(40)

「アドバンスト・データベース」シンポジウム　昭和59年12月

データベースマシンGRACEのアーキテクチャとその実行制御系

伏見　信也[†]　喜連川　優[††]　田中　英彦[†]　元岡　達[†]
([†]東京大学　工学部　[††]東京大学　生産技術研究所)

社団法人 情報処理学会

データベースマシンGRACEのアーキテクチャとその実行制御系

伏見　信也†　喜連川　優††　田中　英彦†　元岡　達†
(†東京大学　工学部　††東京大学　生産技術研究所)

1．はじめに

並列関係データベースマシン GRACE〔1，2〕はhashとsortに基づく並列アルゴリズムによりJOIN等の処理負荷の重い関係代数演算を高速に処理すると共に，マシン全体の記憶空間を階層化することにより大規模データベースを効率良く支援することができる。本稿では GRACEのアーキテクチャについて，その計算モデル，論理アーキテクチャ，物理アーキテクチャ，の3つの側面を実行制御の観点から体系化し，並列処理の制御方式，マシンの資源管理等について考察する。

2．並列データベースマシンの実行制御

1970年代中期から並列データベースマシンの研究は活発化しているが，試作，性能評価を終えているものは極めて僅かである。これらマシンに於いては，並列化に伴う実行制御のオーバヘッドが極めて大きいことが明らかとなった。例えば，〔8〕ではマシン内での制御単位を固定長ページとした実行制御を行っているが，処理時間の大部分は実行制御モジュールに於けるプロセッサ群へのページ割付け処理に費されている。実行制御を司る制御モジュールは一般に過負荷になる傾向があり，並列実行制御に伴うオーバヘッドは無視できない。この様に並列データベースマシンに於ける実行制御方式は極めて重要であることが判る。

本稿ではデータベースマシン GRACEの実行制御方式として，実行制御単位をページ等の小さいgranuleではなくオペランドリレーション全体が構成する巨大なデータフローとし，演算間の並列制御にデータフロー制御を用いた，データベース処理環境に於いて自然で効率の良い並列実行制御方式を提案する。即ち，マシン内のプロセッサメモリ等の資源の割当てを一つの関係代数演算単位に行い，オペランドリレーションが得られると同時にオペランドリレーションを構成するタプル群が割付けられたマシン資源中を一つのデータフローとなって流れることによって一演算の実行を行う。本方式では実行制御が介入することなく関係代数演算が実行され，制御負荷の集中が避けられる為，実行制御時間の対演算時間比は飛躍的に改善され，実行制御オーバヘッドが大幅に減少することが期待される。

3．計算モデル

GRACEの計算モデルに於いては，ユーザが発行したトランザクションは以下の様な階層構造に翻訳される。トランザクションは一つのアプリケーションプログラムであり，1つまたは複数の問合せのシリアルな実行である。問合せとは，問合せ木と呼ばれる木構造を持ったタスクの集合であり，ＳＱＬ等に於ける1 statement に対応する（図1）。タスクはJOIN等の関係代数演算一つの処理や，SORT，AGGREGATION の実行に相当する。問合せ木に於けるノードの親

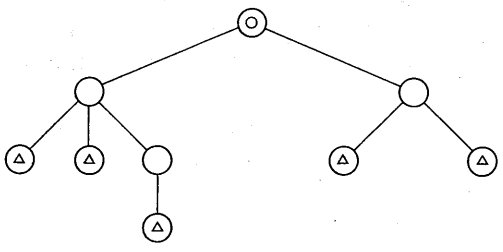


Fig. 1　The Structure of Query

子関係に従ってタスクの親子関係が定義される。データベースは論理ページと呼ばれる単位に分割されており，問合せが参照する論理ページの集合を問合せのread set，また問合せが更新する論理ページの集合及び結果の出力資源（ＬＰ，ディスプレイ等）を問合せの write setと呼ぶ。問合せ木のタスクの内，葉に相当するタスクを基底タスク，根に相当するタスクを出力タスク，他のタスクを中間タスクと呼ぶ。基底タスクはデータベースそのものに論理ページを単位としてアクセスし，SELECTION,重複除去を含まないPROJECTION等を実行する。一方，出力タスクは結果のデータベースへの書込み，結果の表示等を行う。タスクの構造を図２に示す。全てのタスクはSource空間，Filter処理，Clustering処理，Sink空間の４つの要素から構成される。タスクはそのSource空間からSink空間へのデータの送出を行うことにより実行され，これによって１つの関係代数演算の処理が行われる。Source空間には送出されるタプルが格納されており，これらタプルは巨大なデータフローとなって不要なタプルのふるい落しやタプルの変形（Filter処理）を受け，更に次タスク用の前処理（Clustering処理）を施されてSink空間へ流れ込む。

