大規模データに対する 信頼区間を保証した近似集約演算の高速化手法の提案

伊藤　翔† 前田 涼吾† 常　穹† 宮崎　純†

† 東京工業大学情報理工学院情報工学系　〒 152–8552 東京都目黒区大岡山 2–12–1
E-mail: †{ito,maeda}@lsc.c.titech.ac.jp, ††{q.chang,miyazaki}@c.titech.ac.jp

あらまし　本研究では，大規模データに対する近似集約演算のうち，信頼区間を設定した決定論的な手法である Deterministic Approximate Querying をベースとした高速な演算手法を提案する．演算するデータ量の削減を行うことで，本研究の目的である近似集約演算の誤差保証と高速化の両立を図る．提案手法について，生成データや実データを用いて実システムでの実行時間を比較と，より一般に近い状況である混合分布データの場合を検討する．

キーワード　近似集約演算，大規模データ処理

1 はじめに

近年，通信機器やセンサデバイスの普及により，企業や組織が多種多様で大量のデータ（以下，大規模多次元データ）を生成，収集している．これらのデータを活用することで企業や組織の意思決定プロセスを推進していくことが期待されているが，大規模多次元データから有用な情報を得るためにはデータの集約が必要である．データの集約結果を得るためには，データに対する集約演算が必要である．しかし，大規模データに対する集約演算を全データからそのまま計算すると計算コストが膨大にかかる問題点があり，実用性を鑑みると，集約結果の正確性よりも応答性や適時性が重要視されている．このような需要から，誤差を許容しつつ高速に大規模データを集約する演算手法として，近似集約演算が広く用いられている．

誤差を許容する近似集約演算の課題として誤差を明示できない点がある．整数型データを想定する場合に，この課題を解決する有効な手法として，Potti ら [1] が提案した Deterministic Approximate Querying（以下，DAQ）がある．DAQ は，Bit-sliced Indexing [2] という整数型のデータ群を 2 進数の桁ごとに分割し扱うスキーマを用いて，近似集約結果と真値との誤差を保証できる区間の提示する手法である．この手法は計算するビットスライスが多いほど演算にかかるコストが大きくなるので，計算すべきビットスライスを削減することによって計算速度の向上を図ることができる [3].

そこで本研究では DAQ を基盤手法にし，誤差範囲を保証しつつより高速に近似集約演算が収束する手法の提案を目的とする．本研究の要点は DAQ で演算するビットスライスを削減することで，本論文ではこの課題を解決する手法を提案している．提案手法である PBE(Patched Based Encoding)+DAQ は，PBE をデータ群に適用したのちに DAQ を適用する手法である．PBE はデータ群の最小値をベース値と定め，全データからベース値を減算したものを新たなデータとして保持する手法である．この手法を適用することで，ベース値から近い値はデータを表現するために必要な桁が減るため，より DAQ を効果的に適用できることが期待できる．

以下に本論文の構成を示す．第 2 節では提案手法のベースとなる近似集約演算に関する既存研究および，前提となる基礎知識について述べる．第 3 節では本研究で提案する手法を述べ，第 4 節では提案手法の性能を明らかにするために行った評価実験について述べる．第 5 節で本論文の結論を述べる．

2 関連研究

2.1 近似集約演算

近似集約演算は，大規模データに対して集約演算を行うときに高速に集約結果を求めるために用いられる手法である．代表的な近似集約演算としてサンプリング手法がある [4]．全データからランダムサンプリングや層別サンプリングを行ったデータに対して集約演算を施すことで近似結果を算出するものである．サンプリング手法においては，データの分布に大きな偏りがある場合，分布の小さいデータから選択されなくなることでサンプリング後のデータ分布が大きく変わってしまう問題や，MAX や Top-k などの集約演算では正しく演算することができないといった問題があり，データによっては効果的でない場合がある．より精度の高い近似集約演算を行うために様々な研究がなされており，その代表的な手法として，ヒストグラム [5] [6] [7] やカーネル密度推定 [8]，ウェーブレット変換 [9] などを応用したものが挙げられる．これらに代表される手法は高速かつ高精度の近似集約演算を目的としているが，演算の真の解と近似解の誤差を明示できないという課題がある．

2.2 Deterministic Approximate Querying

Potti ら [1] は，近似集約結果と真値との誤差を保証できる区間の提示ができる，整数型データに対して有効な近似集約演算である Deterministic Approximate Querying を提案している．この手法は，Bit-slice indexing [2], [10] と呼ばれる集約演算を行う整数型のカラムデータに 2 進数の桁ごとに区切る処理

を施し，演算結果に対する影響の大きい上位桁から計算することで集約演算の解の高速な収束を実現している．未計算の下位桁に対しては，各ビットが全て 0 の場合と全て 1 の場合を仮定し，それぞれを下限値，上限値とみなすことで誤差の保証ができる区間の提示を可能にしている．DAQ のアルゴリズムで合計値を求めるクエリ処理を行う場合の処理の流れについて説明する．

Potti らによる評価実験 [1] では，DAQ とサンプリング手法による近似集約演算（以下，SAQ）との比較，データ分布の違いによる DAQ の性能の評価行っていた．実験結果によると，一様分布のデータ群に対して平均値が誤差 0.001% 以下になるまで計算するクエリが与えられた場合に DAQ と SAQ を適用すると，DAQ よりも SAQ の方がおよそ 13 倍の速さで求めることができた．データ分布を変更し，Zipf 分布に従うデータに対して Top100 件のデータを取り出すクエリを想定した場合の実験では，SAQ が全データの 90% をサンプリングしても誤差が 1% 以下にならなかった．一方で，DAQ では上位 4 桁分のビットスライスを計算するだけで演算結果の誤差が 1% 以下となった．これは Zipf 分布のような，平均値や度数の高い値よりも大きい値が多く含まれているデータに対して DAQ が特に有効で有ることを示している．

