多重アクセス形仮想記憶を備えた汎用並列計算機の一構成法†

中川 徹†† 北川 一†† 相磯秀夫†††

本研究の目的は，安価，かつ，高速な汎用の並列計算機を構成する一方式論を確立することである．ハードウェア的には，従来の高速バスとキャッシュを用いたプロセッサ結合機構に比較して，より単純で安価な汎用の密結合機構を提示し，ソフトウェア的には，数十台のプロセッサに共通の単一仮想空間を統一的に提供するためのシステム構築法を提示するものである．本報告では，技術蓄積の多い従来のソフトウェア構築手法をそのまま適用できる並列処理システムを実現することを目標にして，従来になかったマルチマイクロプロセッサ構成の汎用並列計算機 (GP³) を提案し，その構成法と基本設計について論じた．具体的には，プロセッサ間の新しい密結合機構として，MVM 結合方式と2段階の番地変換方式を提案し，これらの仮想化と記憶保護に関する基本機構の詳細設計，および，その有効性について論じた．その結果，量産部品を用いた MVM 結合機構と 64 台の物理プロセッサによって実現された単一の仮想空間上で，320 プロセスが各々16組のプロセス間通信を行った場合でも，提案するプロトタイプの GP³ は 51MB/秒の実効転送速度を達成でき，汎用に耐えうる機能と性能を持つことが判明した．

1. まえがき

スーパ・コンピュータを始めとする超高速の並列計算機[1),2)]が実用に供され，また，汎用目的のマルチプロセッサ・システムも商用へ移行している．すなわち，汎用大形機やスーパ・ミニコンにおける CPU の双頭・複頭化に始まり，Elxsi 社のシステム 6400 (OS : Embos)[3)] にみられるような，本格的マルチプロセッサ構成を取る汎用システムも発表されている．また，最近では，32 bit のマイクロプロセッサ (以下，μ-cpu) を用いたマルチ μ-cpu システムとして，米国 Sequent Computer 社の Balance 8000, 同 21000 (OS : DYNIX), および，Encore Computer 社の Multimax (OS : U-Max 4.2) 等も商用になりつつある．特に，1985 年のコンピュータ・アーキテクチャ国際会議においては，商用のマルチプロセッサに関するセッション[10)~12)] が開かれ，実用化への道を歩み始めている．

上記市販システムの多くは，複数の CPU とメモリ等のモジュールを高速度のバス (27～100 MB/秒) を用いて密結合し，CPU モジュールそれぞれにもキャッシュを用意するなどしてバスのビジー率を低下させ，主記憶を共有する際のオーバヘッドを軽減する方式を採用している．しかしながら，キャッシュを用いた結合方式においては，共有データの更新処理を迅速に行うために，高速，かつ，複雑なハードウェア機構を必要とし，さらに，複数のキャッシュ内に散在するデータの一貫性を保証するプロトコルの開発[13)]が実現上の重要な鍵となっている．この結果，複数 CPU の性能を自動的に引出せる制御ソフトウェアや言語処理系の作成も困難さを増している．

そこで，キャッシュを用いた高速バス結合方式に比較して，より単純な機構を持つ，安価で高速なマルチ μ-cpu システムを実現することが望まれる．その際に，キャッシュの一貫性に起因する最適化問題を根絶できる汎用の高速結合機構，および，システム全体を動的に管理・運用する汎用 OS の実現などが解決すべき問題となる．ここで，問題解決のアプローチには以下に示す2つの立場が考えられる．

① 従来技術を採用したマルチ μ-cpu システムに最適な分散処理 OS を研究し，新規開発する．すなわち，開発済みの並列処理ハードウェアに基づいて，その能力を引出せる並列処理ソフトウェア体系 (OS) を設計・実装する立場である．

② 技術蓄積の多い単一 CPU 用 OS の諸機能 (できれば，タスク・スケジューラとタスク同期操作部のみを改造した従来 OS) を直接実行できるマルチ μ-cpu 構成の単一仮想システムを実現する．これは，既存の UNIX などの単一 CPU 用マルチタスク・システムを直接実行するために，並列処理ハードウェアを設計・実装する立場である．

本研究では，後者②の立場で，安価，かつ，高速な

† A Preliminary Architecture for a General Purpose Parallel Processor with a Multi-access Virtual Memory by TOHRU NAKAGAWA, HAJIME KITAGAWA (Faculty of Engineering, Toyota Technological Institute) and HIDEO AISO (Faculty of Science and Technology, Keio University).

