動的処理バケット選択方式による結合演算処理の詳細評価†

喜連川 優†† 中山雅哉††* 高木幹雄††

ハッシュ操作を用いた結合演算処理方式は，特に大容量のデータベースを扱う場合に有効であることが知られており，我々は，不均一なデータ分布に対しても高速にかつ柔軟に処理できる結合演算処理方式（動的処理バケット選択方式）の提案を行っている．この方式では，分割後の各バケットのデータ分布にあわせて動的に各々の処理手順を決定する方法を採っており，従来のハッシュ結合方式（ハイブリッドハッシュ結合方式）では処理効率が大幅に低下してしまう不均一なデータ分布に対しても高速に演算処理を施すことが可能となる．本論文では，動的処理バケット選択方式の処理アルゴリズムを詳細に解析して，分割バケット数の変化に伴う演算処理性能の評価を行い，ハイブリッドハッシュ結合方式の問題点について検討すると共に，不均一なデータ分布をとる場合の分割バケット数の決定方法についてまとめている．この結果，結合演算の処理性能は，あふれバケットの生成を防ぐ様に対象リレーションを分割することが重要であり，分割バケット数を多くとることによる処理オーバヘッドは，大規模なリレーションを扱う限りほとんど処理性能に影響を及ぼさないことが示されている．また，中規模のリレーションに対しては，あふれバケットの生成を防ぐことによる効果に加えて，動的処理バケット選択方式で用いている2種類のバケットまとめあげ処理（R_i バケットのまとめあげ処理，R_i バケットのまとめあげ処理）が有効であることを同様の処理性能評価から示している．前者は，分割バケット数が小さく分割処理にオーバラップして処理するバケットのステージング領域が大きい場合に効果があり，後者は，逆に分割バケット数が大きい場合に，フラグメントページに対する入出力コストを抑える効果がある．

1. はじめに

関係データベースシステムにおいて，結合演算が他の演算に比べて処理負荷が重いことは良く知られている[1),3),8)]．このため，ハッシュ操作を用いて対象リレーションを分割処理し，各分割空間（バケットと呼ぶ）ごとに結合演算処理を施すことで全体の処理コストを抑える各種のハッシュ結合演算処理方式―単純ハッシュ結合方式[4),5)]，GRACEハッシュ結合方式[4)~6)]，ハイブリッドハッシュ結合方式[4),5)]―が提案されてきた．

これらの中でもハイブリッドハッシュ結合方式は，単純ハッシュ結合方式とGRACEハッシュ結合方式の両利点をとり入れたアルゴリズムをとり，対象リレーションのサイズによらず常に最良の性能が得られる方式であることが文献4)，5)に示されている．しかしこれは，対象リレーションのデータ分布に基づいたハッシュ関数の選択を演算処理の事前に行うことで，分割後の各バケットのデータ分布が制御できると仮定しているためで，各バケットのデータ分布が不均一になる場合には，演算処理性能は著しく低下することになる[7),10),11)]．

一般に，ユーザから発行される問合せでは，各種の制約演算，射影演算を施した中間リレーションに対して結合演算処理を施すことが多く，動的に生成される中間リレーションのデータ分布を演算処理前に把握して結合属性に対してうまくデータを分散させることができるハッシュ関数を選択することは困難であると考えられる．そこで我々は，対象リレーションの分割処理を行いながら，各バケットのデータ分布状況を収集することで，動的に各バケットの処理順序を決定するハッシュ結合方式（動的処理バケット選択方式）の提案を行っている[10),11)]．文献7)に示すように，ハイブリッドハッシュ結合方式では分割バケットのデータ分布が不均一になる場合の処理効率低下が激しいのに対して，我々の提案する方式では，ほとんど性能に影響を及ぼさない．

これは，分割バケット数を多くとることで主記憶サイズを越えるバケット（あふれバケット）の生成を未然に防ぐことができるためであるが，分割するバケット数の変化に伴う処理性能の変化や，バケットのまとめあげ操作が処理性能に及ぼす影響等の詳細な評価については述べられていなかった．本論文では，動的処理バケット選択方式を詳細に解析し，分割バケット数の変化に対する性能評価を行いながら，分割バケット数の決定方法についてまとめている．

以下，2章では，ハイブリッドハッシュ結合方式に

† The Detailed Performance Evaluation of the Dynamic Hybrid GRACE Hash Join Method by MASARU KITSUREGAWA, MASAYA NAKAYAMA and MIKIO TAKAGI (Institute of Industrial Science, University of Tokyo).

