以上にする．この様子を図 9 に示す．この例では密度が低い領域を結合しているが，反対に，密度が高すぎるエリアをさらに細かく分割することもデータ処理量の均一化に有効だと考えられる．

ここでは上記のような簡単なアイデアを提案したが，データの分布に応じたセル分割の手法については検討すべき点がいくつか存在する．点の分布に応じて，各セルにおける処理にどの程度のコストが必要となるかというコスト見積もりが行えれば，各セルの処理の負荷ができるだけ均等になるように調整することも可能であると考えられる．このような問題は空間データベースにおける選択率推定[1] やヒストグラム構築などとも関連が深い領域である．今後の課題として検討を行いたい．

5.2 MapReduce による分布の統計の計算

分布の統計の計算は MapReduce を利用して簡単にできる．分布の統計を取りたいデータ

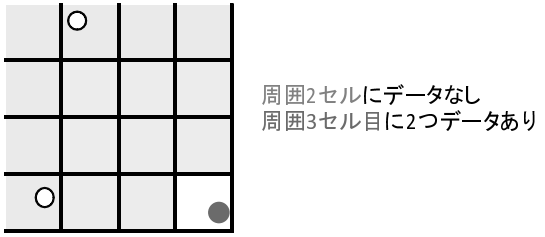

図 10 周囲にデータがない例
Fig. 10 Example: No Data Around

集合を Map 関数の入力とする．また，カウント処理のため，細分化された多数の均一のセルからなるセル分割を事前に作成しておく．Map 関数での処理としては，それぞれのデータの座標情報からそのデータがどのセルに属するかを計算し，結果のセル番号をキーに，1 という数字を値として出力する．こうすることにより，Shuffle 処理により Reduce 関数の入力は，⟨ セル番号，[1,1,1,...]⟩ という形になる．Reduce 関数の処理としては，並んだ 1 の数をカウントし，セル番号とそのカウントした値をペアにして出力する．このような処理は MapReduce が得意とするものであるため，効率的な処理が可能である．

5.3 動的なセル分割のメリット，デメリット

動的にセル分割を行うことに関するメリットとデメリットについて議論する．

動的な分割のメリット

まずメリットについてだが，元々の目的である MapReduce 処理時の計算量の均一化が挙げられる．調整により負荷の偏りが小さくなることで，処理全体のスピードアップが狙えることになる．

さらに，セル分割を調整する際に，「各セルに必ず k 個以上のデータが存在する」という条件を付けることで，手法 1 で述べたような MapReduce 処理の余分な繰り返しをなくすこともできる．MapReduce 処理のループの原因は，図 2 での点 C のような，k 近傍の最大距離がわからないデータである．しかし，セル分割の調整の際に先のような条件を付ければ，点 C のようなデータを生み出さないようにできる．これによって MapReduce 処理のループをなくすことができる．

手法 1 のように動的な調整をしない手法の場合，点 C のようなデータが存在すると，少なくとも k 個のデータが見つかるまで，周囲のセルを順に探索範囲を広げることにより k 最近傍を求めることになり，その結果 MapReduce 処理の回数が増えてしまう可能性がある．

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

処理の手法上，例え点 C のようなデータが存在しても，周囲のセルを調べただけでデータが k 個以上見つかり k 最近傍が確定すれば，追加の MapReduce ループを行う必要はない．しかし，図 10 のように周囲にデータがまったくないと，探索範囲を広げるたびに MapReduce 処理のループを増やす必要が出てくる．図では周囲 2 セルにまったくデータがなく，周囲 3 セルを調べた時ににやっとデータが 2 つ見つかるので，$k = 2$ というゆるい条件のときでも探索範囲を広げるために MapReduce 処理の繰り返しを行う追加する必要がある．さらに，見つかったデータの位置によっては，最大距離の円が未探索の領域に重なって，もう 1 回の繰り返しを追加する必要がある可能性もある．