†† 豊田工業大学工学部制御情報工学科
††† 慶応義塾大学理工学部電気工学科

汎用の並列計算機を構成する方式論を確立することを目的とし，本論文では，新しいプロセッサ間の結合方式を提案し，その詳細設計と効果について論じる．

2. 並列計算機—KDSS—と汎用化機構

本研究における発想の出発点となっている KDSS (Keio Discrete System Simulator) は，本来，離散系シミュレーション専用機である．すなわち，慶応大学における Approach (A parallel processing architecture) グループが，シミュレーションの並列処理に関する研究[4)~6)]を行っている過程で，離散系の専用機として設計したものが KDSS であり，PU (Processing Unit) 間の密結合機構を安価に実現する目的で，多重読出し形共有記憶 (MRPM: Multi-Read Packet Memory) 結合方式を考案・設計し[6)]，試作一号機の KDSS-I として実装した[7)] ことが MRPM アーキテクチャの始まりであった．

一方，市販のマルチプロセッサ・システム[10)~12)]に代表されるように，世界の主流はキャッシュを複数用いた高速バス結合方式となりつつある．しかしながら，MRPM 結合方式においては，文献 12) にて指摘されている接続部品数の増加問題を，書込み専用のバスを介した全 PU への放送機構により，解決することに成功しており，その結果，完全に並列読出し可能なマルチポート・メモリ結合を安価に実現できることを示せた．その後に，MRPM 方式の改良と拡張が，どちらかというと，日本国内各所で行われ，例えば，神戸大学/岡山大学/徳島大学の BC (Broadcast) 結合や慶応大学の RSM (Receiver Selectable Multicast) 結合[8)] 等の方式が実現され，問題向きの専用機を設計する際に有効であることが報告されている．

2.1 並列処理の単位と方式

並列処理の粒度 (Granularity)[12)] により，PU へ割当てる並列処理の単位を以下のように分類する．

① ジョブ（または，プログラム）
② 大タスク[4)]（または，手続き）
③ 小タスク[4)]（または，基本関数）
④ 機械命令
⑤ 機械命令内（または，マイクロ命令）

本研究においては，従来のソフトウェア構築手法を再利用することを目標にし，上述した①～③までの範囲に存在する並列性について，並列処理を行う．この場合，従来のソフトウェアそのものは，逐次処理を前提としているために，実際には，入出力などの資源待ちが発生する時点でしかタスクの分割が行えず，並列処理の粒度としては比較的大きい．しかしながら，最近のモジュール化手法によって，小さなタスク・プロセスを連結して大きなジョブ・プロセスを形成させる傾向が強くなってきている．具体的には，UNIX 上で並行に走行する多数のプロセス群を実例としてあげることができる．加えて，UNIX におけるパイプ[9)]の概念は，①～③において適用可能なプロセス間通信の典型であり，実際，パイプを通るデータは主記憶上で引渡され，高速化が可能である．したがって，PU が①～③のどれかの単位を同程度に受け持つ場合，パイプに類するプロセス間通信機能を高速に提供し，かつ，並列処理の単位数に準じた PU を用意することで，線形倍の高速化が可能になる．さらに，③の小タスク単位の並列処理を記述できるデータ駆動形言語[5),6)]を利用者に提供することで，より粒度の細かい高度な並列処理も行える汎用計算機を実現できる．

2.2 汎用化における問題点と解決策

汎用目的の並列計算機を実現する際に，オリジナルな MRPM 結合，および，その改良形の方式をそのまま用いると，以下の点に問題が発生する．

① 共有する空間は PU 間の通信領域としての性格が強く，このことを意識してソフトウェアを設計し，実装する必要がある．
② プログラムの実行場所が，通常，PU 内のローカル・メモリであり，OS が PU にプロセスを配分する際のオーバヘッドが大きい．

そこで，共有記憶空間を，現在の通信領域としての実番地方式から，本来の共有領域としての論理番地方式に変更して空間を拡張し，さらに，共有データの書込み時に発生していた多重読出しの禁止時間をなくすことにより，多重アクセス可能な共有記憶上に仮想記憶空間を実現することを提案する．すなわち，各 PU が所持するローカル・メモリから不連続な空間を構成するのではなく，共有記憶自身を大容量の主記憶 (4G バイト：256^4 バイト) と考え，そのアクセス機構を仮想記憶と多重アクセスの両機構によって実現し，汎用化と高速化を両立させる方式を考案した．

