Vol.2018-DBS-168 No.10
2018/12/22

NoSQLによる集約演算のデータ要約手法を用いた結果推定の高精度化

張涵[1,a)]　欅 惇志[1,b)]　宮崎 純[1,c)]　中村 匡秀[2,d)]

概要：本研究では，分散キーバリューストア (KVS) 上の集約演算結果の推定について，ヒストグラムにカーネル密度推定を導入した手法を提案し，範囲クエリ処理の効率化とクエリ結果の推定精度の向上を目指す．分散 KVS 上での大規模多次元データに対する集約演算は，データの全スキャンが多発し非効率的である．また，このような大規模なデータベースにおける集約演算は，正確な値ではなく概数で良い場合が多い．そこで過去に我々は，ヒストグラムとカーネル密度推定を用いて，データの集約演算結果を推定し，データの全スキャンを回避することで集約演算処理の効率化する手法を提案した．本稿では，過去に提案した複数の推定手法を，範囲クエリに応じて動的に変更することでさらなる推定精度の向上を図る．

1. はじめに

近年，インターネット，スマートフォン等の普及により，ユーザーの位置情報や利用ログといった，ビッグデータと呼ばれる大規模データが収集されるようになった．また同時にビジネス等では，このようなビッグデータを分析することにより，価値ある情報として有効活用することが不可欠になりつつある．

そのような分析の一例として集約演算を使用するものが存在し，大規模データに対する集約演算機能を提供するデータベースとして，スケールアウトが容易な分散 KVS [6] が注目されている．しかしながら多くの場合，分散 KVS は単純なインデックスしか持たない，もしくはインデックスが存在しない．そのため，大規模多次元データにアクセスする際にはデータの全スキャンが頻繁に発生し非効率的である．

このような問題に対して，分散 KVS における大規模多次元データに対する集約演算を効率的に処理する手法として，様々な研究がなされてきた [1], [8], [14], [15], [16]．Watari ら [15] は，リレーショナルデータベース (RDB) [2] と分散 KVS を相互に利用する手法を提案をした．Watari らの提案手法では，データ空間をグリッドと呼ばれる複数の領域に分割し，グリッドごとの集約演算結果を事前に計算し保持する．これによって，グリッドがクエリ範囲に完全に内包される場合には，対象グリッド内のデータの全スキャンを行うことなく事前に計算された集約演算結果を参照することで処理の効率化を達成した．しかし，範囲クエリに完全に内包されないグリッドについては，依然としてグリッド内のデータの全スキャンを行っている．これは，範囲クエリに完全に内包されない部分に位置するグリッドが多い場合や，グリッドあたりのデータ数が多い場合は，スキャンするデータ数が大量になることがある．

これに対し我々は，大規模なデータベースにおける集約演算では正確な値ではなく概数で良い場合が多いことを踏まえ，Watari らの提案手法にヒストグラムとカーネル密度推定を用いたデータ要約を導入し，集約演算処理時にはその結果の推定値を計算する手法 [17], [18] を提案した．この手法は，グリッドからヒストグラムを構築し，ヒストグラムの各バケットに対してカーネル密度推定を利用して得た統計情報を保持し, クエリの処理にはこの統計情報のみを用いるものである．我々は，この手法を用いてクエリ処理時のデータの全スキャンを完全に回避することで，高いクエリスループットが実現できることを示し，また推定値と真値の平均誤差が 5% 以下の推定精度を達成した．

本稿では，グリッドの範囲，クエリ範囲，エラー率の関係性を調査し，過去に我々が提案した複数の推定手法を，クエリに応じて動的に変更することでさらなる推定精度の向上を目指す．そして，クエリスループットと推定精度に関して，提案手法，我々の過去の手法，Watari らの手法で評価実験を行い，提案手法の優位性を検証する．

[1] 東京工業大学
[2] 神戸大学
[a)] zhang@lsc.cs.titech.ac.jp
[b)] keyaki@lsc.cs.titech.ac.jp
[c)] miyazaki@cs.titech.ac.jp
[d)] masa-n@cs.kobe-u.ac.jp

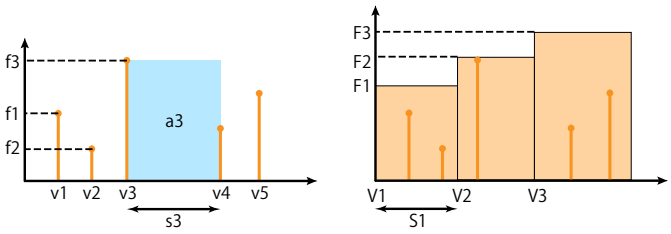

図 1　データ分布とヒストグラムの例

2. 基礎知識

2.1 ヒストグラム

データ分布は属性値 (AttributeValue) と対応する度数 (Frequency) で構成される．ヒストグラムは階級幅 (属性値の幅) とその階級幅内に対応する度数の総和の組（この組をバケットと呼ぶ）によって構成され，データ要約及びクエリ結果の推定の手法として広く使用されている方法である [3]．データ分布がどのように分割されるかによって，構築されるヒストグラムが大きく変化する．図 1 は一次元のデータ分布とそのヒストグラムの例である．以下に示す変数定義を用いて，代表的なヒストグラムへの分割法について説明する．