2.3 Patched Base Encoding

ApacheORC における PBE の構造 [1]を説明する．PBE はビット幅が大きく変化する整数シーケンスに使用される．シーケンスの最小符号付き整数値をベース値と置き，全データから減算される．これらの減算済みデータの各値の表現に必要なビット数を計算し，全データのうち 90% を表現できるビット数 W を求める．続いて，$2^W - 1$ より大きい外れ値を持つ 10% のデータはパッチリストのパッチを用いて，外れ値を表現するために必要な追加のビットを設定する．パッチは，前のパッチからスキップされた要素数を示すギャップと追加ビットによって構成されるリストとしてエンコードされる．

3 提案手法

3.1 PBE+DAQ

DAQ を基盤に PBE を適用することで DAQ の高速化を図る手法である PBE+DAQ を提案する．また，混合分布に対して，より有効に PBE+DAQ を適用する手法を提案する．

3.1.1 PBE の利点と問題点

PBE を適用させたデータを用いて DAQ の演算を行うことで，計算効率の向上に影響を与える 2 つの利点が見込まれる．1 つは，PBE 適用後にベース値と再計算されたデータ群を用いて DAQ の演算を行う過程で，はじめにベース値と圧縮済みのデータ数を掛け合わせることにより，演算過程で求める推定値が底上げされ，演算結果が真値により早く近づくことが可能という点である．この特性によって，より少ない演算によって高速に推定値を求めることができる．もう 1 つは，PBE でのデータ圧縮時に圧縮後のビット数を決定するので，DAQ で計算すべきビット数があらかじめ把握することが可能という点である．集約演算結果を求めるために必要となる最小限の桁を把握することで，データ型の最上位桁から計算する必要がなくなる．

1：https://orc.apache.org/specification/ORCv1/

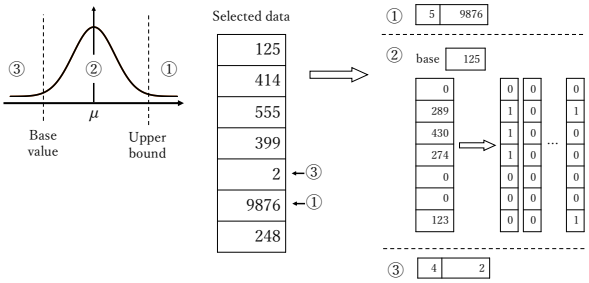

図 1　PBE+DAQ のアルゴリズムの概要.

2.3 節で説明した単純な PBE では，ベース値はデータ群の最小値と定め，データ群のうち下位 90% のデータを圧縮して上位 10% を外れ値とみなして扱っていた．この手法をそのまま DAQ に適用すると，2 つの問題が顕在化する．1 つはより小さい外れ値についての処理を加味できていないという問題で，もう 1 つは外れ値とみなされる上位 10% のデータが多すぎるという問題である．

1 つ目の小さい外れ値について，データ分布から逸脱した小さい値が存在する場合にその値がベース値と設定されるため，PBE を適用後のデータの表現に必要な桁数が変わらずに意味のない処理になってしまう可能性がある．また，新たなデータの追加が行われたときに追加されたデータの値があるカラムで最小値だった場合，追加されたデータの最小値を元に新たに PBE を適用してデータを再計算する必要があり，更新に関する計算コストが頻繁に生じるという懸念がある．

2 つ目の外れ値とみなされる上位 10% のデータが多すぎる問題について，例えばデータ群の上位が多く選択されるようなクエリが与えられたとき，外れ値内のデータを走査・計算する頻度が高くなり，結果として多くの処理時間を要するという懸念がある．

3.1.2 PBE+DAQ のアルゴリズム

3.1.1 節で述べたように，PBE の検討すべき課題として小さい外れ値の扱いと，外れ値とみなすデータの割合がある．両者の課題を解決するための方法として，本手法では図 1 のように，始めに PBE のベース値と上限値を設定する手法を採用する．

この手法によって定められた境界によってデータ分布が 3 つの領域に分割される．②で示されている範囲にあるデータに対しては，従来通りの PBE を適用することでデータの圧縮を行う．①と③で示されている外れ値に対しては生データのまま計算を行う．このようにデータを扱うことで，PBE が効果的に働く範囲に対してのみ処理を行い，PBE の効果を下げる外れ値は圧縮対象から除外することができると考えられる．図 1 では例として正規分布に従うデータで示しているが，他の分布に

従うデータにおいても同様にベース値と上界値を適切に定めることで，本手法を適用することができる．

分割された 3 つの領域を区別して近似集約演算を行う流れを説明する．結論から述べると，① → ② → ③ の順に演算を行う．

まず，①のデータに対して，集約演算を行う．集約した結果が要求精度の範囲内であればここで計算を終了し，そうでなければ続けて演算を行う．次に②の PBE を適用したデータに対してビットスライスを行ったものを用いて集約演算を行うが，始めにベース値を処理する．例えば合計値を求める集 約演算を行う場合には，PBE を適用したデータとベース値をかけ合わせたものを，①で合計を求めた 変数に足し合わせる．この時点で要求精度を満たしていれば終了する．そうでなければ，PBE を適用したデータのビットスライスを用いて節 2.2 と同様の処理を行う．全てのビットスライスを計算しても誤差要求を満たせない場合は，最後に③のデータに対して集約演算を行い，終了する．

3.1.3 PBE+DAQ wide-narrow のアルゴリズム

PBE+DAQ は，特に対象のデータが一つの正規分布からなり，その分散が小さい場合に有効であることが想定できる．これは，平均に近い値を持つデータの割合が大きい分布において，PBE によるエンコードの圧縮率が高くなり，ナイーブな DAQ よりも演算に必要となるデータの読み込み数が減るという理由から説明できる．

ここで，さらなる課題として，より一般の状況を考慮することを考え，一般の状況として混合分布データを想定する．PBE+DAQ の応用手法では，分布を構成する大部分のデータをエンコードする際に，広い範囲の PBE 対象と狭い範囲の PBE 対象に分けてデータをエンコードする．これは，狭い範囲の PBE によって高圧縮可能な分布データと，その範囲に含まれないデータ群に対して別の広い範囲の PBE を施すことで，集約演算結果の信頼区間の収束に必要となるデータをより減らすことを目的としている．あらかじめ，混合分布を構成する分布のパラメータである分散や重みに基づいて，単純な PBE+DAQ よりも高速に特定の信頼区間への収束を達成可能になるかを判断し，より適したエンコード手法を選択することを目標とする．そのため，応用手法として PBE+DAQ wide-narrow を提案する．この手法は，PBE を適用してビットスライス処理を施す範囲として wide と narrow の 2 つを定義することを要点とする．