我々の計算モデルに於ける基本原理は，全ての関係代数演算，AGGREGATION 等をこのタスク構成により実行することである。即ち，オペランドリレーション全体を予め当該演算に対して互いに独立なcluster に分割しておき（Clustering処理），これらcluster に属するタプルが構成する複数の並列データフローの束に対して並列に演算を行う（Filter処理）。全ての演算は，データフロー中のタプルのふるい落し，sort等の組合せであり，ここではこれらを一括してデータフローに対するFilter処理とみなす。タスク中ではFilter処理と同時に次タスクに対するClustering処理がデータフローに沿って施される為，Clustering処理に関するオーバヘッドは実効的に存在しない。

以下では問合せ中のタスク群の実行制御について，その抽象的な実行モデル（論理アーキテクチャ），及びハードウェアによるその実現手法（物理アーキテクチャ）について考察する。

４．GRACEの論理アーキテクチャ

4.1 タスクの実行環境

GRACEの論理アーキテクチャでは，上記の４つのタスク構成要素に対し，各々専用の論理ユニット（Logical Unit: ＬＵ）の存在を仮定する。タスクの実行環境はこれら４つのタスク構成要素に対して各々複数台のＬＵを割り当て，更にデータフローが流れる為に必要な通信資源を割当てることにより実現され，一般に

＜ Source空間構成ＬＵの集合，
Filter処理ＬＵの集合，
Clustering処理ＬＵの集合，
Sink空間構成ＬＵの集合，
通信資源の集合 ＞

の５つ組によって表現される。実行環境はタスクの種類に応じて図３の様に具体化される。即ち，中間タスクのSource空間及び

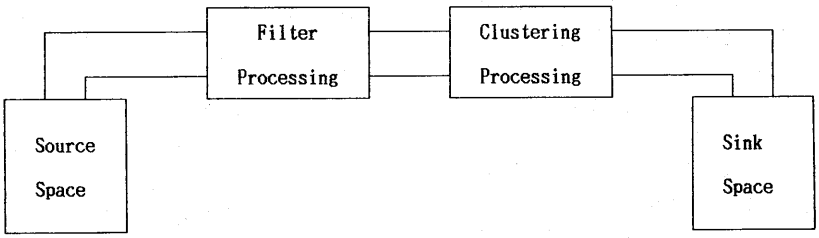
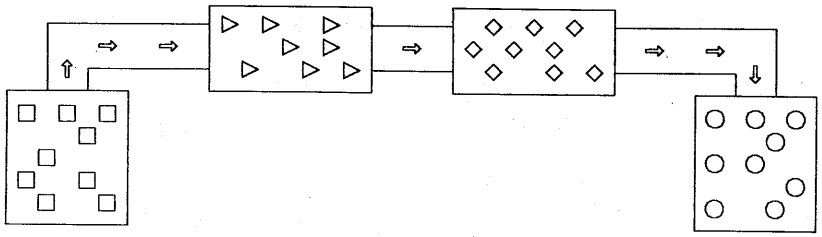
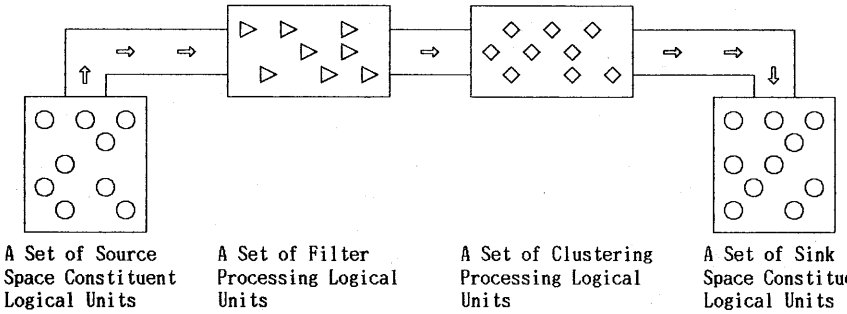