†† 東京大学生産技術研究所
* 現在 豊橋技術科学大学知識情報工学系

おけるリレーション分割方法の問題点を明らかにし，その解決策として提案する動的処理バケット選択方式の概略について述べる．

3章では，バケット分布が不均一になる場合のハイブリッドハッシュ結合方式と動的処理バケット選択方式の性能比較を行う．ここでは特に，動的処理バケット選択方式における分割バケット数を変化させながら処理性能を測定することで，ハイブリッドハッシュ結合方式が性能低下を招く原因について考察している．

また4章では，動的処理バケット選択方式で採用している2種類のバケットまとめあげ処理（R_1 バケットまとめあげ，R_i バケットまとめあげ）の効果についてまとめている．この結果，前者は，小規模のリレーションを少ないバケット数で分割処理する場合の様に，対象リレーションサイズに対する R_1 バケットサイズの割合が高い場合に有効であり，後者は，小規模なリレーションを多くのバケット数で分割処理する場合の様に，各バケットにおけるフラグメントページ（保持データが1ページに満たないページ）の割合が高い場合に有効となることが明らかとなった．

最後に5章で，本論文により示された結果をまとめ，動的処理バケット選択方式における分割バケット数の決定方法について考察している．

2. 動的処理バケット選択方式の概説

2.1 本論文で用いる用語の定義と解説

結合演算処理を施す2つの対象リレーションをRおよびSとし，各リレーションの持つタプル数を $\{R\}$，$\{S\}$ で表現する（ここで，$\{R\} \leq \{S\}$ とする）．また，各リレーションはディスク上では固定サイズのページ単位で管理され，それぞれ $|R|$, $|S|$ ページから成るとする．

動的処理バケット選択方式では，GRACE ハッシュ結合方式や，ハイブリッドハッシュ結合方式と同様に，「スプリット関数」を用いて各リレーションを部分空間（バケット）に分割処理し，さらに各バケットごとに「ハッシュ関数」を用いた結合演算処理を行っている．この時，結合演算処理に利用できる主記憶空間は，各バケットのステージング用に $|M|$ ページ分とリレーションとの入力・出力用に各1ページずつの合計 $|M|+2$ ページ用意されているとする（図 1）．

リレーションRを分割処理した各バケットは R_i で表現し，各バケットのタプル数やページ数は，$\{R_i\}$，$|R_i|$ で表す．リレーションSの分割バケットに関しても同様に表現する．また，結合処理により得られる結果リレーションは RES で表し，各バケット R_i（および S_i）から生成される結果リレーションは，RES_i として表記する．

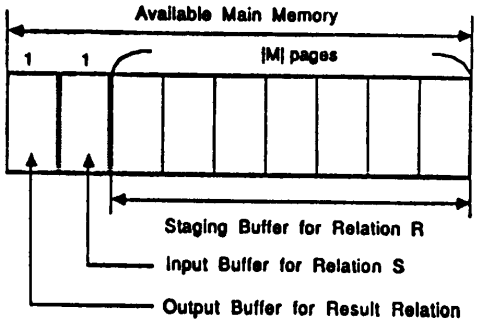


図 1 ハッシュ結合方式での主記憶の利用方法
Fig. 1 Memory allocation for join operations.

2.2 ハイブリッドハッシュ結合方式の問題点

ハイブリッドハッシュ結合方式は，ウィスコンシン大学の D. J. DeWitt らにより提案された結合演算処理方式で，ハッシュ結合方式の代表として良く知られている．この方式は，GRACE ハッシュ結合方式[4),5)]と同様に，対象リレーションの分割処理を行うフェーズ（分割フェーズ）と，各分割空間（バケット）の結合演算処理を行うフェーズ（結合フェーズ）の2フェーズから成るアルゴリズムをとりながら，GRACE ハッシュ結合方式の分割フェーズで無効となる主記憶領域を，最初に結合演算処理するバケット（R_1 バケットと呼ぶ）のステージング領域として利用することで，小規模なリレーションに対する結合演算処理の効率化を図ったものである．対象リレーションを処理に先だって分割する方式が大規模なリレーションに対して有効であることは，文献 2), 5), 6), 9) に示される通りであり，分割後のバケットサイズを制御できる場合には，最も効率良く演算処理を実行できる．

この結合方式は，換言すると，単純ハッシュ結合方式[4),5)]の R_1 バケットに対する結合処理時に，ディスクに書き戻される中間リレーションを分割しながら管理を行う方式と捉えることもでき，両方式の利点を有する方式としてハイブリッドハッシュ結合方式と名付けられた．