一般的な分布として混合分布を想定すると，混合分布データは，構成する各分布の重みや平均などのパラメータにより様々な形をとり，尖度が低い分布が含まれることが考えらえる．シンプルな PBE + DAQ のエンコード手法は，標準偏差が小さく尖度が高い分布データに対しては強い圧縮効果を持つが，逆に，標準偏差が大きく尖度が低い分布データに対しては圧縮効果を期待できない．混合分布データが後者の分布データを含む場合，データ全体に PBE + DAQ で外れ値の割合を抑えたエンコードを施すと，エンコード後のデータを表すビット数が多くなる．それにより，本来であればより少ないビット数で表現可能な分布のデータ群に対しても，圧縮率の低いエンコードを施すことになってしまい，演算時に無駄なコストが生じると考えられる．PBE+DAQ wide-narrow は，このような状況の解消をねらいとした手法である．

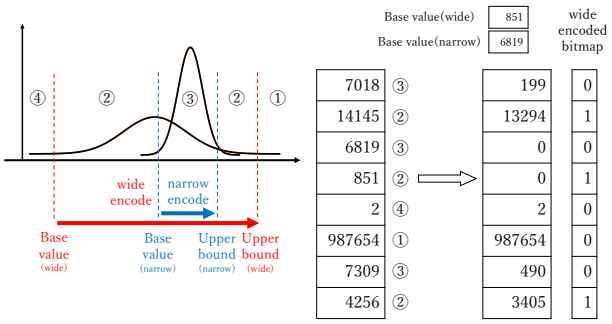

図 2 PBE+DAQ wide-narrow によるエンコードの例

表 1 PBE+DAQ wide-narrow のエンコード領域の説明

領域	説明
①	パッチリストに格納された，大きい値を持つ外れ値のデータ
②	wide-bitsliced の対象データ
③	narrow-bitsliced の対象データ
④	パッチリストに格納された，小さい値を持つ外れ値のデータ

本手法で PBE を適用してビットスライス処理を施す範囲として定義される wide と narrow について，実際のエンコードの例を図 2 で示す．図 2 中の①，②，③，④のそれぞれの領域のデータを表 1 を用いて説明する．

narrow のエンコード範囲は，混合分布を構成する分布の中で，主に尖度が高く重みの大きい分布データの存在する範囲を設定する．その範囲内の値のデータに対し，範囲の下限値をベース値とした PBE 処理とビットスライス処理を施す．これを narrow-bitsliced データとする．wide のエンコード範囲は，混合分布全体を一つの分布として捉え，従来の PBE + DAQ と同様の方法でエンコードとビットスライス処理を行う．これを wide-bitsliced データとする．この際，narrow のエンコード範囲に含まれるエンコード済のデータは，処理の対象としない．wide-bitsliced を表現するビット数を W_w，narrow-bitsliced を表現するビット数を W_n としたとき，wide のエンコード範囲は，narrow のエンコード範囲より広いことから，$W_w > W_n$ の関係を保証できる．wide のエンコード範囲に含まれない値を持つデータについても，従来の PBE + DAQ と同様の方法でパッチリストに保持する．また，ベース値の異なる PBE エンコードでデータを圧縮した場合，データがどのベース値でエンコードされたかを特定できず，もとの値の復元ができない．そのため，データがどちらのエンコード対象となったかをビットで表したビットマップデータを作成し，保存する．

前処理によって，データの読み込みを効率化するため，wide-bitsliced の対象となるデータのうち，高位のビットスライスだけを，wide-bitsliced データとして保存し，下位のビットスライスは narrow-bitsliced データに保存する．wide-bitslice と narrow-bitsliced のエンコードの対象となるデータは，それぞれ W_w 桁と W_n 桁のビットで表現できるようエンコードされ

る．wide-bitslice のデータのビットスライスは W_w 桁で構成されるが，そのうち，下位の W_n 桁のビットスライスについては，narrow-bitsliced と同一のビットスライスに値を保存する．上位 $W_w - W_n$ 桁のビットスライスは，元のインデックス値を考慮せず，wide-bitsliced の対象となるデータが持つ値のみを列方向に圧縮して保存する．集計時には，値の大きいパッチリストの合計値を集計したのちに，大きい値を表現する wide-bitslice の上位のビットスライスのみを読み込み集計する．以上の構成により，ビットスライスの集計時に効率的に大きい値を集計し，集約演算結果の解が存在する保証区間の高速な収束を実現する．

合計値や平均値を求める近似集約演算を行う流れを，図 2 を用いて説明する．近似集約演算で解の範囲を縮小する過程で，集約した結果が要求精度の範囲内であればその時点で計算を終了する．具体的には，初めに①のデータに対して集約演算を行う．続いて，②と③に属するデータはそれぞれのエンコードで用いられたベース値以上の値を持っていることを利用し，解の存在範囲を縮小させる．例えば，合計値を求める集約演算を行う場合には，②に属する wide-bitsliced のデータ数と wide エンコードのベース値を乗算した値と，③に属する narrow-bitsliced のデータ数と narrow エンコードのベース値を乗算した値を①に対して合計を求めた結果に足し合わせる．この時点で要求精度を満たしていれば演算を終了する．次に，②のビットスライスデータのうち，大きい値を持つ一部のビットスライスに対して 2.2 節と同様の演算処理を行う．ここで，大きい値を持つ一部のビットスライスとは，②が持ち③が持たない $W_w - W_n$ 桁のビットスライスのデータを指す．続いて，②の下位 W_n 桁と③のビットスライスデータに対して，DAQ に倣った集約演算を施す．全てのビットスライスを計算しても誤差要求を満たせない場合は，最後に④のデータに対して集約演算を行い，演算を終了する．