- v_i：データ分布での i 番目の属性値
- f_i：データ分布での i 番目の属性値に対応する度数
- s_i：データ分布での i 番目の属性値の幅 $(v_{i+1} - v_i)$
- a_i：データ分布での i 番目の属性値の面積 $(f_i * s_i)$
- V_i：i 番目バケットの定義域の最小値
- F_i：i 番目バケット内の度数の総和
- S_i：i 番目バケットの幅 $(V_{i+1} - V_i)$
- n ：全バケット数

Equi-width histogram [5] は，データ分布の定義域 (属性値の幅) を等幅に $n-1$ 分割を行う $(S_{i+1} = S_i)$．*Equi-depth histogram* [10] は，可能な限り分割後の各バケットの度数の総和が等しくなるように分割を行う $(F_{i+1} = F_i)$．*V-optimal histogram* [4] は，バケット分割後の各バケット内の度数の分散が最小になるように分割を行う．*Max-diff histogram* [11] は，隣接する属性値間の面積差 $(a_{i+1} - a_i)$ を計算し，上位 $n-1$ 件の属性値間で分割を行う．*Compressed histogram* [11] は，データ分布の度数の全総和を $SumF$ とするとき，$f_i > SumF/n$ を満たす属性値を単一のバケットとし，満たなさいものについては Equi-depth histogram と同様の処理をするような分割を行う．Poosala ら [11] の各ヒストグラムを用いた範囲クエリ結果の推定の評価実験によると，Max-diff histogram がもっとも精度の良い推定となることが報告されている．

また，多次元データ分布をヒストグラムへと分割する方法の一つとして MHIST [12] と呼ばれる手法がある．MHIST 分割アルゴリズムを以下に示す．

Step1　データ分布について各次元軸に対してそれぞれ周辺分布を計算し，これをもとに最も分割の必要性の高い次元軸を決定する．最も分割の必要性の高い次元軸とは，いずれの分割手法を採用するかによって判定基準が異なる．V-optimal であれば分散が最も大きい周辺分布を持つ次元軸，Max-diff であれば周辺分布にて隣接面積差が最も大きいものを持つ次元軸，Equi-width，Equi-depth，Compressed であれば周辺分布の総和が最も大きい次元軸が選択されることになる．

Step2　Step1 で選択した次元軸を基準にデータ分布を p 分割する．

Step3　2 回目以降は，複数のバケットが存在しているので，バケットごとにデータ分布の各次元軸について周辺分布を計算し，全バケットで最も分割を行う必要性の高い次元軸を決定する．

Step4　Step3 で選択した次元軸が属するバケットを p 分割する．

Step5　分割上限バケット数に達するまで Step3 と Step4 を繰り返す．

分割方法と分割数 p によっては最終的なヒストグラムが大きく異なり，範囲のクエリ結果の推定の精度にも影響を与える．Poosala ら [12] の評価実験では，分割方法を Max-diff histogram，分割数を $p = 2$ と設定すると最も良い推定精度となることが報告されている．

ここで最も良い推定精度となる，MHIST (Max-diff histogram, $p = 2$) によるデータ分布の分割ついてを具体的に説明する．図 2 は 2 次元のデータ分布を五つのバケットに分割する過程を表している．各データは次元軸 A1 と次元軸 A2 の 二軸の属性を持ち，縦軸及び横軸の値は次元軸 A1 の属性値と次元軸 A2 の属性値を表す．また，図中の点は各属性値に属するデータ群であり，データの横の数字はデータ数を表す．まず，各次元軸に対して属性値ごとに周辺分布を計算する (図 2 の Step 1)．その後各次元軸にて隣接する属性値間の面積 (属性値の幅 * 属性値に対応する周辺分布) の差を計算する (図 2 の Step 2)．今回の例の場合，属性値は連続する整数値なので，面積差は単に隣接する周辺分布間の差となる．計算の結果，もっとも差が大きいのは次元軸 A2 の属性値 3 と 4 の間であるので，ここで 1 回目の分割が行われる．2 回目以降は，各バケットの各次元軸に対して同様の計算を行い，分割箇所を決定する (図 2 の Step 3)．この計算をバケット数が分割上限数に達するまで繰り返し行う．今回の例の場合，最終的には図 2 の Step 4 のようになる．

2.2 カーネル密度推定

カーネル密度推定 (KDE) [13] とは，有限の標本点から全体の分布を推定するノンパラメトリックな推定手法の一つである．特定の範囲に属するデータを集計するヒストグ