2.3 MVM 結合機構

慶応大学の RSM 方式では，小規模ながらも共有データ領域の仮想化が行われ，PU が共有する空間 (64 kW/32 bits) を各 PU に附属の実記憶 (2 kW) に写像している．しかしながら，RSM は実行時の通信バッファとして設計されたために容量も小さく，大きなプ

ログラムの配布などに用いると，ロード完了までにミリ秒台の待ちが生じ，汎用の結合機構として性能と規模の両面において不十分である．そこで，図1に示すMVM (Multi-access Virtual Memory) 結合機構を新たに設計した．図1が KDSS-I と異なる点は，主に以下の3点である．

① 従来，PKM と呼んでいた各 PU に附属する実アドレスの共有記憶要素に，ページ (256 バイト) 単位の番地変換要素 (MAP) を追加した．すなわち，ある PU から結合機構の書込み専用バス (MVM-BUS) へ送出された論理番地を，各 PU に附属した共有記憶要素 (FRM) の実番地へ，それぞれページ単位で変換する．この時，該当ページを共有している PU に附属の番地変換要素のみが作動し，書込み専用バス上の論理番地をそれぞれの共有記憶要素における実番地に変換し，共有データの書込み (更新) を行う．一方，該当ページを共有しない PU に附属の番地変換要素は，ページ不在を検出後，書込み要求を却下する．

② 番地変換要素と共有記憶要素の制御を分離し，アクセス制御要素 (ACn) により，両機構を並列に動作させている．このパイプライン的な制御によって，単体でも2ポートの記憶装置として動作し，他の PUs からの共有データ書込み時にも，並行して PUn 自身の読出し操作を行える．

③ UNIX の実現などにおける実用性を考慮して，KDSS-I において 64k バイトであったローカル・メモリを 2M バイトへ，また，32k バイトの共有記憶要素を 2M バイトへと，その容量を大幅に増加させた．さらに，CPU についても，KDSS-I にて使用したミニコン基板 (東芝 micro 7) の 3〜8 倍程度高速のものを採用した．

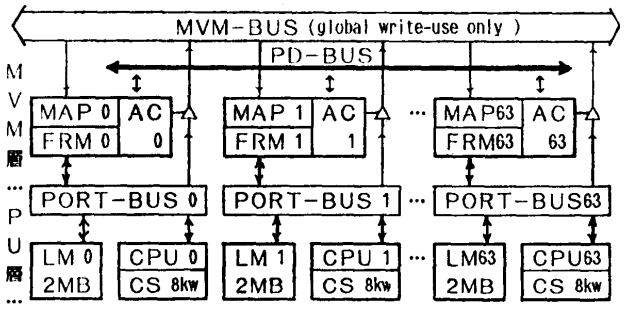

図1 MVM 結合
Fig. 1 Multi-access Virtual Memory arrangement.

以上の改良によって，PUn と対の番地変換要素が，他の PU と共有するデータ (または，テキスト) 部分のみを選択的に書込み専用バスから共有記憶要素上に写像でき，また，各 PU はそれを互いに競合なく読出せる．一方，MVM 空間への書込みにおいては，局所変数と広域 (共有) 変数を取扱うタイミングをずらし，前者を他の PU の影響を全く受けないL相で書込み，後者は他の PU と互いに排他制御するG相で書込むこととした．この時の排他制御は，従来と同様に，アクセス制御要素と優先順位決定バス (PD-BUS) を介して PUn の書込み優先順位を擬似ラウンドロビン[6]によって決定し，データをその論理番地とともに書込み専用バスへ送出 (放送) することで，広域変数を1語/400 ns の速度にて書込める．

従来の MRPM 結合方式では，MRPM をアクセスしているすべての PU が排他制御の対象となり，たとえデータを互いに共有し合う関係にない PU であっても，MRPM を読出している限りは，書込み権を得た他の PU によって，最低1サイクルは待たされる結果となっていた．しかしながら，MVM 結合の場合は，マルチキャスト形のアクセス制御方式，および，MVM を構成する共有記憶要素と番地変換要素のオーバラップ動作によって，同時に広域変数の書込みを要求した PU のみが互いに排他制御されることとなり，従来，共有記憶装置をアクセスしている PU 間で発生していた不要な競合を排除することができた．

