データベース・システム 68－3
（１９８８．１１．１７）

不均一分布データに対する高速結合演算処理方式の一考察

中山雅哉，喜連川優，高木幹雄
東京大学生産技術研究所

ハッシュ操作を用いた結合演算処理方式は，他の結合処理アルゴリズムに代る高速な演算処理方式であり，近年盛んに研究がなされている。本稿では，我々の提案するハッシュ結合演算方式（動的処理バケット選択方式）における，対象リレーションのデータ分布が不均一な場合の処理性能について考察を行っている。Hybrid Hash 方式に代表される従来のハッシュ結合方式では，データ分布が均一な場合の評価を主体に行っていた為，分割バケット数を最小にとる方法をとってきたが，本稿における不均一なデータ分布に対する性能評価の結果，分割バケット数を最大にする方法での性能がデータ分布の変動に対して有効に作用することが明らかとなった。

Performance Analysis of High Speed Equi-join Method for Unbalanced Data Distribution

Masaya NAKAYAMA, Masaru KITSUREGAWA, and Mikio TAKAGI
Institute of Industrial Science, University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo, 106, JAPAN

Hash based join algorithm is one of the fastest algorithm among all of the join processing algorithms and many researchers discuss it. This paper shows the performance analysis of our algorithm (Dynamic Hybrid GRACE Hash Join Method) based on hash join method. Most of previous papers treat the balanced distributions of each bucket, however, we focus the unbalanced distributions of each bucket and get a new result in this environment. For unbalanced distributions of data, we had better choose the largest number of buckets instead of smaller ones, which was chosen by Hybrid Hash Join Method, one of the previous Hash based join methods.

1．はじめに

結合演算の処理負荷は，他の関係代数演算の処理負荷に比べて重いことは良く知られており，多重ループ方式や，ソートマージ方式，ハッシュ結合方式等，種々の処理方式が提案されてきた［1，2，3］。このうち，ハッシュ結合方式は，特に大規模なデータベースを扱う環境において他の結合演算処理方式に比べて高速に処理可能であることが知られている［3，4］。

しかし，従来提案されてきたハッシュ結合方式（ハイブリッドハッシュ結合方式等）は，演算処理に先立って静的に分割バケット数Ｈｓと各バケットの実行順序を決定しており，対象とするリレーションのデータ分布により処理性能が大きく変動する［5，6］。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，対象リレーションのデータ分布に応じて各バケットの処理順序を動的に変更するアルゴリズムをとっており，リレーションのデータ分布の変動に対して性能にほとんど影響を及ぼさないことが示されている［5，6］。

本稿では，この提案アルゴリズムをより詳細に解析し，リレーションを分割するバケット数Ｈｓと結合演算処理性能の関係について考察を行う。これに加えて，提案アルゴリズムの持つ３つの特徴（①オーバーラップ処理バケットのサイズ調整，②各処理バケットのサイズ調整，③各バケットの処理順序調整）が，どの様な環境において有効に作用するかについても，同時に検討している。

以下，２章では，ハイブリッドハッシュ結合方式について概説し，静的に分割バケット数やバケットの処理順序を決定する従来のハッシュ結合方式についてまとめている。３章では，我々の提案する結合演算方式について述べ，アルゴリズムの特徴について概説する。そして，４章で，均一なデータ分布，不均一なデータ分布の各々について，各特徴に対する演算処理性能を分割バケット数Ｈｓを変化させながら評価する。また５章では，これらの結果に基づいてハッシュ結合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，６章で全体のまとめを行う。

2．ハイブリッドハッシュ結合方式について

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 DeWitt 氏らにより提案されたハッシュ結合方式で，従来のハッシュ結合方式の代表的な結合演算処理方式である。この方式は，単純ハッシュ方式と，ＧＲＡＣＥハッシュ方式［3，4］の両利点を取り入れた２フェイズ（分割フェイズと結合フェイズ）アルゴリズムをとる。特に主記憶が処理対象リレーションのサイズ程度に大きい場合は，分割処理とオーバーラップして処理できるバケットのサイズを大きくとることができる為，高速な結合演算処理が可能となる。以下にその処理アルゴリズムの概略を示す。