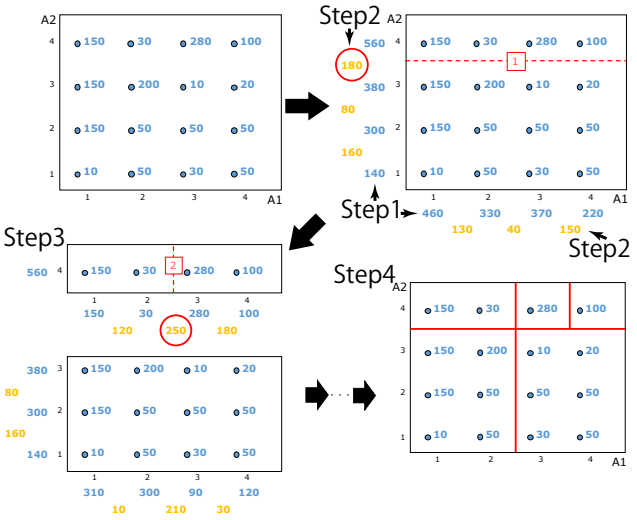

図 2　MHIST による分割の様子

ラムとは異なり，カーネル密度推定は各データ点を中心とした分布を想定し，この重ね合わせをデータ集合の分布とするものである．

同じデータでも階級の境界の設定によって見た目が大きく変化するヒストグラムに対し，カーネル密度推定では階級の境界に依存せずに分布を捉えることが可能であり，分布の複峰性などの特徴がわかりやすい．

ただし欠点としては，標本データを全て保持しておく必要がありメモリ効率が悪いことや，新しいデータ点が追加された際に再度全ての標本点について密度計算を行う必要があることがあげられる．

3. 先行研究

3.1 部分集約法

部分集約法 [9], [15] とは，データベースを複数のブロックに分割してブロックごとに集約演算結果を事前計算した上で，集約演算のクエリを処理する際には事前計算結果を可能な限り再利用することで，データベースへのアクセスを削減して処理の効率化を図る手法である．

例として，年齢 (age) と身長 ($tall$) からなるリレーション B に対して，$age < 15$ を満たすレコードの $tall$ の総和を求める集約演算を考える．このとき，B は三つのブロック B_1, B_2, B_3 に分割されており，各ブロックについて総和が事前に計算されているとする．また、各ブロックについて以下の情報が与えられているとする．

- ブロック B_1 : age は全て 15 未満
- ブロック B_2 : age は全て 15 以上
- ブロック B_3 : age は 15 未満と 15 以上が混在

これより，目的の集約演算を処理する際には，データの全スキャンはブロック B_3 に対してのみ行えばよく，ブロック B_1, B_2 のデータをスキャンする必要はない．

このように部分集約法は，部分集約演算結果を事前に求めておくことで，実データへのアクセスを省略して集約演算を効率化する．

3.2 部分集約法を利用したクエリ効率化

Watari らは，部分集約法を利用した大規模多次元データに対する範囲クエリを効率化する手法 [15] を提案した．Watari らの手法では，まず分割後の領域（これをグリッドと呼ぶ）1 つあたりに入るデータ数（これをグリッドサイズと呼ぶ）を指定して，データ空間を複数のグリッドに分割する．その後，各グリッドについて事前に計算した集約演算結果を保存し，クエリ処理時には集約演算結果を再利用することで処理の効率化を達成している．またこのとき，データサイズは小さいがクエリ処理時には複雑な検索の対象となるグリッドの分割情報はインデックスを用いることができる RDB に，各グリッドについての集約演算結果はスケールアウトを実現しやすい KVS に保存している．

範囲クエリが与えられた際は，以下のアルゴリズムによって処理が行われる．

Step1　範囲クエリと共通部分を持つグリッドを RDB のテーブルから列挙する．このとき同時に，グリッドが範囲クエリに完全に包含されるかどうかも問い合わせる．

Step2　Step1 で列挙されたグリッドのうち，範囲クエリに完全に包含されるグリッドについては，KVS からそれぞれの部分集約結果を取得する．

Step3　Step1 で列挙されたグリッドのうち，範囲クエリと一部が交わるグリッド（これを周辺グリッドと呼ぶ）に含まれるデータをすべてスキャンし，範囲クエリに含まれるデータについて集約演算を行う．

Step4　Step2 および Step3 で得られた集約演算結果を全て統合して最終的なクエリ結果を得る．

Watari らの実験では，約 1,000 万件のデータへの 4 次元の範囲クエリに対して，提案手法はグリッド一つあたりに入るデータ数を適切に設定することで，HBase(KVS の一種) を単体で用いた場合に比べ 5.4 – 36.3 倍，PostgreSQL(RDB の一種) に対しては 2.9 – 13.8 倍のクエリスループットを実現している．

しかし，Watari らの提案手法 (以降，All-Scan と呼ぶ) の問題点として，グリッドサイズを大きく設定すると，範囲クエリに完全に内包されるグリッドの割合が減少し，結果として実データのスキャンが比較的減少しない．また，グリッドサイズを小さく設定すると，範囲クエリに完全に内包されるグリッドの割合が増加し，実データのスキャン量は減少するもの，グリッドの数が増加し，グリッドの管理コストの増加により，クエリスループットは必ずしも高くならないということがあげられる．