ハイブリッドハッシュ結合方式では，対象リレーションのデータ分布に見合ったスプリット関数を演算処理に先だって選択することで，次式に示すサイズの H_i 個のバケットに分割できると仮定したアルゴリズムが用いられている．

$$|R_1|=|M|-H_s+1 \qquad (1)$$

$$|R_i|=\frac{|R|-|R_1|}{H_s-1} \qquad (2\leq i\leq H_s) \qquad (2)$$

ここで，分割バケット数 H_s は次式で与えられる値を用いる．

$$H_s=\left\lceil\frac{|R|-|M|}{|M|-1}\right\rceil+1 \qquad (3)$$

一般に，ユーザの発行する問合せは制約演算や射影演算を含む複雑なものとなるため，各種の演算処理を施した中間リレーションに対して結合演算処理を施す場合が多く，分割バケットのデータ分布をスプリット関数を選択するだけで制御することは困難で，予測通りの分布になることは少ないと考えられる．しかし，ハイブリッドハッシュ結合方式では，各バケットがほぼ主記憶サイズ程度となるように分割バケット数 H_s を小さく抑える方法が採られているため，分割後のバケット分布を予測通りに制御できない場合には，主記憶サイズを越えるバケット（あふれバケット）が容易に生成されることになり，この時の処理性能は，文献7)，11)等に示されるように大きく低下することになる（図 2）．これに対して，我々の提案する動的処理バケット選択方式では，バケットのデータ分布によらずほぼ一定の性能が得られることが示されている．

2.3 動的処理バケット選択方式の概要

前節で述べたように，ハイブリッドハッシュ結合方式では分割バケット数 H_s を小さくとることで各バケットのサイズを主記憶サイズ程度となるようにリレーションの分割を行っており，予測したデータ分布と実際のバケットのデータ分布が異なる場合に，あふれバケットの生成に伴う処理性能の低下が大きいことが問題となっていた．

これに対して動的処理バケット選択方式では，GRACE ハッシュ結合方式と同様に分割バケット数 H_s を大きくとることで，あふれバケットの生成を未然に防ぐことを提案している．これは，次式を満足するように分割バケット数を決定することに相当する．

$$\max|R_i|\leq|M| \qquad (4)$$

また，ハイブリッドハッシュ方式の様に静的に R_1 バケットを決定する方法では，データ分布の変動によっては，分割フェーズとオーバラップして結合演算処理を行うバケット（R_1 バケット）が，ステージング用の主記憶サイズを越える場合（図 3(a)）や，ほとんど利用できない場合（図3(b)）も考えられ，処理性能が低下することになる．我々の提案する方式では，リレーションの分割時に各バケットのサイズ情報を収集し，最も保持タプル数の多いバケットを動的に選択して作業用リレーションに書き出す手法（動的デステージング手法）[7],[11]を用いているため，図3(a)の問題は，以下の条件式を満足する限り生じないと考

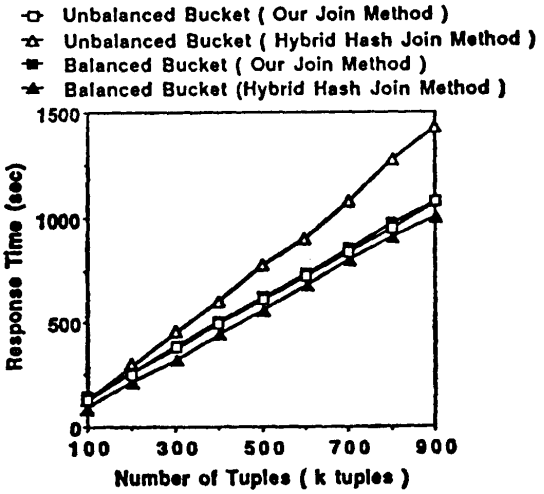
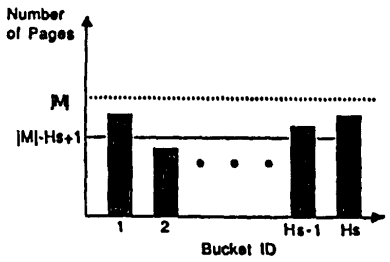
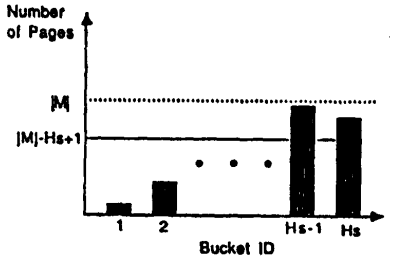


図 2　2つの結合演算処理方式の性能比較
Fig. 2　Comparison of processing time between two hash-partitioned join methods.



(a) A sample data distribution with R1 bucket overflown

(a) R1バケットがあふれるデータ分布例



(b) A sample data distribution without R1 bucket overflown

(b) R1バケットがあふれないデータ分布例

図 3　あふれバケットの生じるデータ分布例
Fig. 3　An example data distribution of buckets.