（１）対象リレーションＲを主記憶上に読み込んで次式①で示されるＨｓ個のバケットに分割していく（分割フェイズ）。このうち，R_1 バケットは主記憶上に常駐する様にスケジュールされ，他の全てのバケットは中間リレーション上で管理する為ディスクに書き戻される。

$$Hs = \left\lceil \frac{R - M}{M - 1} \right\rceil + 1 \qquad ①$$

（２）もう一方の対象リレーションＳを（１）と同様にして主記憶上に読み込みながらＨｓバケットに分割する。この時，S_1 バケットの各タプルは，即座に R_1 バケットの各タプルと結合処理を行って，結果リレーションに書き出していき，それ以外のバケットは，中間リレーションに書き出して管理される。

（３）両リレーションの分割処理が終了すると，各バケット毎に結合演算処理が実行される（結合フェイズ）。この時，各バケットが予測通りのデータ分布となれば，各 R_i バケットは主記憶サイズ以下となり，通常にハッシュ結合処理を実行できる。しかし，データ分布が不均一になる場合は，主記憶サイズを超えるバケット（あふれバケット）が生成されることになり，これらのバケットは主記憶サイズ以下になるまで再帰的に分割が行われて結合処理が実行される。

この方式の特徴は，対象リレーションを予め分割処理して結合可能性を絞り込み，対象リレーションを２回分入出力するだけで結合演算処理を実行できる点（ＧＲＡＣＥ方式）と，リレーションＳの分割処理とオーバーラップして Ｒ バケットの結合処理を行うことで，このバケットに関する入出力操作を軽減させることができる点（単純ハッシュ方式）にある。

対象となるリレーションのサイズが大きくなると，分割バケット数Ｈｓが大きくなり分割フェイズにオーバーラップして処理される R_1 バケットのサイズが小さくなる為，後者の特徴は，リレーションサイ

ズが小さい場合に特に有効なものとなる。

3．動的処理バケット選択方式による結合演算処理

3．1．ハイブリッドハッシュ方式との違い

ハイブリッドハッシュ方式では，図1（②式）に示す様にR_1バケットと，それ以外のバケットのサイズを変えてリレーションを分割できるとして処理アルゴリズムを構築している。

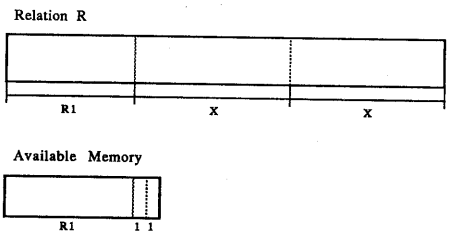

図1　ハイブリッドハッシュ方式における
各バケットのデータ分布

$$Ri = \begin{cases} M - Hs + 1 & (i = 1) \\ \dfrac{R - R_1}{Hs - 1} & (2 \leqq i \leqq Hs) \end{cases} \qquad ②$$

しかし，用意したスプリット関数がうまくリレーションを分割できない場合には，(1) R_1バケットが（M−Hs＋1）ページを超えるサイズになったり，(2) R_iバケット（$2 \leqq i \leqq Hs$）が主記憶サイズを超えたり（あふれバケット）して演算処理性能が低下することになる。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（Dynamic Hybrid GRACE Hash Join）では，各バケットのサイズが不均一になる場合でも高速に演算処理を実行できる様に以下の点を改良したアルゴリズムをとっている。

まず，(1) の問題に対しては，R_1バケットを他のバケットと区別しないで，分割処理時に動的に書き出すバケットを決定しながら（動的デステージング手法）主記憶空間にフィットするバケットを最終的にR_1バケットとする方法をとっている。