3.2 各手法のモデル化

提案手法が既存の DAQ よりも効率的に近似集約演算の解を求められるか検証するために，各手法をモデル化する．また，データの分布と要求される誤差に基づいて，クエリに効率的に解答可能となる手法を検討する．

DAQ を基盤とした手法では，各ビットスライスのビット演算よりも I/O コストの方が大きく，読み込むデータサイズが実行時間に比例する．これを踏まえ，要求精度を満たすために読み込む必要があるデータサイズを比較する観点から，

- DAQ ,
- PBE+DAQ ,
- PBE+DAQ wide-narrow

の 3 手法をモデル化する．

このモデル化では，

- 整数値データ群の合計値を求める集約演算と，
- 別のカラムの選択属性を持つ選択クエリであること，
- 選択クエリが一様ランダムにデータを選択する仮定と，
- 選択属性ビットマップ S の全てを同時にメモリに載せることできないこと

を仮定する．特に，大規模なデータに対する演算を想定すると，選択属性ビットマップ S はファイルサイズが非常に大きくなる恐れがあるため，モデル化する手法全てで，逐次アクセスして集計に用いる．

表 2　モデル化に用いた共通する変数または関数

変数または関数	説明
N	データの総数
s	選択クエリによるデータの選択率
$N_{selected}$	N 個のデータのうち，選択クエリによって選択されたデータ数
r	要求精度
B	ビットスライスデータ $B = (B_{n-1}, B_{n-2}, \cdots, B_0)$
E_i	ビットスライスデータ B_i
MAX	データの最大値
α_p	PBE+DAQ のエンコードのベース値
β_p	PBE+DAQ のエンコードの上限値
a_p	α_p より小さい値のデータの割合
b_p	β_p より大きい値のデータの割合
α_{pw}	wide-bitsliced のエンコードのベース値
β_{pw}	wide-bitsliced のエンコードの上限値
a_{pw}	α_{pw} より小さい値のデータの割合
b_{pw}	β_{pw} より大きい値のデータの割合
c_{pw}	wide-bitsliced でエンコード処理されるデータの割合
α_{pn}	narrow-bitsliced のエンコードのベース値
β_{pn}	narrow-bitsliced のエンコードの上限値
a_{pn}	α_{pn} より小さい値のデータの割合
b_{pn}	β_{pn} より大きい値のデータの割合
c_{pn}	narrow-bitsliced でエンコード処理されるデータの割合
ϵ	近似解と真値の誤差率の最大値
LB	近似解である保証区間の下限値
UB	近似解である保証区間の上限値
CV	保証区間の中央の値
$ceil(x)$	x 以上の最も小さい整数値を返す関数
$Count1(x)$	ビットスライス x 中の 1 の値の数を返す関数
t_{naive}	DAQ で読み込む必要のある最小のビットスライス桁数
a_{naive}	DAQ で読み込む必要のあるデータの推定値（バイト）
t_p	PBE+DAQ で読み込む必要のある最小のビットスライス桁数
a_p	PBE+DAQ で読み込む必要のあるデータの推定値（バイト）
t_{pwn}	PBE+DAQ wide-narrow で読み込む必要のある最小のビットスライス桁数
a_{pwn}	PBE+DAQ wide-narrow で読み込む必要のあるデータの推定値（バイト）

また，データの分布と要求される誤差に基づいて，クエリに効率的に解答可能となる手法を検討する．そのためには，近似集約演算前に，各手法が要求精度を満たすために読み込む必要があるデータサイズを事前に評価する必要がある．評価は，各手法において特定のビットスライスまでの集計を終えた時点での誤差率を推定して行う．誤差率の推定には，近似解の下限値 LB および上限値 UB との差の値が必要となる．この 2 つの値のうち，後者は事前に計算可能な値であるのに対して，前者は未知であるため，推定値を用いる．下限値 LB の推定は，

- 計算済みの各ビットスライス B_i の $Count1(B_i)$ と，
- 選択率 s

を用いて計算する．

各手法のモデル化において，使用した変数を表 2 に示す．

3.2.1 DAQ のモデル化

DAQ によって要求精度を満たすために読み込む必要があるデータサイズを説明する．DAQ は前処理によって，m 桁のビットスライスデータに区切る．ここで m は，

$$m = ceil(log_2(MAX)) \tag{1}$$

で与えられる．

合計値を求める近似集約演算で，ビットスライス B_i まで集計した状況を考える．このときの誤差率 ϵ は，

$$\epsilon = \frac{(UB - CV)}{CV} = \frac{(UB - LB)}{UB + LB} \tag{2}$$

と表される．未集計のビットスライスのとりうる値の上限値と下限値の差が，そのまま UB と LB の差となるため，事前の評価では LB の推定値 LB' を推定することで ϵ の推定値を得る．

UB と LB の差は, [1] より合計値を求める場合には,

$$UB - LB = N_{selected}(2^i - 1) \quad (3)$$

で求められる．LB' は，集計済みのビットスライスの選択データの総和から計算される値であるため，集計済みのビットスライスの総和の推定値を用いて計算する．ビットスライス B_j の値の和 E_j は

$$E_j = 2^j \cdot Count1(B_j) \quad (4)$$

と 計算できることから，LB' は

$$LB' = s \sum_{j=i+1}^{m-1} (E_j) \quad (5)$$

と推定できる．

以上から，t_{naive} を DAQ が要求精度 r を満たすのに必要となるビットスライスの桁数とすると，t_{naive} は

$$(1-r) > \epsilon \approx \frac{2^{m-t_{naive}} - 1}{2 \cdot \sum_{j=m-t_{naive}}^{m-1} (\frac{E_j}{N}) + (2^{m-t_{naive}} - 1)} \quad (6)$$