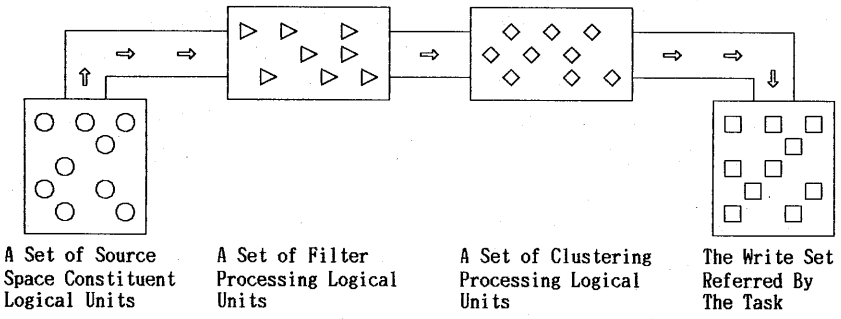
Fig. 2 The General Structure of Task



(a) Base Task



(b) Internal Task



(c) Output Task

○: Source Space Constituent Logical Unit ▷: Filter Processing Logical Unit
◇: Clustering Processing Logical Unit □: Logical Page

Fig. 3 The Execution Environment of Task in The Logical Architecture of GRACE

Sink空間は各々の構成ＬＵの集合により形成されるが，基底タスクに対するSource空間は問合せのread setの内基底タスクが参照する論理ページの全体となり，また出力タスクに対するSink空間は問合せの write set により構成される。

4.2 実行制御方式

GRACEの論理アーキテクチャに於ける問合せの実行制御は，所謂データフロー制御を基本としている。問合せはそのread setが確保可能の時実行可能となる。問合せの実行開始と共に，問合せ中のタスクはその木構造が許す限り並列に発火，実行され，出力タスクの終了をもって問合せの実行が終了する。タスクの接続は，タスクの終了後そのSink空間を次タスク（親タスク）のSource空間とすることによりなされ，問合せの木構造は，子タスク達がそのSink空間を共有することによって実現される。

タスクは，その実行環境の割付が完成し，そのタスクに対する一意的なId（タスクId）が与えられた時発火可能であると言う。即ち，タスクのSource空間に対して子タスクからのデータフローが全て流れ込み，上記５つ組で表される実行環境が与えられ，各モジュールに対し処理すべき関係代数演算に応じたコマンド群が与えられ，更にこれらモジュール群がタスクIdによってタスクの構成を認識することによってタスクは駆動される。この意味でタスクは GRACEの論理アーキテクチャに於いて最小の制御／実行の単位であり，その発火の後は実行制御を必要とせず，自律的に動作する。

ここで基底タスクのSource空間の割付は，問合せの発火，即ちread setに属する論理ページの確保によって保証されており，これは論理ページのreadロックに対応する。また前述のタスクの接続によって，以降のタスクの発火に際しては，常にSource空間が与えられていることになる。一方，出力タスクのSink空間の割付けは write setの確保，即ち更新論理ページの writeロックとなり，データベースのロックと資源割付は統一的に行うことができる。

4.3 タスク処理アルゴリズム

論理アーキテクチャに於いては，個々の関係代数演算の処理はタスクの処理とほぼ一対一に対応する。ここでは代表的な関係代数演算であるJOINとSELECTION を例にとって，その処理アルゴリズムを説明する。

4.3.1 SELECTION

SELECTION 演算は通常基底タスクとして処理される。SELECTION の実行は，先づリレーションテーブルによって参照リレーションを決定し，続いて論理ページインデックスによりその論理リレーションを構成する論理ページの内，必要とされるものを絞り込みこれをSource空間とする。その後，これら論理ページにアクセスして，格納されている全てのタプルをデータフローとしてFilter処理に送り込む。SELECTION タスクに於けるcluster 及びSource空間は各々論理ページ，必要最小限の論理ページの集合として定義され，そのアクセス単位はタプルではなく論理ページとなる。また，この場合のFilter処理は実際のSELECTION 演算であり，流れ込んでくるデータフローからSELECTION predicate を満たすタプルのみをClustering処理に送り，次タスク（親タスク）で実行される演算に対し，Clustering 処理を施した後，Sink空間に転送する。従ってSELECTION タスクに於けるClustering 処理とは，二次記憶系の内部に於ける SELECTION演算に対しての最適な論理ページへの分割であり，SELECTION predicateを満たすタプルを局所化してアクセスされる論理ページの数をできる限り小さく保ち，二次記憶系に対するアクセス回数をできるだけ少なくすることに帰着される。このClustering処理をここでは静的Clustering （Static Clustering）と呼ぶ。

4.3.2 JOIN

GRACEに於けるJOIN処理はhashによる処理負荷の分散を基本とする〔２〕。JOIN演算は所謂nestedループアルゴリズムによってO（Ｍ・Ｎ）時間（Ｍ，Ｎ：リレーションサイズ）で実行されるが，タプルのJOIN属性に対して予めhashを施し，hash値の等しいタプルを集めてバケットを構成し，オペランドリレーション全体を小さなバケット