また，(2) の問題に対しては，予め分割するバケット数Hsを多くとる様にして，最も大きいバケットでも主記憶サイズを超えない様にしている（図2）。しかし，分割バケット数を増大させると，R_1バケットをステージングする為の空間が減少する点や，各バケットにおけるフラグメントページの占める割合が増加する点で新たに問題が生じる為，この解決策として，複数のバケットをまとめて処理する（バケットサイズ調整）手法を用いている。

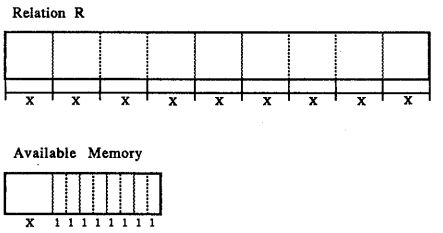

図2　動的処理バケット選択方式における
各バケットのデータ分布

3．2．動的処理バケット選択方式の特徴

動的処理バケット選択方式において，全てのバケットが主記憶サイズを超えない様にする為には，次式③の条件を満足すれば良い。また，ハイブリッドハッシュ方式と同様に分割フェイズとオーバーラップして一部の結合演算処理を実行させる為の条件は，式④の様になる。

$$\max Ri \leqq M \qquad ③$$

$$\min Ri \leqq M - Hs + 1 \qquad ④$$

これらの式は，各バケットがほぼ均一なサイズになると仮定すると，次式⑤の様にまとめられて，ある対象リレーションRを分割処理する際にとるべきHsの値を，主記憶サイズMを用いて規定することになる。

$$Hs^2 - (M+1) * Hs + R \leqq 0 \qquad ⑤$$

結合演算処理を施す対象リレーションのサイズを固定した時に⑤式の範囲でHsの値を変化させると，各バケットのサイズが変化して，オーバーラップ処理を施すバケットや中間リレーションに書き出すバケット（デステージングバケット）が，単独では主記憶サイズにフィ

ットしない状況が生じる。まず，Ｈｓを大きな値にとると，各バケットのサイズが主記憶サイズに比べて小さくなり，分割処理時や結合処理時に無効な領域（フラグメントページ）が生じることになる。我々は，これに対してデステージングするバケットをいくつか集めて主記憶サイズにフィットする様にまとめあげる（処理バケットのサイズ調整）処理を行って，余分な入出力を抑える手法とっている。これはデステージングされるバケットだけでなく，オーバーラップ処理するバケットについても適用することが可能であり（オーバーラップ処理バケットのサイズ調整），対象リレーションのサイズが小さい場合に特に有効な手法である。

また，逆にＨｓの値が小さい場合には，図３の様にオーバーラップ処理するバケットがステージング空間を超えない様に，動的デステージング手法を用いて小さなバケットが主記憶上に残る様にスケジューリングを行う（バケットの処理順序調整）。この手法により，各バケットのデータ分布が不均一になる場合でも有効にオーバーラップ処理するバケットを用意することができる。

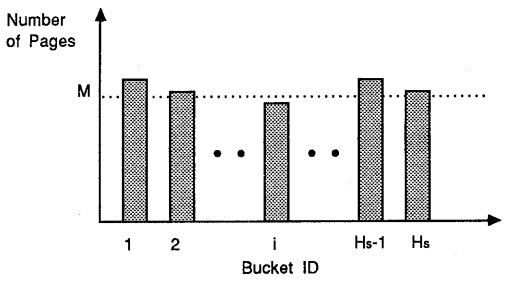

図３　あふれバケットの生じた不均一なデータ分布例

以上に示した様に，動的処理バケット選択方式には３つの特徴があり，各々次の様な状況に対して有効である。

（１）オーバーラップ処理バケットのサイズ調整：

対象リレーションのサイズが小さく，分割フェイズにおいて複数のバケットを主記憶上にステージングできる場合には，常駐させるバケットをＲ　だけに限定せず，いくつかまとめてオーバーラップ処理バケットとして扱うことで入出力コストの削減を図ることができる。