3.3 データ要約を利用した集約演算結果の推定

我々が過去に提案した手法 [17], [18] では，Watari らの

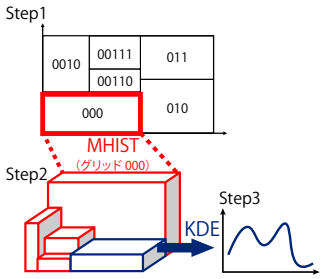

図 3　データ要約の手順

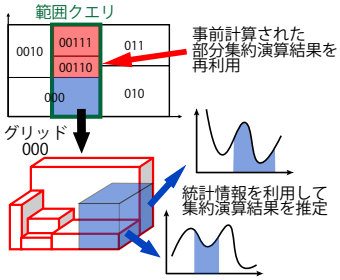

図 4　推定の手順

手法 [15] における各グリッドに対して，ヒストグラムの作成 (バケット分割) 及び，ヒストグラムの作成とカーネル密度推定を組み合わせた方法を用いてデータ要約を行う．この結果（これを統計情報と呼ぶ）を保持しておき，そして，クエリ処理時には統計情報を利用して集約演算結果の推定を行うことで，周辺グリッドについても全スキャンを省略することによって更なる処理の効率化を目指した．

以下では，事前処理であるデータ要約の手順と，統計情報を用いて集約演算結果を推定する手順について説明する．

3.3.1　データ要約

Step1　Watari らの提案手法を用いてデータ空間を複数のグリッドに分割し，部分集約演算の結果を保持する．図 3 の Step 1 では 5 回のグリッド分割の結果，6 個のグリッドに分割される．

Step2　各グリッド内のデータ分布を MHIST (Max-diff histogram，$p = 2$) (2.1 節) を用いて複数バケットに分割する．図 3 の Step 2 は，グリッド 000 のバケット分割の例である．

Step3　さらにカーネル密度推定を用いる場合には，構築した各バケットについてバケット内のデータ分布をもとにカーネル密度推定を行い，その結果を保持する．(図 3 の Step 3)

3.3.2　推定

範囲クエリが与えられたとき，クエリに完全に内包されているグリッドについては事前に計算済みの部分集約演算を用いる．一方，完全に内包されていない周辺グリッドについては，事前処理でのカーネル密度推定によって得られた統計情報を使用し，集約演算結果を推定する．

図 4 にクエリ処理の具体例を示す．グリッド 00111 とグリッド 00110 はクエリに完全に内包されるため，事前計算された部分集約演算結果を用いる．グリッド 000 の一部の領域については，統計情報を用いて集約演算結果を推定する．これら 3 個のグリッドの部分集約演算結果を統合し，最終的なクエリ結果を得る．

この際の統計情報を用いて集約演算結果を推定する手法について，我々は以下の 3 種を提案した．

(1) MHIST

はじめに，グリッドに 2.1 節にて説明した MHIST (MaxDiff Histogram, p=2) による分割を適用した後，範囲クエリが与えられた際は MHIST によって作成されたヒストグラムを用いて，範囲クエリを満たす部分の集約演算結果の推定を *Uniform Scheme* [7] と呼ばれる方法によって行うことを考えた．この手法では，MHIST による分割後の各バケット内のデータ分布を一様分布と仮定し，あるバケット B が範囲クエリ Q と交わるような位置関係にある場合，B 内に関して Q を満たすデータの総和 $Sum(B, Q)$ を以下の式で算出する．

$$Sum(B, Q) = F_B \cdot \frac{Volume(B \cap Q)}{Volume(B)}$$

ただし，F_B は B 内の度数の総和とする．この手法を以降，*MHIST* と呼ぶこととする．

(2) All-KDE

Uniform Scheme による推定では，分割後のバケット内のデータ分布を一様分布と捉え，バケット全体に対してバケットとクエリが交わる部分の割合を用いて計算を行ってきた．しかしながら，大規模な多次元データに対しバケット内のデータ分布を一様分布として扱うことによって推定精度が悪くなる場合がある．そこで，カーネル密度推定を利用した事前計算から得た統計情報を用いることとする．カーネル密度推定を利用することにより，詳細にデータ分布が把握できるため，Uniform Scheme と比較して，より精度の高い推定が行える．MHIST によるバケット分割後，各バケット内のデータ分布にカーネル密度推定を用いて，具体的なデータ点の属性値の代わりに統計情報を保持する．このとき，バケットの各次元軸の属性値の範囲を n 等分し，n^d (d はデータ分布の次元数) 個の点における統計情報を保持することとした．したがって，$Sum(B, Q)$ は以下の式で計算される．

$$Sum(B_i, Q) = \sum_{K_{ij} \in Q} K_{ij}$$

$K_{ij} \in Q$ は，バケット B_i の j 番目の推定点がクエリ Q の範囲内にあることを意味する．この手法を以降，*All-KDE* と呼ぶこととする．