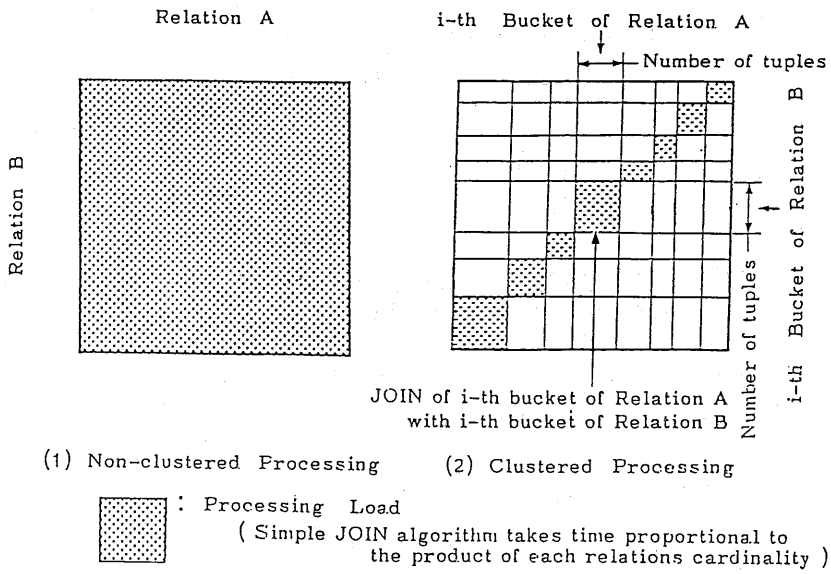


Fig. 4 Load Reduction Effects by Clustering Feature of Hash in Join Operation

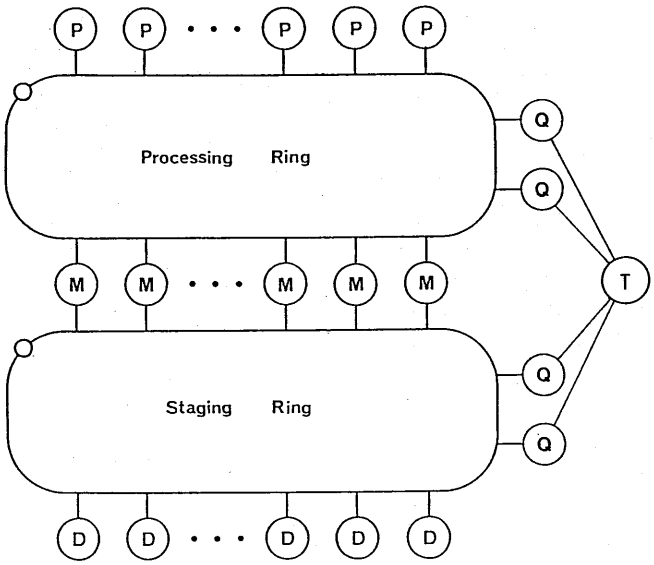
に分割することにより$O(\Sigma mi \cdot ni)$時間（mi, ni：バケットサイズ）で実行することができる（図4）。即ち，異なるバケットに属するタプルはJOIN属性に関してhash値が異なる為，JOINが行われる可能性は無く，リレーション間のJOINはJOINに関して互いに独立な複数の小さなバケットJOINに還元される。GRACEではこれらバケットは並列に処理され，JOINの演算時間として$O((\Sigma mi \cdot ni)/k)$（k：並列度）が得られる。（後に示す様にGRACEの物理アーキテクチャに於いては，各バケットJOINはバケット内タプルを$O(N)$ハードウェアソータを用いることにより$O(\Sigma mi+ni)$時間で実行出来，結果としてGRACEに於けるJOINの処理時間は，$O(\Sigma (mi+ni)/k) = O((M+N)/k)$となる。）

JOIN演算は通常，中間タスクとして実行される。この場合，Source空間中のタプルは前タスク（子タスク）のClustering処理に於いて当該JOINに対しhashによるClustering処理が施された状態で格納されている。従って，この場合のcluster，Source空間は各々バケット，バケットの集合となる。JOINタスクに於いては，Source空間から流れ出るデータフローは複数本のバケットデータフローの束であり，一つのバケットデータフローに対して一つのFilter処理LUが割当てられる。Filter処理LUに於いては，流れ込んでくるバケットデータフローに対してバケットJOINが実行される。この様にJOINに対するClustering処理は子タスクの実行時に動的に行われる為，二次記憶系に於ける静的Clusteringに対し，これを動的Clustering（Dynamic Clustering）と呼ぶ。重複除去を伴うPROJECTION, DIVISION, GROUP-BY や AGGREGATIONの処理もJOINと同様な方法で実行される。