（２）各処理バケットのサイズ調整：

同様に，各バケットのサイズが主記憶サイズに比較して小さい場合は，処理バケットを主記憶サイズにフィットする様にまとめあげることができる。特に，各バケットにおけるフラグメントページ（１ページに満たないデータ領域）の割合が多い場合には有効である。

（３）各バケットの処理順序調整：

分割バケット数Ｈｓが小さい場合や，分割バケットのデータ分布が不均一となる場合には，Ｒ　バケットがステージング空間を超えることがある為，動的デステージング手法を用いてバケットの処理順序の調整を行うことで，あふれバケットの生成に伴う処理性能の低下を防ぐことができる。また，バケットの処理順序の調整により，主記憶に常駐されるバケットをサイズの小さいものだけ集める効果があり，（１）のオーバーラップ処理バケットの調整にも有効に利用できる。

４．動的処理バケット選択方式の性能評価

提案する結合演算処理方式の性能評価を行うにあたり，対象となるリレーションや，データ分布について，以下の様な仮定をする。

(1)　対象リレーションは固定長タプルから成り，入出力の単位である１ページ内にｐ（＝３２）タプルが保持されるとする。

(2)　結合演算処理に利用できる主記憶サイズは演算処理の間は一定であり，Ｍ（＝１００）ページとする。

(3)　結合処理される属性は重複値をとらず，結果総量は対象リレーションサイズの和と等しいとする。

４．１．バケットサイズが均一となる場合の性能評価

動的処理バケット選択方式において，分割後の各バケットが均一なサイズとなる場合には，⑤式で示した様な範囲のＨｓをとれば，各バケットのサイズが主記憶サイズ以下になり，分割フェイズとオーバーラップして処理するバケットを用意することもでき，高速に演算処理を施すことができると予想される。ここでは，最適なＨｓの決定方法について検討を行う為に，全てのＨｓの値に対する処理性能について評価を行う。

対象リレーションが１０ｋ，２０ｋ，３０ｋ，４０ｋ，５０ｋ，６０ｋ，７０ｋ，８０ｋ，９０ｋ，１００ｋタプルとなる各場合について，Ｈｓの値を２～９９まで変化させた時に演算処理に要する入出力回数を測定すると，図４の様になる。また，規模の小さいリレーション（１０ｋ，２０ｋ，３０ｋタプルのリレーション）についての結果だけをまとめてグラフにすると，図５の様になる。

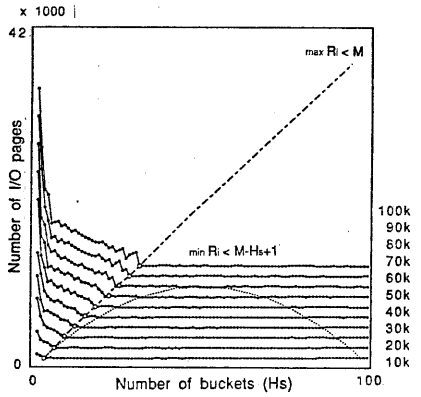

図４　均一なデータ分布をとる場合の

動的処理バケット選択方式の性能

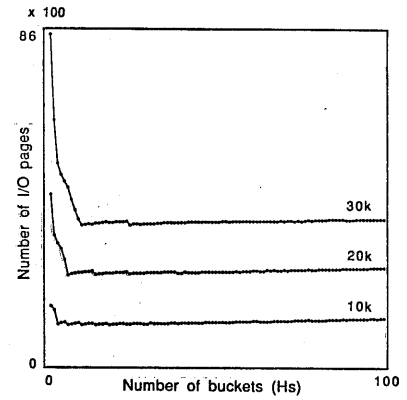

図５　小規模リレーションに対する性能（均一）

ここで，本方式の処理性能を他の方式と比較する為に，図中の○点でハイブリッドハッシュ方式における演算処理入出力回数を示している。また，図中の一点鎖線による直線は，③式の条件を満足する最小のＨｓを示すもので，このラインの示す値より小さなＨｓをとって分割処理を行うと，あふれバケットが生成されて処理性能が低下することが示されている。最後に，破線で示した曲線は，④式（⑤式）に相当するもので，この曲線内のＨｓをとれば，分割フェイズにオーバーラップして処理するバケットを用意することができる。