えて良い．

$$\min|R_i| \leq |M| - H_s + 1 \qquad (5)$$

また，図3(b)に関しては，上記の動的デステージング手法を用いることで，分割フェーズの終了時に $\sum_{i=1}^{t} |R_i| \leq |M| - H_s + t$ を満足する t 個のバケットをまとめて主記憶上にステージングさせておくことができる（R_1 バケットのまとめあげ処理）ため，処理効率の低下を防ぐことが可能となる．

結合属性値に重複がない場合は，結合演算処理に要するコストの大半はディスクとの入出力コストで占められることになるため，ここでは入出力コスト式で結合演算処理の性能を評価する．動的処理バケット選択方式における入出力コストは，その処理アルゴリズムから次式で表現されることになる．

$$C(R,S) = |R| + |S| + \sum_{i=1}^{t} |RES_i| + \sum_{i=t+1}^{H_s} (|R_i| + |S_i|) + \sum_{i=t+1}^{H_s} C(R_i, S_i) \qquad (6)$$

ここで，i 番目のバケットは，スプリット関数値に基づくバケット番号とは異なり，バケットのサイズに基づいて論理的に割り当てられたバケット番号を意味しており，t 番目のバケットまでが，分割フェーズにオーバラップして結合演算処理されることを示している．

3. 不均一なデータ分布に対する結合演算処理性能評価

3.1 性能評価に用いるデータ分布と処理コスト式

本章では，制約演算や射影演算を施した中間リレーションに対して結合演算処理を施す場合を想定し，対象リレーション分割後の各バケットのデータ分布が予測した分布と非常に異なる場合についての性能評価を行う．解析的評価の容易さから各バケットのデータ分布が図4に示す様な三角分布となる場合を仮定すると，

$$\{R\} = \sum_{i=1}^{H_s} \{R_i\}$$

$$\{R_i\} = \{R_1\} \cdot i \qquad (1 \leq i \leq H_s)$$

となり，各バケットに属するタプル数 $\{R_i\}$ は次式で表される．

$$\{R_i\} = \frac{2 \cdot \{R\}}{H_s \cdot (H_s + 1)} \cdot i \qquad (7)$$

また，対象リレーションのタプル数が等しく，結合率が1である（$\{RES\} = \{R\} = \{S\}$）と仮定すると，$|RES| \cong |R| + |S|$ となるため，(6)式で示した動的処理バケット選択方式の処理コストは，

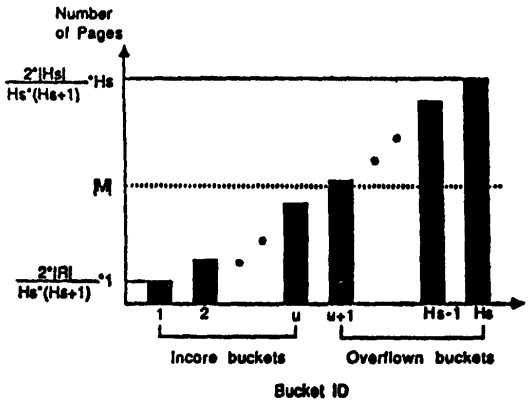


図4 性能評価に用いる不均一データ分布
Fig. 4 An unbalanced data distribution of buckets.

$$C(R,R) = 2 \cdot |R| + \sum_{i=1}^{t} 2 \cdot |R_i| + \sum_{i=t+1}^{H_s} 2 \cdot |R_i| + \sum_{i=t+1}^{H_s} C(R_i, R_i)$$

$$= 4 \cdot |R| + \sum_{i=t+1}^{H_s} C(R_i, R_i) \qquad (8)$$

と表すことができる．

3.2 不均一分布に対する演算処理性能の解析的評価

各バケットのデータ分布が(7)式で表される時，前章で述べた動的処理バケット選択方式の利点を生かす2つの条件式は以下の様になる．

(4)式：$i = H_s$ の時

$$\frac{2 \cdot |R|}{H_s + 1} \leq |M|$$

(5)式：$i = 1$ の時

$$\frac{2 \cdot |R|}{H_s \cdot (H_s + 1)} \leq |M| - |H_s| + 1$$

これら両式を満足する範囲の分割バケット数 H_s をとると，各バケットは再帰的な分割処理を必要としないため，(6)，(8)式で示した動的処理バケット選択方式の処理コストは，

$$C(R,R) = 4 \cdot |R| + \sum_{i=t+1}^{H_s} \left(2 \cdot |R_i| + 2 \cdot \sum_{j=1}^{H_s} |R_{ij}| \right)$$

$$= 4 \cdot |R| + \sum_{i=t+1}^{H_s} 4 \cdot |R_i|$$

となる．これに対して，(5)式は満足するが，(4)式を満たさない場合は，あふれバケットに対する再帰的な分割処理が必要となる．再分割したバケットが(4)

式を満足する場合には，処理コストは次式で示される．

$$C(R,R)=4\cdot|R|+\sum_{i=t+1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}C(R_{ij},R_{ij})\right\}$$
$$=4\cdot|R|+\sum_{i=t+1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}\left(2\cdot|R_{ij}|+2\cdot\sum_{k=1}^{H_s}|R_{ijk}|\right)\right\}$$
$$=4\cdot|R|+\sum_{i=t+1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}\left(4\cdot|R_{ij}|\right)\right\}$$