(3) Part-KDE

All-KDE は全てのバケットについてカーネル密度推定を行うことで各バケット内のデータ分布を詳細に捉えることができるが，保持する統計情報のデータサイズは増大し，結果としてクエリスループットの低下を招く原因となる場合がある．MHIST(Max-diff histogram, $p = 2$) による分割の結果，バケットによってはある次元軸の属性値の幅がない (その次元の属性値の最小値と最大値が一致する) 場合がある．このようなバケットを，次元が落ちたバケットと呼ぶことにする．All-KDE に対し，この手法では，分割の結果次

表 1　クエリスループットの比較

手法	スループット (クエリ/秒)
All-Scan	28.64
MHIST	58.07
All-KDE	50.78
Part-KDE	53.17

表 2　誤差率の比較

手法	最大値	平均値	標準偏差
MHIST	0.731	0.250	0.152
All-KDE	0.229	0.038	0.039
Part-KDE	0.260	0.046	0.045

元が落ちたバケットについては，バケット内のデータ分布を一様分布として捉えても差し支えないと仮定し，カーネル密度推定を行なわない．次元が落ちたバケットが範囲クエリを満たす場合は，Uniform Scheme を用いて推定値を算出する．次元が落ちのないバケットについては，手法 1 と同様にカーネル密度推定を行い，推定では事前計算した推定点における推定値を用いる．したがって、$Sum(B,Q)$ は以下のような計算式となる．

$$Sum(B_i,Q)=\begin{cases} F_{B_i}\cdot\frac{Volume(B_i\cap Q)}{Volume(B_i)} & (B_i \text{ が次元落ち}) \\ \sum_{K_{ij}\in Q} K_{ij} & (\text{その他}) \end{cases}$$

これによって，部分的にカーネル密度推定を行わず統計情報のデータサイズを削減することで，推定精度の悪化を軽減しながら，All-KDE と比較して高いクエリスループットが実現できる．この手法を以降，*Part-KDE* と呼ぶこととする．

上記 3 種の推定手法に関する我々の評価実験 [17], [18] の結果，表 1 に示す通り，All-Scan と比較して高いクエリスループットを達成できることを示した．また，表 2 は以下の式で定義される誤差率を各推定手法に関して測定を行った結果である．

$$\text{誤差率}=\frac{\text{誤差}}{\text{真値}}=\frac{|\,\text{真値}-\text{推定値}\,|}{\text{真値}}$$

この結果より，MHIST にカーネル密度推定を組み合わせることで，平均誤差 5% 以下を達成できることを示した．また，我々の手法は範囲クエリに完全に包含されるグリッドについては事前計算された部分集約演算結果を利用して正確な値を取得し，周辺グリッドに該当する部分に関してのみ統計情報を用いて集約演算結果を推定するので，従来のヒストグラムを用いた集約演算結果の推定手法 [4], [7], [10], [11], [12] よりも高い精度で推定を行えることを示した．

4. 提案手法

本稿では，クエリ範囲の周辺グリッドへの重複率（以降，オーバーラップ率と呼ぶ）と推定誤差の関係性に着目し，オーバーラップ率に基づいて推定方法を動的に変更することにより、推定精度をさらなる向上を目指す．

表 3　クエリと周辺グリッドの例

	真値	推定値	誤差	誤差率	誤差指数
Q_1	10001	15003	5002	50%	—
G_{11}	10000	15000	5000	50%	0.5000
G_{12}	1	3	2	200%	0.0002
Q_2	11	14	3	27%	—
G_{21}	10	11	1	10%	0.0909
G_{22}	1	3	2	200%	0.1818

4.1 用語定義

(1) オーバーラップ率

クエリ Q と，クエリ Q の周辺グリッドの一つである G_i とのオーバーラップ率 OR_i を以下のように定義する．

$$OR_i=\frac{Volume(G_i\cap Q)}{Volume(G_i)}$$

(2) 誤差指数

表 3 は範囲クエリとその周辺グリッドの一例を表している．G_{11} と G_{12} はクエリ Q_1 の周辺グリッドであり，G_{21} と G_{22} はクエリ Q_2 の周辺グリッドである．このとき，G_{12} の誤差率は非常に高いが，クエリ Q_1 全体としてみれば，非常に小さな誤差である．一方で，G_{22} の誤差及び誤差率は G_{12} と同じであるものの，G_{22} の誤差はクエリ Q_2 にとっては大きな誤差であることがわかる．