４．２．バケットサイズが不均一となる場合の性能評価

次に，分割後の各バケットのデータ分布が不均一となる場合の性能評価を行う。我々の方式では，どのバケットをステージングするかについて分割処理時に動的に決定している為，各バケットに属するタプルの処理順序によってオーバーラップ処理するバケットの構成が変ってくる。ここでは，各バケットに入るタプルの到着確率がバケットサイズに比例すると仮定して，分割処理の途中でデステージングされるバケットが最終的に多くのタプル数を保持するバケットから選択されるとして評価を行っている。また，不均一なデータ分布として各種のデータ分布が考えられるが，ここでは，評価結果の解析が比較的楽になる三角分布を仮定することにする（図６）。

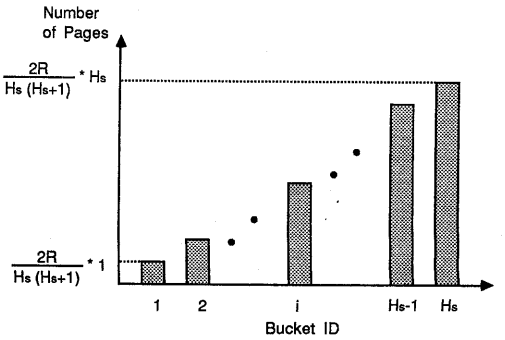

図６　不均一なデータ分布の例

この様なデータ分布に対して前節と同様にして，対象リレーションを１０ｋ～１００ｋタプルとした時に，分割バケット数Ｈｓを変化させながら演算処理に必要な入出力回数を測定する（図７）。

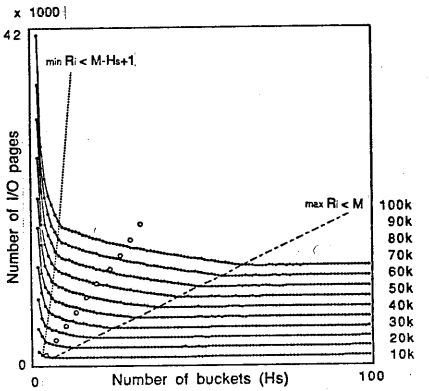

図７　不均一なデータ分布をとる場合の

動的処理バケット選択方式の性能

図中の○点，一点鎖線，破線は均一なデータ分布をとる場合と同様に，ハイブリッドハッシュ方式の性能と，全てのバケットが主記憶サイズ以下となる最小Ｈｓ値及び，オーバーラップ処理されるバケットが存在するＨｓ値を示している。

5．提案方式の処理性能に対する考察とＨｓの決定方法

本章では，前章に示した各データ分布に対する提案方式の処理性能について，各特徴がどの様な環境で有効に作用するかについて示すと共に，分割バケット数Ｈｓの決定方法について検討を行う。

5．1．各バケットのデータ分布が均一な場合の性能

バケットのデータ分布が均一になる場合の性能は，図４及び，図５に示す様になる。両図からも明らかな様に，各バケットのサイズが主記憶サイズを超えない様に分割バケット数Ｈｓをとれば，特に⑤式を満足するＨｓとしなくても処理に必要な入出力回数はほとんど変化せず，ほぼ一定の処理性能が得られることになる。