ここで，t' は，再分割処理を行う際に分割フェーズとオーバラップして処理されるバケットの数を表しており，$|R_i|$ のサイズにより異なった値をとる．

さらに，(5)式も満足しない範囲の H_s をとる場合の処理コストは，R_1 バケットの再分割処理も必要となるため，次式の様になる．ここでも，再分割後のバケットは両式共満足するとしている．

$$C(R,R)=4\cdot|R|+\sum_{i=1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}C(R_{ij},R_{ij})\right\}$$
$$=4\cdot|R|+\sum_{i=1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}\left(2\cdot|R_{ij}|+2\cdot\sum_{k=1}^{H_s}|R_{ijk}|\right)\right\}$$
$$=4\cdot|R|+\sum_{i=1}^{H_s}\left\{4\cdot|R_i|+\sum_{j=t'+1}^{H_s}4\cdot|R_{ij}|\right\}$$

3.3 シミュレーションによる演算処理コストの評価

ステージング領域として，$|M|=100$ ページ用意し(1ページに 32 タプル格納できるとする)，対象リレーションを 10,000 タプルから 100,000 タプルまで 10,000 タプルおきに変化させて分割バケット数が $1<H_s<|M|$ の各値をとる時の演算処理コストをシミュレーションにより測定した時の結果を図5に示す．

ここで，図中の一点鎖線の右領域は(4)式を，点線の右領域は(5)式を満足する H_s の範囲を表している．また，白抜きの各点はハイブリッドハッシュ結合方式における分割バケット数とその処理性能を示している．

ハイブリッドハッシュ結合方式における分割バケット数は，(3)式で与えられるが，各バケットのデータ分布が図4の様に不均一になる場合には(4)式を満足しなくなり，あふれバケットが生成されて，処理性能が低下することは図2にも示した通りである．

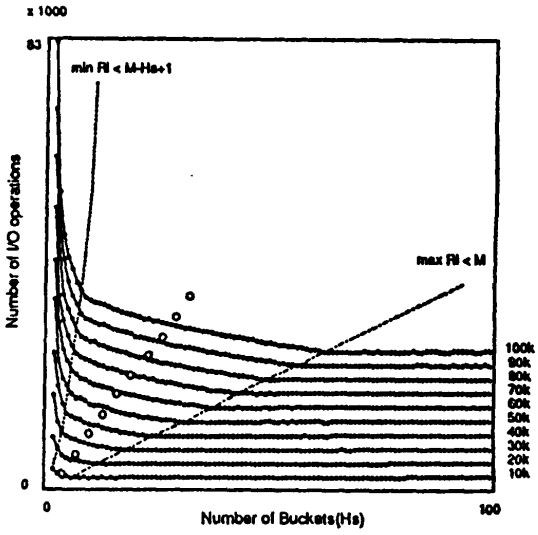


図5 不均一分布データに対する結合演算処理性能
Fig. 5 The performance results for triangular distribution.

前節で示したように，同一サイズのリレーションを結合演算処理する場合でも，分割バケット数を変えて処理すると，(4)，(5)式の条件を満たすかどうかで処理性能に大きな差が生ずる．両式とも満足する環境では，H_s バケットに分割処理する場合と H_s-1 バケットに分割する場合では，R_1 バケットとしてまとめられるバケット数 t が等しいとすると，次式で示すだけ処理性能に差が生じることになる．

$$C_{H_s}-C_{H_s-1}=8\cdot|R|\cdot\frac{t\cdot(t+1)}{(H_s-1)\cdot H_s\cdot(H_s+1)}$$

対象リレーションのサイズが大きい場合には，$t\approx1$ となり，図5に示すように分割バケット数によらずほぼ一定の性能を示すことになるが，対象リレーションのサイズが小さい場合には，$t\approx H_s$ となるため，分割バケット数を変えることで処理性能が大きく変わることになる．この場合の詳細な検討は，次章で行う．

また，(4)式を満足する最小の H_s バケットに対象リレーションを分割処理する場合と H_s-1 バケットに分割する場合の演算処理性能は，次式で示すだけ差が生じることになる．

$$C_{H_s-1}-C_{H_s}$$
$$=8\cdot|R|\cdot\left(\frac{2}{H_s^2}-\frac{t\cdot(t+1)}{(H_s-1)\cdot H_s\cdot(H_s+1)}\right)$$

そして，(5)式を満足する最小の H_i バケットに分割して演算処理を行う場合と，H_i-1 バケットに分割して処理する場合の性能は，次式分の差が生じることになる．

$$C_{H_i-1}-C_{H_i}$$
$$=8\cdot|R|\cdot\left(\frac{1}{H_i\cdot(H_i+1)}-\frac{2}{(H_i-1)\cdot H_i^2\cdot(H_i+1)}\right)$$