このように，同じ誤差および誤差率だとしても，その周辺グリッドがクエリ全体の誤差に与える影響は異なるということを反映するために，以下の誤差指数と呼ぶ評価指標を定義する．

$$\text{誤差指数}=\frac{\text{グリッドの誤差}}{\text{クエリ全体の真値}}$$

この指数は，個々のグリッドの誤差が，クエリ全体の真値と比較してどれだけ悪影響を与えるかを意味する．クエリ全体に対する誤差が大きくなるほど，評価値は高くなる．

4.2 予備実験

3.3 節の各推定手法を用いて，総和及びデータ数 (カウント) を求める 4 次元の範囲クエリを実行し，オーバーラップ率と誤差指数との関係を検証するために予備実験を行った．

実験では以下のデータセットを用いた．我々は，2010 年 1 月 14 日から 2014 年 4 月 11 日までの間の室内に設置された気象センサのデータを 2,032,918 件（約 200 万件）収集した．各データは，時刻，温度，湿度，照度，風速などの 16 個の属性を持つ．このデータのうち 2011 年から 2013

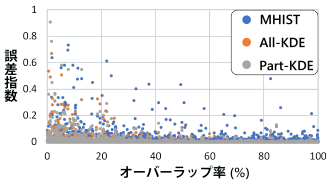

図 5　プロット (総和)

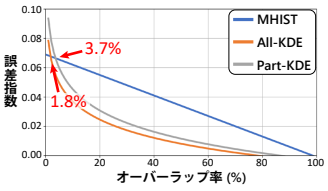

図 6　傾向線 (総和)

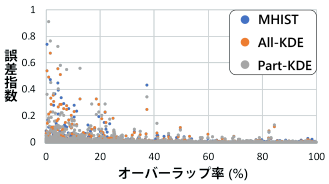

図 7　プロット (カウント)

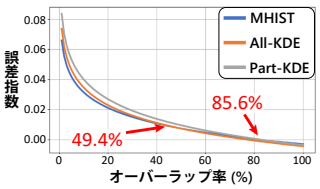

図 8　傾向線 (カウント)

年までの 3 年分のデータを抽出し，時刻を 3 年ずつ遅らせながら 7 倍に複製することで，2011 年から 2031 年までの擬似的な約 1,000 万件のデータ（以後，この疑似データを室内気象センサデータと呼ぶ）を生成した．この室内気象センサデータをもとに 9 個のコピーを作成し，合計約 1 億件のデータとして実験に使用した．

実験環境として，Intel Core i7-3770 CPU（3.4GHz），メモリ 32GB，ハードディスク 2TB，及び CentOS 6.7 にて HBase 1.2.0 が動作する 13 台の PC クラスタを用いた．このとき，全ての PC が Region Server の役割を持つ．また，全ての PC に PostgreSQL 9.6.1 をインストールし，マルチスタンバイ構成のレプリケーションを構築した．

図 5 は総和の範囲クエリに関して，オーバーラップ率と各推定手法を用いたときの誤差指数の値をプロットしている．図 5 の結果をもとに，各推定手法のオーバーラップ率と誤差指数の関係性を検証するために，各推定手法のプロットに対応する傾向線を引いた．図 6 は総和の範囲クエリの各推定手法の傾向線を表しており，これより，誤差指数の値の大小関係が入れ替わる閾値が存在することが判明した．表 2 では，MHIST は最も精度が低い推定手法と考えられていたが，オーバーラップ率 1.8%以下では最も精度の高い推定手法であることがわかる．図 7 及び図 8 はカウントの範囲クエリに対して同様の処理を行った結果を表している．この結果から，カウントの範囲クエリに関しては，オーバーラップ率 49.4%以下にて MHIST は最も精度の高い手法となることがわかる．

4.3　推定手法の動的変更

前節の予備実験の結果をもとに，オーバーラップ率に基づいて適用する推定手法を変更する新しい推定手法を提案する．

室内気象センサデータについて，総和の範囲クエリに関しては，オーバーラップ率 1.8%以下では MHIST を適用し，それ以外の場合は All-KDE を適用する．同様に，カウントの範囲クエリに関しては，オーバーラップ率が 49.4%以下では MHIST を適用し，それ以外の場合は All-KDE を適用する．MHIST と All-KDE を組み合わせる（この手法を以降，*MA* と呼ぶ）ことにより，推定精度とクエリスループットの両方の向上が期待される．

さらに，Part-KDE は All-KDE よりも高いクエリスループットで All-KDE とほぼ同等の推定精度を実現できるので、MHIST と Part-KDE を組み合わせる（この手法を以降，*MP* と呼ぶ）ことも検討する．この場合，総和の範囲クエリでは，オーバーラップ率 3.7%以下では MHIST を適用し，それ以外の場合は Part-KDE を適用する．カウントの範囲クエリでは，オーバーラップ率が 85.6%以下では MHIST を適用し，それ以外の場合は Part-KDE を適用する．

5.　評価実験

二つの提案手法の性能を評価するため，我々の過去の手法 [17], [18] 及び，Watari らの手法 [15] と比較を行う．実験環境及び使用するデータセットは，4.2 節と同様である．