これは，２種類のバケットサイズ調整手法の結果得られるものであるが，それぞれの効果についてより詳細に評価を行う為に，ここでは，次の３種類のアルゴリズムに対して同様の性能評価を行っている（図８～図１３）。

(1) ハイブリッドハッシュ方式と同様に，オーバーラップして処理するバケットを固定にし，処理バケットのまとめあげをしないアルゴリズムを指定されたＨｓ値を用いて処理する方法（多分割ＨＨ法）

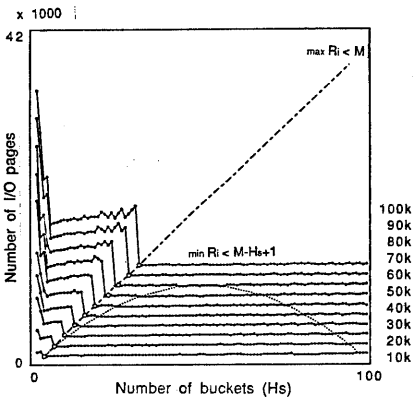

図８　均一なデータ分布をとる場合の

多分割ＨＨ法の性能

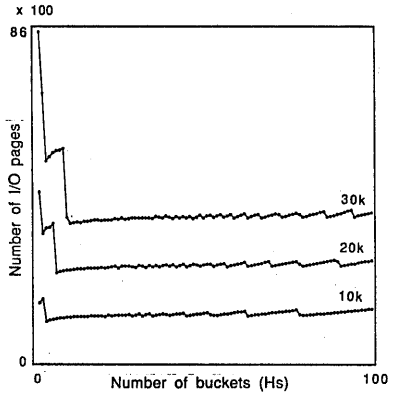

図９　小規模リレーションに対する

多分割ＨＨ法の性能（均一分布）

(2) オーバーラップ処理するバケットのまとめあげ処理を行い，複数バケットを一括処理する場合の性能（Ｒ１バケットまとめあげ法）

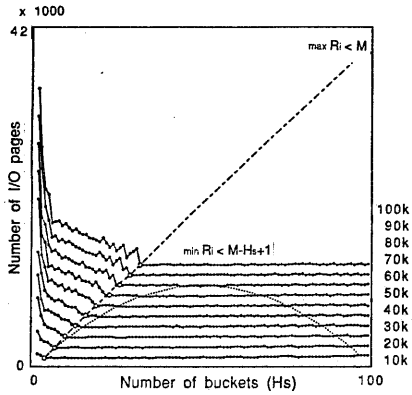

図１０　均一なデータ分布をとる場合の

Ｒ１バケットまとめあげ法の性能

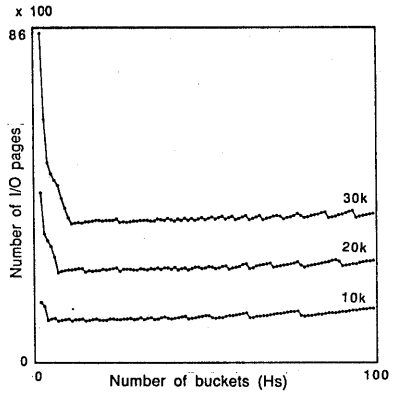

図１１　小規模リレーションに対する

Ｒ１バケットまとめあげ法の性能（均一分布）

⑶ バケットをデステージングする際にいくつかまとめて主記憶サイズにフィットする様にスケジュールする場合の性能（Ｒｉバケットまとめあげ法）

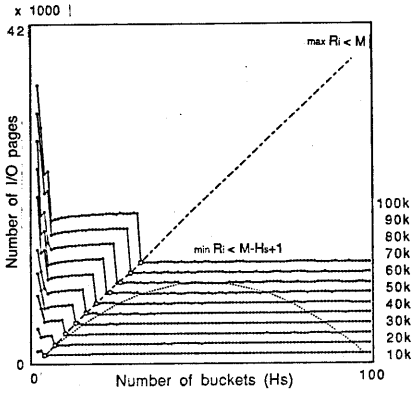

図１２　均一なデータ分布をとる場合の

Ｒｉバケットまとめあげ法の性能