この様に，分割後に主記憶サイズを越えるバケットが生成される場合が最も処理性能の低下が大きく，あふれバケットが生成されない様にリレーションを分割して演算処理を施す場合に比べて，約 $8|R|/H_i^2$ 程度の処理コストが増加することになる．また，(5)式を満足しないほど分割バケット数 H_i が小さい場合の処理コストは著しく増加するが，これは，R_1 バケットの再分割処理が必要となるのに加えて，再分割後のサブバケットで主記憶サイズを越えるものが生成されることに起因している．

このことからも明らかな様に，大規模なリレーションに対して結合演算処理を施す場合には，分割後のバケットにあふれバケットを生成させないように極力多くのバケット数で分割処理を行う手法が有効であることが分かる．実際に，分割バケット数を多くとることで生じる処理コストの増加は，あふれバケットの生成に伴うコスト増に比べて非常に小さなものとなることが，図5からも明らかである．

4. バケットサイズ調整機構の有効性について

動的処理バケット選択方式では，分割バケット数を多くとりながら，ハイブリッドハッシュ結合方式と同様に分割フェーズと結合フェーズのオーバラップ処理を行っており，常に GRACE ハッシュ結合方式と同等以上の処理性能を得ることができる．特に対象リレーションのサイズが小さい場合には，オーバラップ処理により削減される入出力コストの割合が，全体の演算処理コストに比べて高くなり，その有効性が明らかとなる．本章では，この様に対象リレーションのサイズが小さい環境での提案方式の性能評価を詳細に行っている．

4.1 バケットサイズ調整機構の種類

対象リレーションのサイズが小さい場合や各バケットのデータ分布が不均一となり図3(b)に示す様な分布をとる場合には，主記憶サイズに満たない小さなバケットが多く存在することになる．この様な場合，動的デステージング手法を用いる我々の結合演算処理方式では，保持タプル数の少ないバケットをいくつかまとめて主記憶上にステージングしたままでリレーションRの分割処理が終了することになるため，それらをまとめて R_1 バケットとしてオーバラップ処理することが可能となる．これを「R_1 バケットのまとめあげ処理」と呼んでいる．

これに対してハイブリッドハッシュ結合方式では，固定的にオーバラップ処理バケットを定めており，他のバケットのデータはバッファを経由して即座に中間リレーションに書き出されるため，この様なバケットのまとめあげ処理を施すことができず，R_1 バケットのサイズが小さくなる場合の性能は，GRACE ハッシュ結合方式の性能とほぼ等しくなってしまう．

また，動的処理バケット選択方式では，中間リレーションに書き出した各バケットに対してもこの手法を適用し，主記憶サイズ程度になる様にいくつかのバケットをまとめて管理することにより，各バケットのフラグメントページ（保持するデータが1ページに満たないページ）に対する入出力コストの削減を図っている．これを「R_i バケットのまとめあげ処理」と呼び，先のまとめあげ処理と区別する様にしている．

フラグメントページは，主記憶内でのバケットの管理を入出力操作にあわせてページ単位で行っているために生じるもので，各バケットに1ページずつ存在する．次節に示す様に，各バケットのサイズが小さく，フラグメントページの占める割合が高い場合には，これらのページに対する入出力コストが無視できなくなり，必要以上の処理コストがかかることもある．

4.2 小規模リレーションに対するまとめあげ処理の有効性

それぞれのまとめあげ処理による効果を調べるために，次に挙げる4種類の方式について処理性能を測定する．この時，対象リレーションのサイズは 10,000 タプルに固定して評価を行っている．

1. ハイブリッドハッシュ結合方式と同様に，まとめあげ処理を行わないで分割バケット数 H_i だけ変化させる方式．
2. R_1 バケットのまとめあげ処理だけ行って，中間リレーションとの入出力処理時のまとめあげ操作は行わない方式．
3. 分割フェーズとオーバラップして処理するバケッ

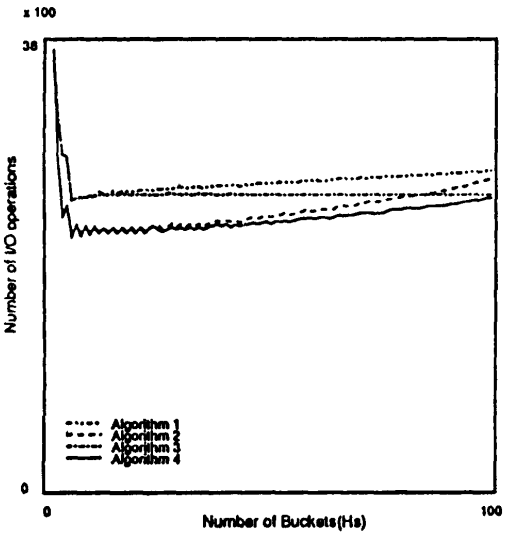


図 6 まとめあげ処理に伴う性能評価(不均一分布)
Fig. 6 The I/O performance for medium sized relation (triangular distribution).