5．GRACEの物理アーキテクチャ

5.1 ハードウェア構成

GRACEのハードウェア構成を図5に示す。ディスクモジュール（DM）はGRACEの二次記憶系を構成し，ベースリレーションを格納する。DMは，可動ヘッドディスク，Filterプロセッサ，Clusteringプロセッサ等からなり，更に格納されているリレーションに対するインデックスを管理する。リレーションのアクセスに際しては，インデックスを用いて必要なディスクページ（物



P : Processing Module
M : Memory Module
D : Disk Module
T : Transaction Control Module
Q : Query Control Module

Fig. 5 The Hardware Organization of GRACE

理ページ）を決定し，これを読み出すことによりデータフローを生成する。このデータフローは，Filterプロセッサに送られ，SELECTION や重複除去を含まないPROJECTION を行い，更にClusteringプロセッサに送られて次演算に関するClustering処理を施された後，メモリモジュールに流される。メモリモジュール（ＭＭ）はディスクから送られてくるデータフローに対して複数台割当てられ，ステージされてくるデータフローを受取り，複数個の clusterからなるcluster 空間を生成する。その後，個々のＭＭはcluster を順にプロセッシングモジュールに送出する。従って，JOIN等の演算実行に於ける並列度は演算に割付けられたＭＭの台数に等しい。プロセッシングモジュール（ＰＭ）は内部にＯ（Ｎ）ハードウェアマージソータ〔３〕，タプル処理ユニット，Clusteringプロセッサを内蔵し，ＭＭから送られてくるデータフローに沿ったソートを実行し，JOIN等の集合演算を行い，結果タプルに対して次演算に関するClustering 処理を施してこれをＭＭ群に送り返す。トランザクションコントロールモジュール（ＴＣＭ）は各トランザクションに対し，トランザクション中の問合せの生成，駆動，実行，消滅を制御する。問合せコントロールモジュール（ＱＣＭ）は，問合せ中のタスクの生成，駆動，消滅を制御する。作業用ディスクモジュール（ＷＤＭ）は巨大なリレーション処理の際に，メモリモジュール群に対する仮想空間を構成する為に用いられる。これらモジュールは時分割多重チャネル方式の高速リングバスによって結合されている。

5.2 物理データベースの構造

リレーションは，静的クラスタリング技法として新しく開発された多次元クラスタリング〔5，6〕を用いて，リレーション毎にトランスポーズ，及びページ分割を施されてＤＭ内のディスクに格納される。本技法は，SEL-ECTION演算のpredicate の分布，及びリレーションの原空間に於けるタプルの分布に応じてページ分割を行うものであり，従来のアクセス法，更には多次元クラスタリング技法に比較して，大幅に平均アクセスページ数を減少させることが可能である。このページ分割は，属性のdomainの直積空間を再帰的に分割することにより行

われ，これに対応して一つのインデックスが生成される。このページ分割によって得られたページは論理ページであるが，論理ページに属するタプルを集め，これを一つの物理ページとすることにより，論理ページと物理ページは一対一に対応する。

5.3 タスク実行環境とその制御

ユーザから発行されたトランザクションはＴＣＭでコンパイルされ，最適化の後に内部形式に変換される。ＴＣＭはトランザクション中の問合せを順にＱＣＭに送ってその実行を指示する。ＴＣＭはトランザクションレベルの制御モジュールであり，トランザクション中の個々の問合せの起動を行うと共に，トランザクションのアボート，再実行等の制御も行う。ＱＣＭは個々の問合せの制御を司るモジュールであり，ＴＣＭから送られてきた問合せに対し，その木構造に従ってタスクの発火を行い，問合せ中のタスク群のデータフロー制御を行う。物理アーキテクチャに於ける通信資源はリングバスのチャネルである。 GRACEの物理アーキテクチャに於ける実行環境は図６の様になる。即ち GRACEの物理アーキテクチャに於いては一般に複数台のＭＭがSource空間／Sink空間を構成するが，基底タスクに於けるSource空間構成ＬＵはディスク中の物理ページとなり，複数のこれらＬＵとFilter処理ＬＵ，Clustering処理ＬＵが一つのＤＭに一体化されている。また，中間タスクに於けるFilter処理ＬＵ，Clustering 処理ＬＵは一つのＰＭに一体化されている。従って，物理アーキテクチャに於ける各タスクの実行環境は，