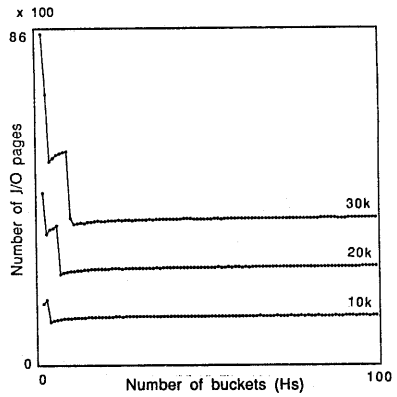

図１３　小規模リレーションに対する

Ｒｉバケットまとめあげ法の性能（均一分布）

図8，図１２に示される様に，主記憶サイズを超えるバケットばかりになるＨｓをとると，処理性能が大きく低下するが，これに対して，オーバーラップ処理するバケットのサイズを調整する方法（図4，図１０）を用いると，性能低下を大幅に抑えることが可能となる。これは，再帰的に分割処理される各あふれバケットがほぼ主記憶サイズとなる為に生じる結果である。

これに対し，図9，図１１に見られる様に，処理バケットのまとめあげを行わない方法では，分割バケット数Ｈｓの値により処理性能がある程度変化する。これは，対象リレーションのタプル数と分割バケット数Ｈｓの関係でフラグメントページが生じる為に起る現象である。ここで，デステージングの際にバケットのまとめあげを行う方法を用いると，フラグメントページの影響を打ち消すことができ，図5，図１３に示される様に，Ｈｓの値によらずほぼ一定の性能を得ることができる。但し，これにより得られる性能向上の度合は低いものとなる。

5．2．各バケットのデータ分布が不均一な場合の性能

データ分布が不均一な場合の結合演算処理性能は，図７に示す様になる。この場合は，データ分布が均一な場合と異なり，③式を満足するＨｓの値の範囲が④式を満足する値の範囲より狭くなる為，ハイブリッドハッシュ方式で用いている分割バケット数Ｈｓでは，主記憶サイズを超えるバケットが発生して，処理性能が大幅に低下することになる。前節と同様に，提案方式における図７中の破線と一点鎖線の間