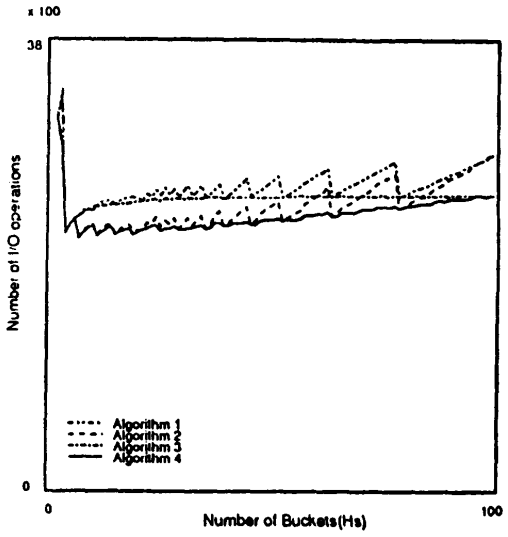


図 7 まとめあげ処理に伴う性能評価 (均一分布)
Fig. 7 The I/O performance for medium sized relation (flat distribution).

トは固定的に決定しておくが，R_i バケットのまとめあげ処理は行う方式.

4. 2つのバケットまとめあげ処理とも行う動的処理バケット選択方式.

まず，分割後のバケット分布が図4に示した三角分布となる場合，分割バケット数 H_s を変化させながら結合演算処理に要する入出力回数を測定すると，図6に示す様な結果を得る.

この結果からも明らかな様に，R_1 バケットのまとめあげ処理は，分割バケット数が少ない場合，すなわち R_1 バケットのステージングに利用できる主記憶容量が多い場合に有効であり，分割バケット数の増加に伴ってその効果が少なくなっていく．しかし，あふれバケットが生成される場合に比べると，処理性能の低下率は十分低いといえる.

また，本章で扱う場合の様に対象リレーションが小さい場合には，分割バケット数 H_s の増加に伴い R_1 バケットのサイズが小さくなり，方式(1)，方式(3)の性能とも GRACE ハッシュ結合方式の性能とほぼ一致するはずである．しかし実際には，方式(1)では，フラグメントページに対する入出力コストが余分にかかるために，分割バケット数を多くとるほど性能が低下することになる．これに対して R_i バケットのまとめあげ処理を行う場合は，フラグメントページの入出力コストを削減する効果があり，分割バケット数が増加しても処理性能にほとんど影響しない.

この R_i バケットのまとめあげ処理の効果は，分割後のバケット分布が均一となる場合に顕著に現れる．図7は，各バケットのデータ分布が均一になる時 ($\{R_i\} = \{R\}/H_s$) の処理性能を分割バケット数 H_s を変化させながら測定した結果であるが，方式(1)，方式(2)の演算処理性能が H_s の値により大きく変動するのに対して，方式(3)，方式(4)の演算処理性能がほとんど H_s の値によらず変動していない．これは，前者がすべてのバケットに対してフラグメントページの入出力操作を必要とするために，フラグメントページの有無による処理コストの増加が分割バケット数により大きく左右することによっているのに対して，後者ではフラグメントページがまとめバケットごとにしか生じないため，分割バケット数にはさほど影響されないことによるものである.

つまり，R_1 バケットのまとめあげ処理は，分割バケット数が小さい場合に有効であり，R_i バケットのまとめあげ処理は，分割バケット数が大きい場合に効果が高いことになる．ともに，処理対象となるリレーションのサイズが小さい場合や不均一な分布をとる場合には効果があるが，対象リレーションが大きい場合には，削減できる入出力コストの上限が主記憶サイズにより制限されるため，効果が薄くなる.

5. ま と め

本論文では，文献 7)で提案した動的処理バケット

選択方式について，分割バケット数の変化に伴う処理性能の変動を中心とした詳細な性能評価を行い，ハイブリッドハッシュ方式では十分な性能を得られなかった不均一なデータ分布に対する分割バケット数の決定方法についてまとめてきた．

この結果，GRACE ハッシュ結合方式と同様に分割バケット数 H_i を多くとることで，主記憶サイズを越えるバケット（あふれバケット）の生成を未然に防ぐことで，分割後の各バケットのデータ分布が不均一になる場合の性能低下を抑えることが，高い処理効率を推持するために必要であることが明らかとなった．特に，動的処理バケット選択方式を用いて大規模なリレーションに対する結合演算処理を施す場合には，すべてのバケットが主記憶サイズを越えない範囲の分割バケット数 H_i をとる限り，ほぼ一定の処理性能が得られることになる．