基底タスク：
＜ read setに対応する物理ページの集合及びこれらページを格納しているＤＭの集合，
ＭＭの集合，
チャネルの集合 ＞

中間タスク：
＜ ＭＭの集合（Source），
ＰＭの集合，
ＭＭの集合（Sink），
チャネルの集合 ＞

出力タスク：
＜ ＭＭの集合，
write setに対応する物理ページの集合及びこれらページを格納しているＤＭの集合（または出力デバイス），
チャネルの集合 ＞

となる。ＱＣＭは，Source空間を形成するモジュール群からの要請に従い，タスクの種別に従ってこれらモジュールを各々複数台割付けることによって，タスクの実行環境を構成する。次にＱＣＭは各モジュールに対して適当なコマンド群を送出し，更にタスクを構成するモジュール群全てに対してタスクIdを知らせることによってタスクの起動を行う。タスクに割付けられたチャネル群にもタスクIdが与えられており，各モジュールは自タスクIdと同一のタスクIdを持つチャネルのみを用いてデータフロー転送を行うことによりタスク間干渉の無い通信環境が実現されている。

5.4 タスク処理

物理アーキテクチャに於けるSELECTION演算，重複除去を含まないPROJECTIONは，基底タスクとしてＤＭからのデータフロー送出時にデータフローに沿って実行される。一方，JOIN処理は以下の様な方式で実行される。一般に，動的クラスタリングの際のhash値の分散不均一性は避けられず，バケット対応にＭＭを割付けると，個々のＭＭのレベルでのオーバフローが回避不能となる。そこで，各バケットはSink空間を構成するＭＭ群にわたって均等に分散配置する様に転送される。従って，タスクの終了時には，Sink空間を構成する個々のＭＭにはhashによって生成された各バケットの一部（サブバケット）が格納されている。一般にバケットの大きさはかなり小さい為，ＭＭ群はバケットをＰＭの容量程度に集めてこれを次タスクの実行の際にＰＭに送ることを繰り返す。ＭＭ群がＰＭ群へ一度に送出するバケット（cluster）の集合をprocessing cluster と呼ぶ。この転送はＰＭ