5.1　推定精度の評価実験

提案手法と我々の過去の手法の推定精度の比較を行った．いくつかの設定値は以下の通りとした．

- グリッドのデータ数の上限（グリッドサイズ）: 1000
- MHIST によるバケット分割の上限数: 25
- 属性値範囲の分割数（3.3 節における n）: 5

4 次元の総和クエリとカウントクエリを実行し，クエリは選択率が 0.001%，0.1%，10% になるようランダムに生成したものを用いた．各選択率について 50 個のクエリを実行し，各推定手法の誤差率をクエリ全体と周辺グリッドのみについてそれぞれ計測した．

5.1.1　結果

図 9 及び図 10 は，各クエリの各選択率において計測された誤差率の平均値を表している．

図 9 は MA が全ての選択率において，我々の既存の手法より高い推定精度を達成していることを示している．特に周辺グリッドの選択率 0.001%に関しては，オーバーラップ率が低いグリッドが多数存在するため，顕著に精度が改善されていることがわかる．しかしながら，図 10 からは MA が必ずしも最高の精度になるとは限らないことがわかる．この原因としては，カウントクエリの傾向線の上下関係にあると考えられる (図 8)．カウントクエリの傾向線の上下関係は，総和クエリの場合 (図 6) ほど明確ではないために，オーバーラップ率 49.4%以下であっても MHIST が最良の推定手法にはならない場合が発生したと考えられる．ただし，カウントクエリにおいては，MA は必ずしも最良の推定手法ではないものの，クエリ全体として見れば軽微な推定精度の悪化に留まっている．

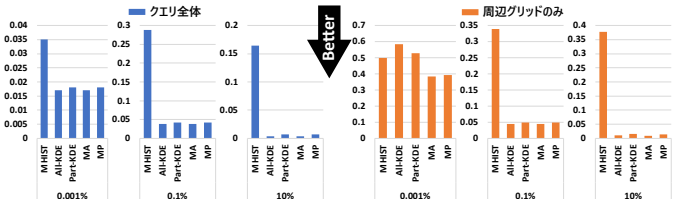

図 9　平均誤差率 (総和)

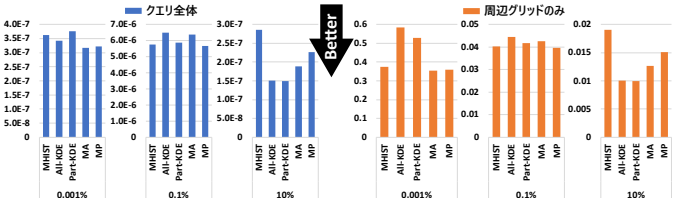

図 10　平均誤差率 (カウント)

一方で，MP に関しては，多くの場合において MA より推定精度が劣るものの，大きな精度の悪化にはなっていないことがわかる．

5.2　クエリスループットの評価実験

次に，提案手法と過去の手法および Watari らの手法について，クエリスループットの比較を行った．前節の実験と同様の条件でクエリを生成し，選択率ごとに 1，16，32，64 および 128 個のクライアントから同時にクエリを発行した．

5.2.1　結果

図 11 及び図 12 は各選択率における各手法のクエリスループットを表している．なお，縦軸はクエリスループット（クエリ/秒）を表し，横軸は同時実行クライアント数を表している．

図 11 は MA，MP 共に，部分的に MHIST を利用することで，それぞれが All-KDE，Part-KDE より高いクエリスループットを達成していることを示している．また，図 12 も同様に結果を示しているが，さらに，同時実行クライアント数が増えると MA は Part-KDE より高いクエリスループットを達成し，MP は MHIST とほぼ同等に高いクエリスループットを達成していることを示している．これは，MA はほぼ半分，MP は半分以上のオーバーラップ率にて，MHIST が適用されるからである．

以上二つの評価実験より，MA は推定精度を向上させながら，従来手法よりクエリスループットを達成でき，また，MP は少ない精度の悪化で MA よりも高いクエリスループットを実現できることを示した．

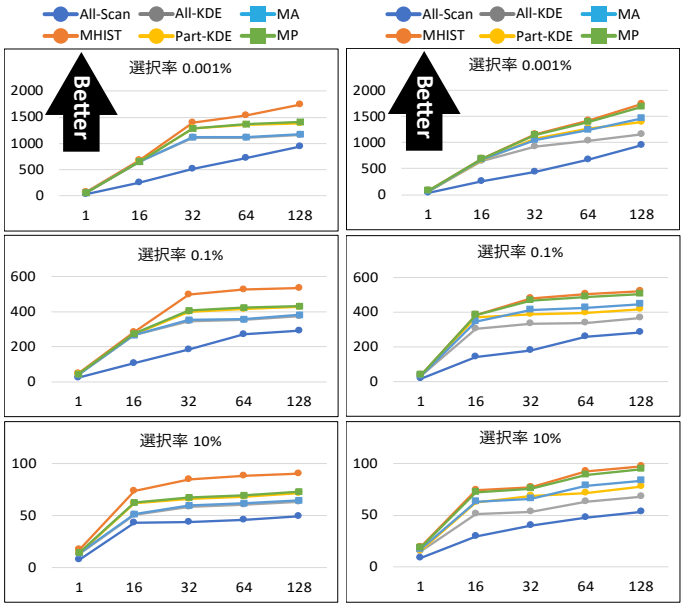

図 11　スループット (総和)