を満たす最小の整数となる．

DAQ は 1 桁のビットスライスデータを集計する際に，ビットスライスデータと選択クエリビットマップの情報を読み込む必要がある．このことから，DAQ によって要求精度を満たすために読み込む必要があるデータサイズの推定値 A_{naive} は t_{naive} を用いて

$$A_{naive} = \frac{Nt_{naive}}{4} (\text{バイト}) \quad (7)$$

と評価できる．

3.2.2 PBE+DAQ のモデル化

PBE+DAQ によって要求精度を満たすために読み込む必要があるデータサイズを説明する．PBE+DAQ は前処理によって，l 桁のビットスライスデータに区切る．ここで l は,

$$l = ceil(log_2(\beta_p - \alpha_p)) \quad (8)$$

で与えられる．

合計値を求める近似集約演算で，ビットスライス B_i まで集計した状況を考える．このときの誤差率 ϵ は，式 (2) と同様に表される．事前の評価では選択クエリが一様ランダムにデータを選択する仮定を用いて，ϵ の推定値を得る．UB と LB の差は，未集計のビットスライスとベース値未満の値を持つデータがとりうる値の上限値と下限値の差となるため，これは

$$UB - LB \approx sN((1 - a_p - b_p)(2^i - 1) + a_p\alpha_p) \quad (9)$$

より求められる．

また LB は集計済みのビットスライスのとベース値以上の値を持つデータで表されるため，推定値 LB' を

$$LB' = s(\sum_{j=i+1}^{l-1} (E_j) + b_p N \cdot \beta_p) \quad (10)$$

より得る．

以上から，t_p を PBE+DAQ が要求精度 r を満たすのに必要となるビットスライスの桁数とすると，t_p は

$$(1-r) > \epsilon \approx \frac{(1 - a_p - b_p)(2^{l-t_p} - 1) + a_p\alpha_p}{2(\sum_{j=l-t_p}^{l-1} (\frac{E_j}{N}) + b_p\beta_p) + (1 - a_p - b_p)(2^{l-t_p} - 1) + a_p\alpha_p} \quad (11)$$

を満たす最小の整数となる．

PBE+DAQ は 1 桁のビットスライスデータを集計する際に，ビットスライスデータと選択クエリビットマップの情報を読み込む必要がある．また，ビットスライスの集計の前に，上限値以上の値を持つデータの総和とベース値未満の値を持つデータの数を取得する必要がある．これらから，PBE+DAQ によって要求精度を満たすために読み込む必要があるデータサイズの推定値 A_p は t_p を用いて

$$A_p = \frac{N(t_p + (a_p + b_p)k)}{4} (\text{バイト}) \quad (12)$$

と評価できる．

3.2.3 PBE+DAQ wide-narrow のモデル化

BE+DAQ wide-narrow によって要求精度を満たすために読み込む必要があるデータサイズを説明する．

この手法では，前処理によって wide-bitsliced の対象と narrow-bitsliced の対象に，異なるベース値の PBE を施す．それぞれ対象となるデータの割合は c_{pw} と c_{pn} は,

$$c_{pw} = 1 - (a_{pw} + b_{pw}) - c_{pn} = a_{pn} + b_{pn} - (a_{pw} + b_{pw}) \quad (13)$$

$$c_{pn} = 1 - a_{pn} - b_{pn} \quad (14)$$

となる．

wide-bitslice と narrow-bitsliced のエンコードは，それぞれ l_w 桁と l_n 桁のビットで表現できるようエンコードされる．ここで l_w と l_n は,

$$l_w = ceil(log_2(\beta_{pw} - \alpha_{pw})) \quad (15)$$

$$l_n = ceil(log_2(\beta_{pn} - \alpha_{pn})) \quad (16)$$

で与えられる．

この手順で合計値を求める近似集約演算で，ビットスライス B_i まで集計した状況を考える．このときの誤差率 ϵ は，式 (2) と同様に表される．事前の評価では選択クエリが一様ランダムにデータを選択する仮定を用いて，ϵ の推定値を得る．UB と LB の差は，未集計のビットスライスとベース値未満の値を持つデータがとりうる値の上限値と下限値の差となるため，これは

$$UB - LB \approx \begin{cases} sN(c_{pw}(2^i - 1) + c_{pn}(\beta_{pn} - \alpha_{pn}) + a_{pw}\alpha_{pw}) & i > l_n \\ sN((1 - a_{pw} - b_{pw})(2^i - 1) + a_{pw}\alpha_{pw}) & \text{otherwise} \end{cases} \quad (17)$$

より求められる．

また LB は集計済みのビットスライスのとベース値以上の値を持つデータで表されるため，推定値 LB' を

$$LB' = s(\sum_{j=i+1}^{l_w-1} (E_j) + b_{pw}\beta_{pw}) \quad (18)$$

より得る．以上から，DAQ が要求精度 r を満たすのに必要となる最小の計算ビットスライスの桁数 t_{pwn} を求める．

PBE+DAQ wide-narrow は 1 桁のビットスライスデータを集計する際に，ビットスライスデータと選択クエリビットマップの情報を読み込む必要がある．また，ビットスライスの集計の前に，上限値以上の値を持つデータの総和とベース値未満の値を持つデータの数を取得する必要がある．加えて，wide-bitsliced の高位のビットのみのビットスライスに対応した選択クエリビットマップを作成する必要がある．これらから，PBE+DAQ wide-narrow によって要求精度を満たすために読み込む必要があるデータサイズの推定値 A_{pwn} は t_{pwn} を用いて

$$A_{pwn} = \begin{cases} \frac{N((t_{pn}c_{pw}+1)+(a_{pw}+b_{pw})k)}{4} (\text{バイト}) & t_{pwn} \leqq l_w - l_n \\ \frac{N((l_w-l_n)c_{pw}+(t_{pwn}-(l_w-l_n)+1)+(a_{pw}+b_{pw})k)}{4} (\text{バイト}) & \text{otherwise} \end{cases} \quad (19)$$