このような動的なセルの調整をしない場合は，MapReduce 処理の回数がデータ分布によって増加する可能性がある．しかし，セル分割の調整により必ずセルの中で k 個以上のデータが見つかると保証されていれば，点 C のような点は生まれないために追加の MapReduce 処理を行う必要はなく，3 つの MapReduce フェーズを 1 回ずつ行うだけで k 最近傍を求めることができる．

動的な分割のデメリット

セル分割を調整することでデメリットとしては，調整のための統計情報を得るために，MapReduce 処理が 1 回増えることが挙げられる．また，統計情報を基にした調整そのものについても計算コストがかかることになる．

メリット，デメリットについてまとめたものが以下である．セル分割の調整のために掛かるデメリットよりもそれによって得られるメリットが上回れば，この手法を採用する決め手となる．

メリット

負荷バランスの均一化

MapReduce 処理のループをなくす

デメリット

MapReduce 処理が 1 回増加

セルの分割・調整のためのコストがかかる

5.4 セル分割の調整についての考察

もし元々のデータ分布が均一だったならば，各セルに含まれるデータ量も均一になるため，MapReduce 処理での計算ノード間の負荷バランスも均等になる．そして，図 10 のような極端なデータ配置にもならないため，MapReduce 処理のループが増えることもない．さらに，そもそも分割の調整が偏りの撤廃を目的としているため，元々偏りのない均一なデータが相手では調整しようがない．結果，統計を取るための MapReduce 処理のコストが無駄に掛かることになる．

以上のように，均一なデータ分布を対象にセル分割の調整をする手法を適用した場合は，メリットがなくデメリットのみが生じる．よって，調整をするか，しないかのどちらの手法が優れているかは全 k 最近傍を求めたいデータの分布のパターンによって変わることになる．

6. 今後の課題

ここまでで，分散コンピューティング環境で全 k 最近傍問合せを行うための基本的なアルゴリズムと，セル分割を調整することでの負荷バランスの均一化を狙う手法を示した．今後の課題としては，以下に示すものが挙げられる．

分散コンピューティング環境を用意しての実験

実データ，もしくは人工データを対象に提案したアルゴリズムが正しく動作するかの検証を行う必要がある．分散環境では特にスケーラビリティが重視されるので，計算ノードを増やしたときの処理時間の変化に注目する．さらに，セル分割の細かさや，k の値を変化させたときの処理時間の変化も注目したい．

距離計算の効率化

今回の提案では，セル番号で集められたデータ同士の距離計算は単純にすべての組合せで行うというものだった．しかし，例えば計算がまだ途中でも，途中までに求めた値を保持しておくことで，何らかの条件で距離計算をするまでもないデータを無視することができるかもしれない．また，集まったデータについて座標データに関して並べ替え処理をすることで，より効率的に距離計算をできる可能性がある．そのような距離計算についての最適化をすることで，全体の処理時間の短縮を狙う．

セル分割調整のパフォーマンスへの影響調査

セル分割の調整処理において，データの密度が低い領域のセルを結合したり，逆に高い領域のセルをさらに細かく分割するという大まかな方針は示した．しかし，具体的にどの程度性能に影響を与えるかは現時点ではわからない．詳細なコストモデルの構築や実験による調査により，セル分割の調整のベストな値を見つけられるかもしれない．