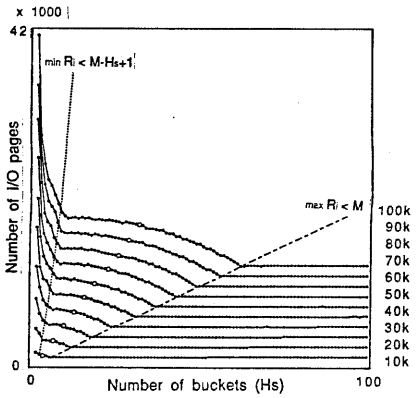

図１４　不均一なデータ分布をとる場合の

多分割ＨＨ法の性能

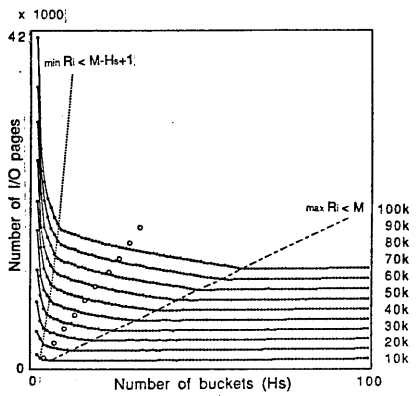

図１５　不均一なデータ分布をとる場合の

Ｒ１バケットまとめあげ法の性能

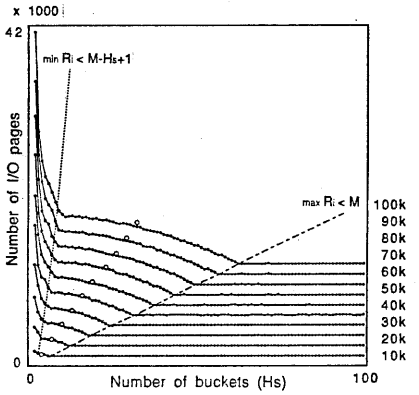

図１６　不均一なデータ分布をとる場合の

Ｒｉバケットまとめあげ法の性能

の性能低下を抑えている要因を調べる為に，先に示した３種類のアルゴリズムに対してＨｓを変化させながら不均一データ分布に対する性能評価を行った（図１４～図１６）。

この様に，不均一なデータ分布に対しても，オーバーラップ処理するバケットのサイズ調整が有効に働くことがわかる。特に，図中の破線と一点鎖線の間では，この手法があふれバケットに対する再帰処理においても，最初のステージング処理においても作用できる為，非常に有効である。

５．３．分割バケット数Ｈｓの決定方法

２種類のデータ分布に対する性能評価の結果，動的処理バケット選択方式では，各バケットを主記憶サイズ以下とする様に分割バケット数Ｈｓをとれば，その性能はＨｓの値によらずほぼ一定にできることが明らかとなった。これは，Ｈｓを小さくして，あふれバケットを生成する場合の処理効率の低下は十分に処理アルゴリズムで吸収できないのに比べて，Ｈｓを大きくしてオーバーラップ処理できなくなる場合の処理効率の低下はバケットサイズの調整機構で吸収することが可能な為である。任意のデータ分布に対して，最も柔軟に処理を高速に実行するには，ハイブリッドハッシュ方式等で用いている小さな分割バケット数Ｈｓを用いないで，最も大きな値を採用することが望ましいことになる。

６．まとめ

本稿では，我々の提案するハッシュ結合演算方式（動的処理バケット選択方式）における性能について，対象リレーションのデータ分布が均一な場合と不均一な場合のそれぞれについて評価を行い，採用している処理アルゴリズムの有効性について考察を行った。ハイブリッドハッシュ方式に代表される従来のハッシュ結合方式では，データ分布が均一な場合の評価を主体に行っていた為，分割バケット数を最小にとる方法がとられてきたが，本稿における不均一なデータ分布に対する性能評価の結果，分割バケット数を最大にとる方法が最も有効に，データ分布の変動に対して最も有効に作用することが明らかとなった。

本稿で扱っているリレーションは，結合属性のデータに重複がなく，結果リレーションのサイズが対象リレーションのサイズの和になる場合を想定している。この為，入出力回数で演算処理性能として評価することが可能となるが，実際のリレーションにおける結合演算処理では，分割バケットのデータ分布が変動するばかりでなく，データ重複によるハッシュ値の衝突や，結合率による結果リレーションのサイズの増減等，種々の要因をさらに加味して評価する必要がある。今後は，ワークステーション上に実装を終えている動的処理バケット選択方式を用いた処理システムを用いてこれらの評価を行っていくつもりである。これらに関する評価結果については，機会をかえて報告する。

参考文献

[１] M. Stonbraker, et al.,「The design and implementation of INGRES」, ACM TODS, vol.1, no.3, 1976

[２] M. M. Astrahan, et al.,「System R: Relational Approach to Database Management」, ACM TODS, vol.1, no.2, 1976

[３] M. Kitsuregawa, et al.,「Application of hash to data base machine and its architecture」, New Generation Computing, vol.1, no.1, 1983

[４] D. DeWitt, et al.,「Multiprocessor Hash-Based Join Algorithms」, Proc. of Conf. on VLDB 1985

[５] 中山他,「動的クラスタリング技法を用いた結合演算処理の性能評価」, 信学技法, DE87-21, 1988

[６] M. Nakayama, et al.,「Hash-Partitioned Join Method Using Dynamic Destaging Strategy」, Proc. of Conf. on VLDB 1988