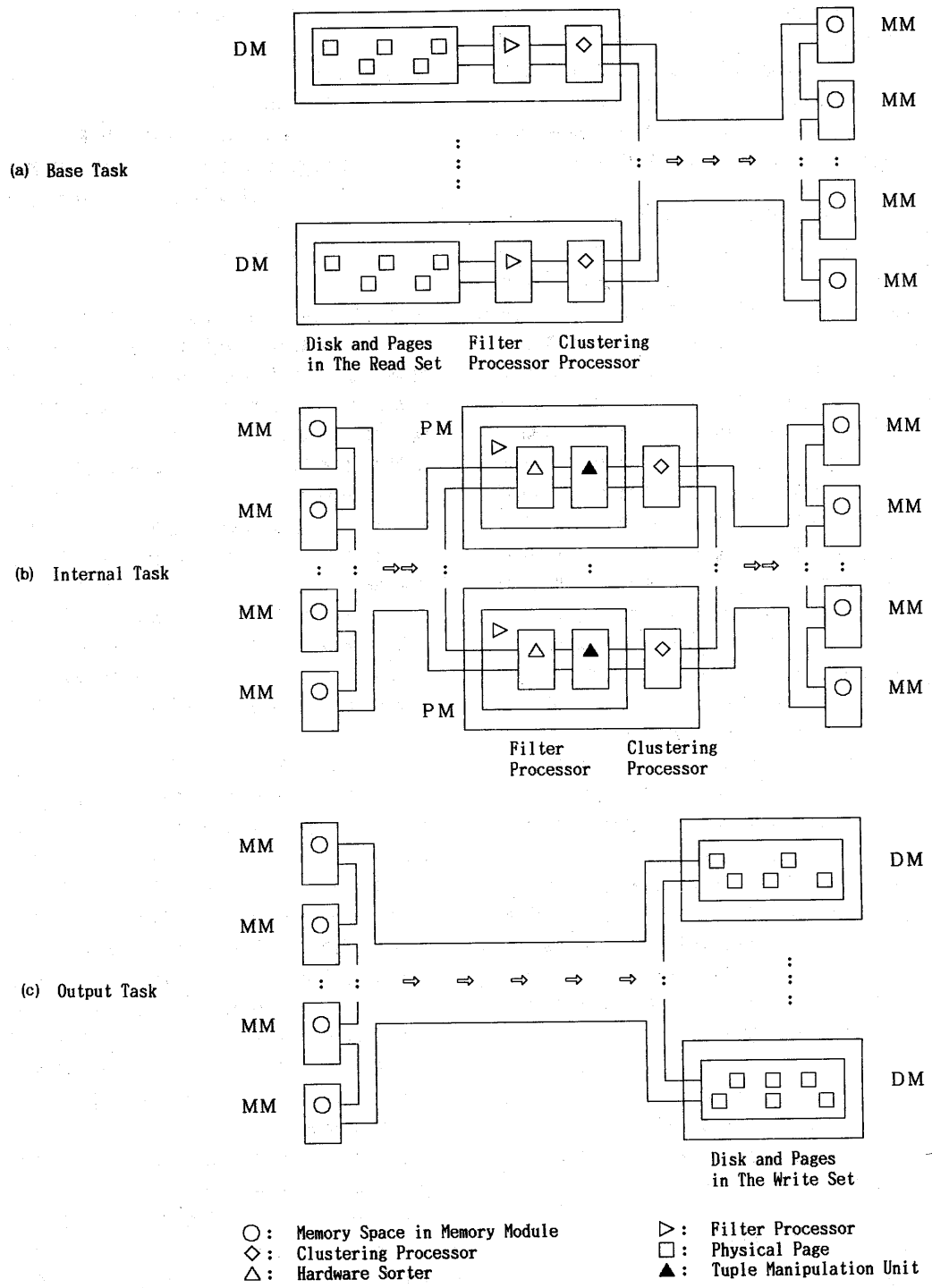


Fig. 6 The Execution Environment of Task in The Physical Architecture of GRACE

群に対してＭＭ群が巡回的にサブバケット群をパイプライン転送することを繰り返すことにより，効率の良く実現される〔１〕。

また，GRACEの物理アーキテクチャの実行環境では，ＰＭ内のハードウェアソータを用いることにより全てのFilter処理はO（Ｎ）時間で実行され，データフローの流れを乱すことなくタスクの処理が実行される。即ち，JOIN処理の際のバケットJOINは，先づバケット内タプルをO（Ｎ）時間でソートし，そのソート出力に対して実際のタプルの結合処理を行うことにより，O（mi＋ni）時間で実行することが可能である。

一方，hash値の不均一性により，巨大なバケットが生成され，ＰＭの記憶容量を溢れる場合がある。この様な場合には，ＰＭをチャネルを介して必要なだけ結合し，ハードウェアソータの線形結合〔４〕を実現して，動的にプロセッシングモジュールの記憶空間を拡大することにより対処する。

5.5 タスク処理の仮想化〔７〕

一般にFilter処理によって生成されるタプル数の正確な予測は不可能であり，Sink空間としてのＭＭ群の最適な割付けは困難である。特にＭＭ群の割付けの失敗によりＭＭ群全体でオーバフローが生じる場合には，ＭＭ群だけでなく，これと作業用ディスクモジュール（ＷＤＭ）群によってSink空間を構成し，Sink空間を仮想的に巨大化することによって対処する。これを空間の仮想化と呼ぶ。Sink空間の仮想化を行った場合のタスク処理は，Clustering処理によって生成されたclusterの集合を，当該演算に関して互いに独立な，割付けられたＭＭ群の総記憶容量程度の大きさの cluster（staging cluster）に分割し，各staging clusterに対して従来のJOIN処理を繰り返すことを基本とする。従って，当該演算のオペランドリレーションは，ＷＤＭ群，ＭＭ群，ＰＭ群の各々の記憶容量に従ってclusterを最小単位とする４つの階層構成を持つことになる（図７）。実際にはClustering処理によって生成されたcluster（Atomic cluster）を集め，ＭＭ群の記憶容量に等しくなる様にstaging clusterを生成し，これをＭＭ群にステージすることを繰り返すことにより仮想化が実現される。即ち，データフロー転送中にＭＭにオーバフローが発生すると，ＭＭ群が適当なバケットを選んでこれをＷＤＭにデステージする。ＷＤＭは内部にディスクキャッシュを持ち，デステージされてくるclusterをデータフローに遅れずに，ディスク内に効率良く格納する。ＤＭ群からＭＭ群へのデータフロー送出が終了した時点でＭＭ群の中には最初のstaging clusterが形成されており，他のclusterはＷＤＭに格納されている。次タスクの実行環境の割付けが完了すると，ＭＭ群中のstaging clusterは直ちに実行され，同時にＷＤＭは次 staging clusterを構成するclusterを選択し，これをＭＭ群に再ステージする。これにより前段のstaging clusterの処理が終了した時点で次のstaging clusterがＭＭ群に配置されていることになる。またＰＭのオーバフローに関しては，特にSink空間が仮想化されている場合には，clusterのデステージの際にその時点で最も大きいclusterをデステージclusterとし，ＷＤＭに転送して，ＷＤＭからの再ステージの際にこのclusterを更に細かくhashし直すことによりこの様なオーバフローを最小限に抑えることができる。