図 12　スループット (カウント)

6.　まとめ

本稿では，大規模多次元データに対する集約演算を効率的かつ高精度で結果を推定する手法を提案した．提案手法では，我々が過去に提案した三つの手法（MHIST，All-KDE，Part-KDE）[17], [18] から，オーバーラップ率に応じて最良の推定精度となる手法を動的に選択して推定に使用すること検討した．

評価実験では，提案手法を PostgreSQL と HBase を用いて実装し，推定精度とクエリスループットを評価した．その結果，オーバーラップ率に基づいて推定方法を動的に選択することにより，推定精度が改善される傾向があることを示した．また，MHIST を部分的に使用することによって，従来手法 [15], [17], [18] より高いクエリスループットを達成した．

今後の課題としては，推定手法の切り替えを判断するための誤差指数の大小関係が入れ替わる閾値を，任意のデータに対して自動的に見つける方法を検討することである．今回の提案手法では，特定のデータセットである室内気象センサデータに対応した閾値を調査した．提案手法の一般性を高めるために，任意のデータに対して閾値を発見する方法が必要である．

謝辞　本研究の一部は，JSPS 科研費 (18H03242, 18H03342，16H02908，17K12684)，JST ACT-I の助成を受けたものである．ここに記して謝意を表す．

参考文献

[1] Eldawy, A., Mokbel, M.F.: *SpatialHadoop: A MapReduce Framework for Spatial Data*, 2015 IEEE 31st International Conference on Data Engineering. pp. 1352–1363 (2015).

[2] Garcia-Molina, H., Ullman, J.D., Widom, J.: *Database Systems: The Complete Book*, Prentice Hall (2002).

[3] Ioannidis, Y.: *The History of Histograms (Abridged)*, Proceedings of the 29th International Conference on Very Large Data Bases. pp. 19–30 (2003).

[4] Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: *Optimal Histograms with Quality Guarantees*, Proceedings of the 24th In-

ternational Conference on Very Large Data Bases. pp. 275–286 (1998).
[5] Kooi, R.P.: *The Optimization of Queries in Relational Databases*, Ph.D. thesis, Case Western Reserve University (1980).
[6] Lakshman, A., Malik, P.: *Cassandra: A Decentralized Structured Storage System*, ACM SIGOPS Operating Systems Review, Volume 44 Issue 2. pp. 35–40 (2010).
[7] Muralikrishna, M., DeWitt, D.J.: *Equi-depth Multidimensional Histograms*, Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data. pp. 28–36 (1988).
[8] Nishimura, S., Das, S., Agrawal, D., El Abbadi, A.: *$\mathcal{MD}$-HBase: Design and Implementation of an Elastic Data Infrastructure for Cloud-scale Location Services*, Distributed and Parallel Databases, Volume 31 Issue 2. pp. 289-319 (2013).
[9] Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: *Efficient OLAP Operations in Spatial Data Warehouses*, Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases. pp. 443–459 (2001).
[10] Piatetsky-Shapiro, G., Connell, C.: *Accurate Estimation of the Number of Tuples Satisfying a Condition*, Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data. pp. 256–276 (1984).
[11] Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: *Improved Histograms for Selectivity Estimation of Range Predicates*, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. pp. 294–305 (1996).
[12] Poosala, V., Ioannidis, Y.E.: *Selectivity Estimation Without the Attribute Value Independence Assumption*, Proceedings of the 23rd International Conference on Very Large Data Bases. pp. 486–495 (1997).
[13] Silverman, B.W.: *Density Estimation for Statistics and Data Analysis*, CRC Press (1986).
[14] Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.C.: *Indexing Multi-dimensional Data in a Cloud System*, Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. pp. 591–602 (2010).
[15] Watari, Y., Keyaki, A., Miyazaki, J., Nakamura, M.: *Efficient Aggregation Query Processing for Large-Scale Multidimensional Data by Combining RDB and KVS*, Proceedings of the 29th International Conference on Database and Expert Systems Applications. pp. 134–149. (2018).
[16] Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: *An Efficient Multi-dimensional Index for Cloud Data Management*, Proceedings of the First International Workshop on Cloud Data Management. pp. 17–24 (2009).
[17] 張涵, 渡佑也, 欅惇志, 宮崎純.: ヒストグラムとカーネル密度推定を組み合わせた集約演算結果の推定, DEIM 2017 論文集, E5-1 (2017).
[18] 張涵, 渡佑也, 欅惇志, 宮崎純.: 分散環境における多次元データに対する集約演算結果の推定とその評価, DEIM 2018 論文集, I6-1 (2018).