と評価できる．

4 評価実験

本節では，2 つの提案手法の評価を行うための実験と結果について示す．大規模な実データを用いて実験し，PBE+DAQ と応用手法の有効性について評価する．

4.1 PBE+DAQ の評価実験

4.1.1 アルゴリズムの実装

本節では PBE+DAQ を実システムを構築して実験するための実装に関する説明を行う．システムの概要を図 3 に示す．

ディスクには，前処理を施したデータが保存されている．保存されているデータは，上界値より大きい値を生データのまま

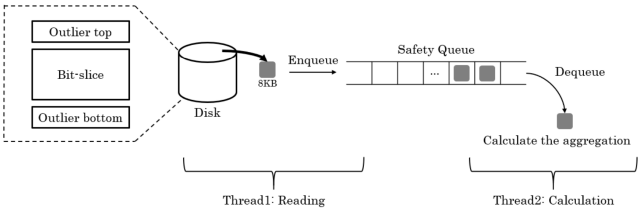

図 3　PBE+DAQ のシステム構築

表 3　大規模データの実験に使用したマシンの構成

CPU	製品	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
RAM	容量	32GB
HDD	製品	Seagate 社 ST1000DM003-1SB1
	容量	1TB
	回転数	7200RPM
	平均データ転送速度	156MB/sec
OS		Ubuntu 18.04

保存しているもの，ベース値と上界値の間のデータに対して PBE を適用後にビットスライスしたもの，ベース値より小さいデータを生データのまま保存しているものの 3 つに大別することができる．これらのデータを読み込み計算するが，読み込み処理と演算処理を別々のスレッドに割り当て，この 2 つ（もしくはそれ以上）のスレッドを並列に実行する．ディスクからの読み込み速度とメモリ上でのビット演算を並列化することで，ディスクからの読み込み速度の方が遅いので，集約演算全体の計算コストがディスクからの読み込み時間と等価になる．そのため，本実験の実行時間をデータの読み込み時間として計測することができるようになり，結果として近似集約演算に使われるデータ量の比較を行うことができる．また，ディスクから読み込むデータ量は適切なブロックサイズで設定する．図 3 では 1 つのブロックを 8KB で示している．

4.1.2　実 験 環 境

評価実験に用いた計算機の構成を表 3 に示す．大規模データを用いた実験システムで行った評価実験について，マルチスレッドプログラムのスレッド数は 2，スレッド間でデータの授受を行うキューのサイズは 10 とした．

4.1.3　実システムでの実験

本節では PBE+DAQ について行った評価実験について説明する．本実験では，実データとして HIGGS データ [11]，ビットコインの売買データ [12] を用いて実験し，PBE+DAQ の有効性について評価する．

HIGGS データは UCI Machine Learning Repository から提供されているデータセットで，カラム数 28，データ数 11,000,000 で構成されている．ここでは，特徴の異なる 2 つのカラムに対して実験を行った．データの詳細は節 a) で説明している．ビットコインの売買データは Coincheck から提供されているデータ数 154,048,253 のデータセットである．

a）　HIGGS データでの実験

HIGGS データについて，特徴の違う 2 つのカラムに対して実験を行った．HIGGS データを構成している 28 カラムのうち，6 列目と 26 列目のデータを用いている．各カラムのデータ分布を図 4，図 5 に示す．HIGGS データは浮動小数点型のデータなので，32bit 非負整数となるように線形変換を施している [3]．線形変換後のデータについて，6 列目のデータの平均値は 223,939,203，標準偏差は 94,260,720 であり，26 列目のデータの平均値は 114,630,078，標準偏差は 58,998,898 である．また，選択率は 0.1，要求精度は 0.999 で実験している．ベース値と上界値は両端 0.15%, 2.5%, 16% を外れ値とする 3 パターンで実験し，結果を比較している．

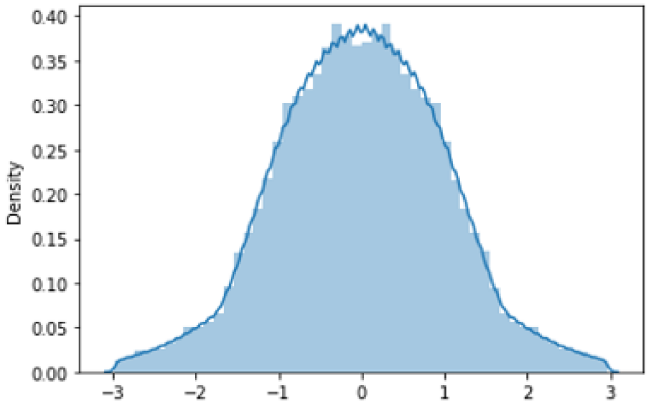

図 4　HIGGS データの 6 列目のデータ分布

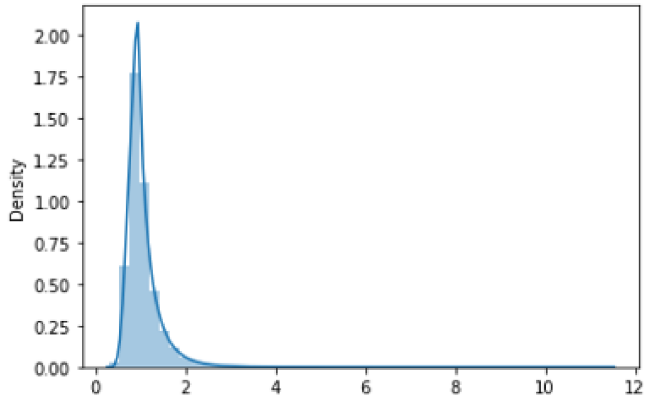

図 5　HIGGS データの 26 列目のデータ分布

図 6，図 7 は HIGGS データを用いてシステムを構築し行った実験の結果として，ベース値と 上界値を変化させたときの精度要求を満たすまでにかかった実行時間を示している．オレンジ色の線は DAQ の実行時間，青色の折れ線グラフはベース値と上界値の基準を変化させた場合の実行時間を示している。

b）　ビットコインの売買データでの実験

Coincheck が提供しているビットコインの売買データを用いて実験を行う．データ分布は図 8 の通りで，最小値は 135，最大値は 240,000 のデータである．ベース値と上界値は両端 0.15%, 2.5%, 16% を外れ値とする 3 パターンで実験し，結果を比較している．図 9 はビットコインの売買データを用いてシステムを構築し行った実験の結果として，ベース値と上界値を変化させたときの精度要求を満たすまでにかかった実行時間を示している．オレンジ色の線は DAQ の実行時間，オレンジ色の点線は最大値を保持することで必要最小限のビットスライス