６．おわりに

並列関係データベースマシン GRACEのアーキテクチャについて，計算モデル，論理アーキテクチャ，物理アーキテクチャの３つの側面から体系化を行い，その実行制御について述べた。GRACEはオペランドリレーションが構成する巨大なデータフローを制御単位としており，実行制御に伴う大きなオーバヘッドは軽減されている。現在，実行制御系の詳細化を行っており，今後モジュール割付けのアルゴリズム，インデックス管理等を中心に研究を進めてゆく計画である。

参考文献

〔１〕Kitsuregawa,M.et al,
『Architecture and Performance of Relational Algebra Machine GRACE』,
Proc. of 1984 Int. Conf. on Parallel

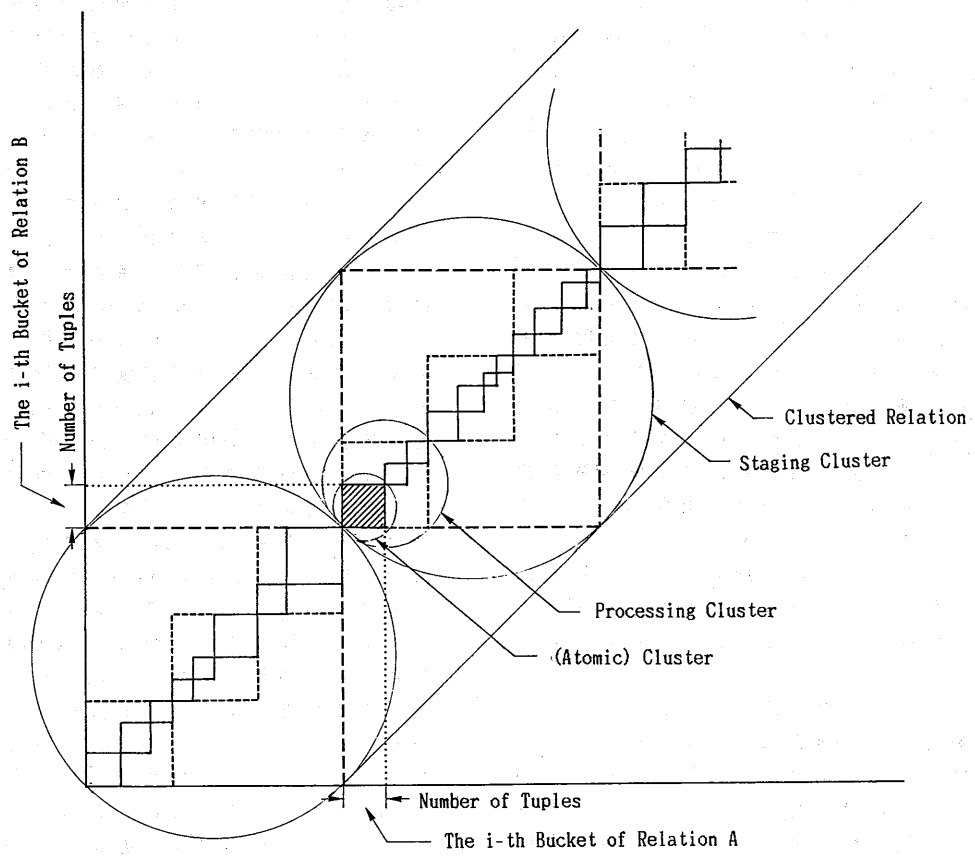


Fig. 7 Join Operation and Cluster Hierarchy

Processing, (1984)
〔2〕 Kitsuregawa,M.et al, 「Application of Hash to Data Base Machine and Its Architecture」, New Generation Comuting, Vol.1 (1983)
〔3〕 喜連川他, 「パイプラインマージソータの構成」, 信学論, D-40, (1982)
〔4〕 喜連川他, 「パイプラインマージソータの構成」, 信学技報, EC81-5, (1981)
〔5〕 伏見, 他, 「多次元クラスタリング技法に基づく GRACE二次記憶系の設計と評価」 信学技報, EC83-49, (1984)
〔6〕 伏見, 他, 「GRACE二次記憶系に於ける拡張多次元クラスタリング技法」, 情報処理学会第27回全国大会, 2K-2, (1983)
〔7〕 伏見, 他, 「GRACEに於けるディスクモジュール の構成」, 情報処理学会第28回全国大会, 2K-2, (1984)
〔8〕 Boral,H.et al, 「Implementation of The Database Machine DIRECT」, IEEE Trans. on Software Engineering, Vol. SE-8,No.6, Nov. (1982)