全 k 最近傍を求めるための別のアプローチの考察

今回は空間をセルという小さなサイズに分割することで効率化を図るという手法を

Vol.2011-DBS-152 No.16
Vol.2011-IFAT-103 No.16
2011/8/3

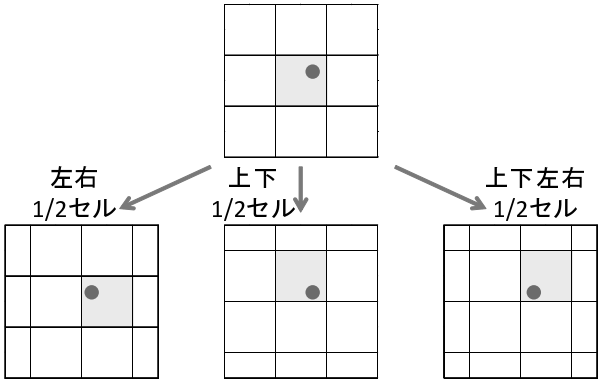

図 11 セル分割のシフト
Fig. 11 Shift of Cell Division

取った．このときの問題として，セルの境界の向こうにあるデータをどう調べるかというものがある．論文では，k 最近傍に含まれる最大距離の円との重なりで調べたが，また別のアプローチとして分割をシフトさせるというものを考えている．図 11 のようにセルの長さの 1/2 のサイズだけずらしたセル分割を見ると，同じデータでもセルの中の位置が変化する．これによって，例えば図 2 の点 A のように，セルの中だけで k 最近傍が確定するデータを増やすことができる．すべてのシフトにおいてセル単独では k 最近傍が確定しなくても，各シフトでの結果を合わせることで，k 最近傍を確定することもできる．しかし，図 10 のような極端な分布が存在する場合など，分割のシフトでは距離が遠すぎるとうまくいかないこともある．また，単純に分割が 4 通りに増えるために計算量が多くなるという問題もある．しかし，1 つ目のシフトの計算結果を利用して以降の計算量を抑えることも考えられるので，適切な最適化を行えば，この手法で提案手法よりも効率的に全 k 最近傍を求められるかもしれない．

7. おわりに

本論文では，並列分散コンピューティング環境で MapReduce に基づく全 k 最近傍問合せを処理するための問合せ処理方式を提案した．さらに，統計情報に基づくセル分割の調整という負荷バランスの均一化を狙う手法も提案した．今後の課題は 6 にまとめた通りであり，これらを解決することが今後の目標である．

謝辞 本研究の一部は科学研究費（22300034）の助成による．

参 考 文 献

1) Acharya, S., Poosala, V. and Ramaswamy, S.: Selectivity Estimation in Spatial Databases, *Proc. SIGMOD*, pp.13–24 (1999).
2) Afrati, F.N. and Ullman, J.D.: Optimizing Joins in a Map-Reduce Environment, *EDBT 2010*, pp.99–110 (2010).
3) Böhm, C. and Krebs, F.: The k-Nearest Neighbour Join: Turbo Charging the KDD Process, *Knowledge and Information Systems*, Vol.6, No.6, pp.728–749 (2004).
4) Chen, Y. and Patel, J.M.: Efficient Evaluation of All-Nearest-Neighbor Queries, *ICDE 2007*, pp.1056–1065 (2007).
5) Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *OSDI*, pp.137–150 (2004).
6) Emrich, T., Graf, F., Kriegel, H.-P., Schubert, M. and Thoma, M.: Optimizing ALl-Nearest-Neighbor Queries with Trigonometric Pruning, *SSDBM 2010*, pp.501–518 (2010).
7) Jiang, D., Tung, A. K.T. and Chen, G.: MAP-JOIN-REDUCE: Towards Scalable and Efficient Data Analysis on Large Clusters, *IEEE TKDE*. (accepted for publication).
8) 西田圭介：Google を支える技術：巨大システムの内側の世界，技術評論社 (2005).
9) The Apache Software Foundation: Hadoop Homepage, http://hadoop.apache.org/.
10) Vernica, R., Carey, M.J. and Li, C.: Efficient parallel set-similarity joins using MapReduce, *SIGMOD 2010*, pp.495–506 (2010).
11) White, T.: *Hadoop*, オライリー・ジャパン (2010).
12) Zhang, J., Mamoulis, N., Papadias, D. and Tao, Y.: All-Nearest-Neighbors Queries in Spatial Databases, *SSDBM 2004*, pp.297–306 (2004).