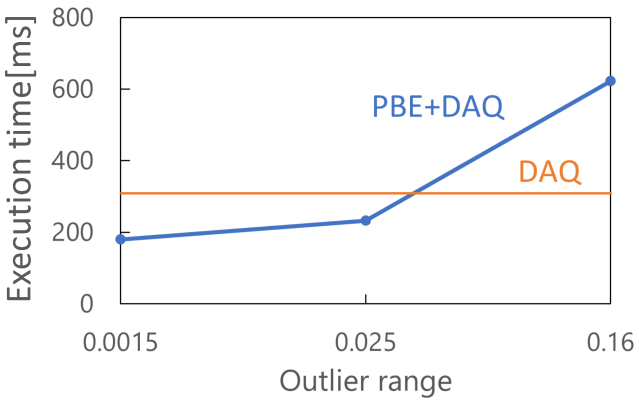

図 6　HIGGS データの 6 列目の実験結果

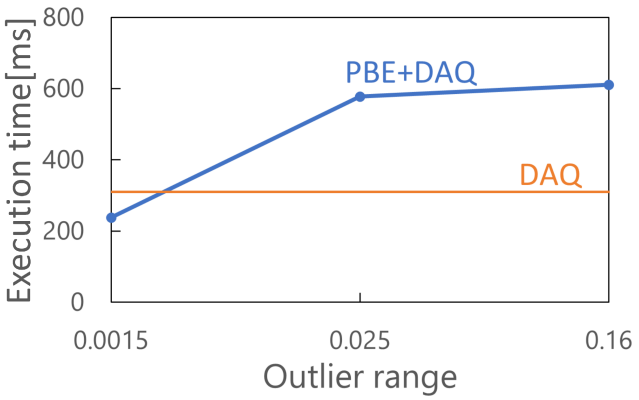

図 7　HIGGS データの 26 列目の実験結果

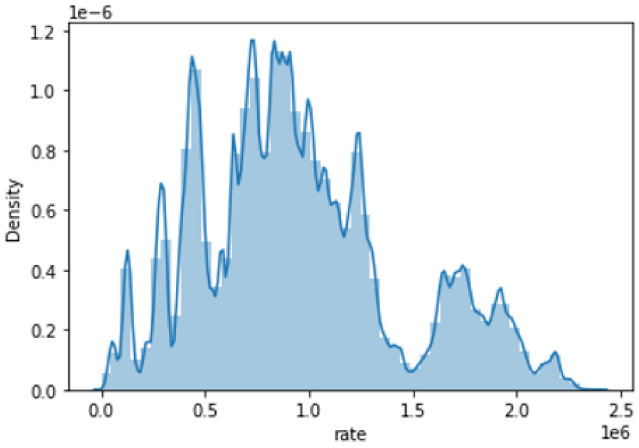

図 8　ビットコインの売買データの分布

を計算する DAQ の実行時間，青色の折れ線グラフはベース値と上界値の基準を変化させた場合の実行時間を示している．

図 8 から，図 9 のような複雑なデータ分布に対しても PBE+DAQ は機能しており，DAQ よりも計算時間を改善していることが確認できる．また，外れ値のデータ量と計算時間に比例関係があることが確認できる．外れ値とするデータの割合を変化させたときに，3 パターンの試行全てで DAQ の実行時間を上回っている理由として，データの最大値が小さいことが考えられる．DAQ ではデータ分布によらず 32bit 整数の場合最大 32 個のビットスライスを計算する が，1 が立っていない上位桁も無駄に計算してしまう．PBE+DAQ は事前に

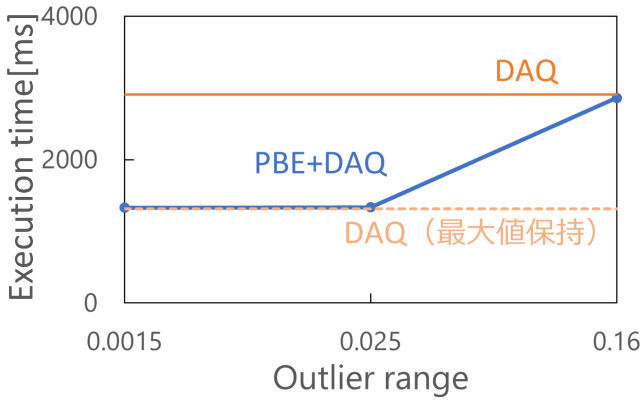

図 9　ビットコインの売買データの実験結果

ベース値と上界値を定めることで，その範囲内にあるデータに対してのみビットスライスを生成するので，余計な桁を計算する必要がなくなる．そのため，全体で計算するデータ量が大きく減少するので，外れ値の割合を 16% と大きくしても DAQ よりも計算時間が改善したと考えられる．

4.2　PBE+DAQ の応用手法の評価

最後に，PBE+DAQ の応用手法について，2 つの項目を検証することを目的とし，評価実験を行った．1 つめに，PBE+DAQ の応用手法が混合分布データに対してシンプルな PBE+DAQ よりも有効となるか検証すること．2 つめに，実システムが混合分布データに対して最適な手法を選択することが可能であるかを評価すること．これは，モデルによって推定されるデータ読み込み量と，実験の実行時間の結果を比較し，検証した．評価のためにビットコインの売買データでの実験を行った．データ分布は図 10 の通りである．最小値は 206,000，最大値は 7,071,867 の 70,772,733 件のデータを 32bit 非負整数として扱い，実験している．ベース値と上界値は両端 0.15%を外れ値とし，結果を比較している．

図 11 はビットコインの売買データを用いてシステムを構築し行った実験の結果として，各精度要求を満たすまでにかかった実行時間を縦軸に示している．結果として，PBE+DAQ wide-narrow の手法では，ベースとなる手法の DAQ や PBE+DAQ よりも 1.1 から 1.2 倍程度の高速化を実現した．要求精度を満たすまでに必要となった計算ビットスライス数は全ての手法で同一であったが，応用手法において，上位のビットスライスの計算時に，一部のデータに対する読み込みを削減する効果が実行時間の結果に表れたと考えられる．また，モデルを用いた実行時間の評価と実際の実行時間を表 4 と表 5 に表す．結果として，同一要求精度内で，読み込みデータ量の予測値と実行時間の実測値が概ね比例した関係になっているが，要求精度の異なる条件間では予測値と実行時間の比例関係にずれが生じた．このことから，同一のクエリ条件では，手法ごとの実行時間関係を概ね予測可能であり，最適な手法選択の目標を達成することが可能であると考えられる．しかし，要求精度などのクエリ条件が異なる場合，モデルを用いて実行時間の比例関係を評価できず，予測値と実測値が異なる結果になる例を確認した．