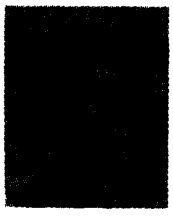
これは，従来のハイブリッドハッシュ方式が分割バケット数を極力小さく抑え，各バケットが主記憶サイズ程度となる様に分割処理を行いながら GRACE ハッシュ結合方式で無効領域となっていた領域を有効利用して処理効率をあげてきたのに対して，逆に，分割バケット数を大きくして細かくリレーションを分割処理し，いくつかのバケットを主記憶サイズに合うように動的にまとめて処理することで処理効率の低下を抑えることができ，かつ各バケットのデータ分布によらず，常に高速に結合演算処理を施すことができることを示している．

さらに本論文では，動的処理バケット選択方式で採用している2つのバケットサイズ調整機構の有効範囲について，その特徴が顕著となる小規模リレーションを対象とした場合の処理性能を測定することで明らかにしている．この結果，分割フェーズとオーバラップして結合演算処理を施す R_1 バケットに対するまとめあげ処理は，分割バケット数 H_i が小さな値をとりステージング用の領域を多く確保できる場合に有効であることが示された．また，中間リレーションに書き出される各バケットに対しても，同様のまとめあげ処理（R_i バケットのまとめあげ処理）を行うことで，分割バケット数が多い場合のフラグメントページに対する入出力コストを削減することができ，処理コストを一定に保つ効果があることが示されている．

参考文献

1) Astrahan, M. M. et al.: System R: Relational Approach to Database Management, *ACM Trans. Database Syst.*, Vol. 1, No. 2, pp. 97-137 (1976).
2) Bratbergsengen, K.: Hashing Methods and Techniques for Main Memory Database Systems, *Proc. of the 10th Int. Conf. on VLDB*, pp. 323-333 (1984).
3) Date, C. J.: *An Introduction to Database Systems*, Vol. II, Addison-Wesley Pub. Co. (1983).
4) DeWitt, D. J. and Gerber, R.: Multiprocessor Hash-Based Join Algorithms, *Proc. of the 11-th Int. Conf. on VLDB*, pp. 151-164 (1985).
5) DeWitt, D. J., Katz, R. H., Olken, F., Shapiro, L. D., Stonebraker, M. R. and Wood, D.: Implementation Techniques for Main Memory Database Systems, *ACM SIGMOD '84*, pp. 1-8 (1984).
6) Kitsuregawa, M., Tanaka, H. and Moto-oka, T.: Application of Hash to Data Base Machine and Its Architecture, *New Generation Computing*, Vol. 1, No. 1, pp. 66-74 (1983).
7) Nakayama, M., Kitsuregawa, M. and Takagi, M.: Hash-Partitioned Join Method Using Dynamic Destaging Strategy, *Proc. of the 14-th Int. Conf. on VLDB*, pp. 468-478 (1988).
8) Stonebraker, M. et al.: The Design and Implementation of INGRES, *ACM Trans. Database Syst.*, Vol. 1, No. 3, pp. 189-222 (1976).
9) Yamane, Y.: A Hash Join Technique for Relational Database Systems, *Proc. on Foundations of Data Organization*, pp. 388-398 (1985).
10) 中山雅哉，喜連川優，高木幹雄：クラスタリング技法に基づく大規模リレーションの結合演算処理方式とその評価，第 36 回情報処理学会全国大会論文集，3F-5, pp. 509-510 (1987).
11) 中山雅哉，喜連川優，高木幹雄：動的クラスタリング技法を用いた結合演算処理方式の性能評価，電子情報通信学会技術報告，DE 87-21 (1988).

(昭和 63 年 12 月 28 日受付)
(平成 元 年 4 月 11 日採録)



喜連川 優（正会員）

昭和 30 年生．昭和 53 年東京大学工学部電子工学科卒業．昭和 58 年同大学院情報工学専門課程博士課程修了．工学博士．同年，東京大学生産技術研究所講師．現在，同研究所助教授．並列コンピュータアーキテクチャ，データベースマシン，データ工学等の研究に従事．電子情報通信学会，電気学会，IEEE 各会員．

中山 雅哉（正会員）

1961年生．1984年慶応義塾大学工学部電気工学科卒業．1989年東京大学大学院工学系研究科情報工学専攻博士課程修了．同年より豊橋技術科学大学知識情報工学系助手．工学博士．データベースシステム，データベースマシンの研究・開発に従事．

高木 幹雄（正会員）

昭和35年東京大学工学部電気卒業．昭和40年同大大学院博士課程修了．工学博士．同年同大生産技術研究所助教授．昭和54年同大教授．昭和59年機能エレクトロニクス研究センター長（兼），現在に至る．昭和46～47年カリフォルニア大学（サンタバーバラ）研究員．昭和40年稲田賞，昭和59年TV学会丹羽・高柳賞業績賞．画像処理の研究などに従事．