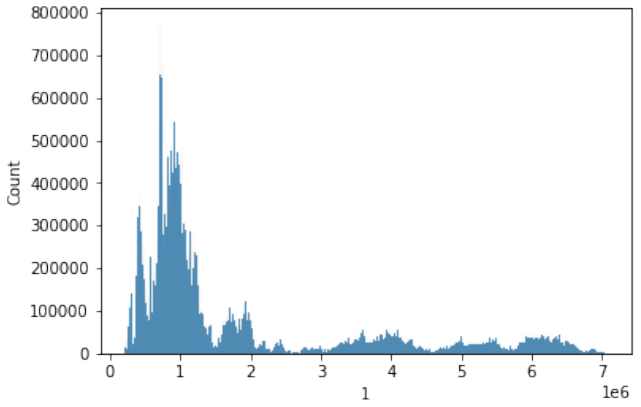

図 10　ビットコインの売買データの分布（応用手法評価）

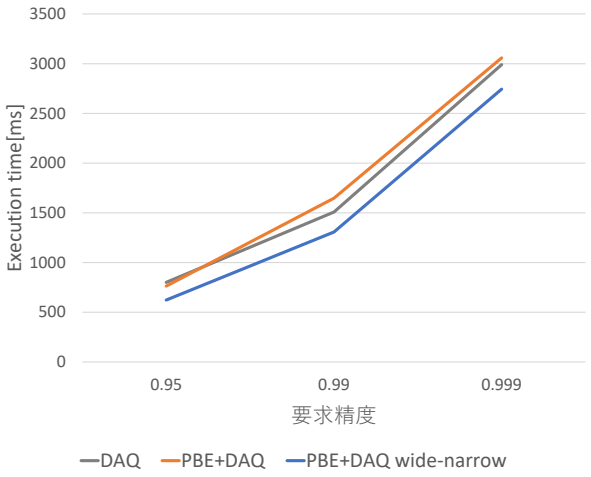

図 11　ビットコインの売買データの実験結果（応用手法評価）

表 4　実行時間の評価と実際の実行時間（要求精度 0.95）

	DAQ	PBE+DAQ	PBE+DAQ w-n
読み込みデータ量の予測値（MB）	106.2	106.5	90.3
計測実行時間（ms）	801	763	623

表 5　実行時間の評価と実際の実行時間（要求精度 0.999）

	DAQ	PBE+DAQ	PBE+DAQ w-n
読み込みデータ量の予測値（MB）	212.3	212.7	196.5
計測実行時間（ms）	2990	3058	2744

5　おわりに

本論文では，近似集約演算において誤差を保証しつつ高速に演算できる手法を 2 つ提案し，その性能評価を行った．本論文の 2 つの提案手法は，誤差保証できる近似集約演算手法である Deterministic Approximate Querying を基盤手法とし，演算するデータ量の削減を行うことで，本研究の目的である近似集約演算の誤差保証と高速化の両立を図った．

評価実験による検証では，実データを用いて DAQ と実行時間の比較を行い，PBE+DAQ がより高速に演算できることを示した．また，PBE+DAQ wide-narrow の評価実験では，混合分布を想定したデータセットにおいて，提案手法が既存手法の適用よりも効率的となる条件の存在を確認し，実際に実行時間を減らすようなシステムの構築が可能であることを確認した．ベースとなる手法の DAQ や PBE+DAQ よりも 1.1 から 1.2 倍程度の高速化を実現し，narrow-bitslice エンコードの対象となるデータの読み込みの削減が結果に表れたと考えられる．

今後の課題として，実行時間が早くなるような手法を自動的に選択するシステムの構築が挙げられる．自動的に効率的な手法を選択するシステムでは，事前の実行時間を評価を，任意の分布の場合や要求精度によって，各手法ごとに異なる前処理のメリットとデメリットを定量的に比較するだけでなく，実システムにおける実行時間をより多様なデータセットで検証する課題となる．

謝　　辞

本研究の一部は，JSPS 科研費 (18H03242, 18H03342, 19H01138) の助成を受けたものである．

文　　献

[1] Navneet Potti and Jignesh M. Patel. Daq: A new paradigm for approximate query processing. *Proc. VLDB Endow.*, Vol. 8, No. 9, p. 898–909, may 2015.

[2] Denis Rinfret, Patrick O'Neil, and Elizabeth O'Neil. Bit-sliced index arithmetic. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, p. 47–57, New York, NY, USA, 2001. Association for Computing Machinery.

[3] Xixian Han, Bailing Wang, Jianzhong Li, and Hong Gao. Efficiently processing deterministic approximate aggregation query on massive data. *Knowledge and Information Systems*, Vol. 57, pp. 437–473, 2017.

[4] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, p. 29–42, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Yannis Ioannidis. The history of histograms (abridged). In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, p. 19–30. VLDB Endowment, 2003.

[6] M. Muralikrishna and David J. DeWitt. Equi-depth multidimensional histograms. *SIGMOD Rec.*, Vol. 17, No. 3, p. 28–36, jun 1988.

[7] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. *SIGMOD Rec.*, Vol. 25, No. 2, p. 294–305, jun 1996.

[8] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[9] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. *The VLDB Journal*, Vol. 10, No. 2–3, p. 199–223, sep 2001.

[10] Patrick O'Neil and Dallan Quass. Improved query performance with variant indexes. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, p. 38–49, New York, NY, USA, 1997. Association for Computing Machinery.

[11] `https://archive.ics.uci.edu/ml/datasets/HIGGS/`. (2021 年 2 月 16 日アクセス).

[12] `https://api.bitcoincharts.com/v1/csv/`. (2021 年 2 月 16 日アクセス).