3. 共有仮想記憶機構

本章では，3レベルの番地空間からなる仮想記憶方式を提案する．すなわち，最下位の物理空間 (各 PU に存在する共有記憶要素の空間) 上に，論理的な共有仮想空間 (MVM 空間) を構成し，さらに，その上位に全資源を登録・管理する単一の論理空間を構築する方式を考案し，設計した．以下，その詳細を述べる．

3.1 仮想記憶の方式と番地付

最下位では，IBM のシステム/38 と同様の考えに基づき，全資源 (ハード，ソフト) を単一の論理空間に配置する．ここで，特に，この論理空間を単一仮想資源空間 (最大 256^7 バイト) と呼ぶことにする．その時の番地付の様子を図2の上部に UVR (Unified Virtual Resource) 空間として示す．空間の最小構成要素はページ (256 バイト) であり，それが最大 256 個集まって1セグメ

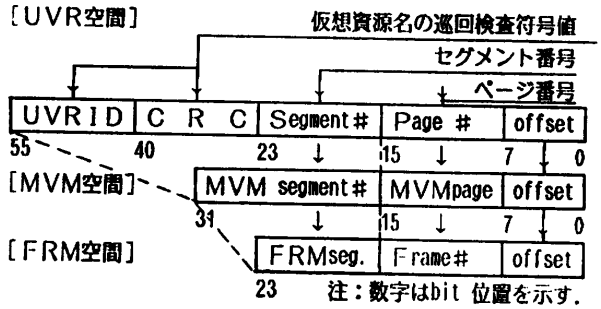

図 2 単一仮想資源空間における番地変換
Fig. 2 Address conversion schema in a space for Unified Virtual Resources.

ント（最大 64 k バイト）を構成する．さらに，セグメントが最大 256 個で 1 レコード（最大 16 M バイト）を構成し，16 M バイトを越えるファイルは連続した複数のレコードに記録される．

図 2 における仮想資源名は従来のファイル，入出力装置，プロセス等の論理名に対応し，また，仮想資源名の CRC コードをそのまま UVR 空間の番地上位部に使用している．この結果，従来のファイルのみならず，入出力ドライバやバッファ，もしくはプロセス等も，そのまま仮想資源空間にランダム（最短距離で 16 M バイトごと）にハッシングされ，これら資源へのアクセス方法を統一化できるとともに，その処理速度も，以下に述べる資源の仮想空間への常駐化と MVM 機構によって，主記憶並に高速化できることとなった．

従来の仮想記憶システムに比較して，本論文における UVR と MVM の両仮想空間は常駐形式の仮想空間である点が基本的に異なる．すなわち，従来の仮想記憶システムに見られるような一時的な実行時のみの仮想空間とは異なり，登録された仮想資源が常時 UVR 空間上に存在し，また，特別に指定をしない限り，システムによって割り当てられた MVM 空間にも常駐する．その結果，仮想資源へのアクセスを高速に行えるとともに，比較的低速，かつ，小容量のディスク装置しか装備していない場合においても，実記憶（共有記憶要素とディスク，および，ローカル・メモリ）の容量が許す限りのファイルを保有することが可能である．

3.2 UVR-MVM 番地変換方式

UVR 番地を MVM 番地（最大 256^4 バイト：4 G バイト）へと変換するためのテーブルは，MVM 空間前方のシステム領域中に存在し，現時点で登録，または，使用中の仮想資源に関する情報を保持する．本テーブルは，既存 OS におけるファイル管理台帳，および，プロセス管理テーブルの機能を同時に有し，UVR 空間から MVM 空間へのセグメンテーションによる番地変換をする際に用いられる．このテーブルを UNIX 流にインデックス・リスト (i-list)[9] と呼び，本来の UNIX における概念を拡張した．i-list には，仮想資源に関する定義情報を格納するための i-node が多数集められ，各 i-node には，順番に i-番号 (i-number) が付けられている．1 つの i-node 内で定義される仮想資源の情報としては，以下のものがある．

① 仮想資源の種類（ファイル，入出力ドライバ，プロセス/イメージ等）
② 保護ビット（UNIX の保護ビットと同等）
③ リンク数（UNIX のリンク数と同等）
④ 識別子（利用者 ID，グループ ID，CRC）
⑤ 最終アクセス日時
⑥ 最終変更日時
⑦ 作成日時
⑧ 構成セグメント各々の先頭番地（MVM segment #，MVM page #）
⑨ 構成セグメント各々の大きさ（ページ総数）
⑩ スケジューリング/スワッピング用情報

以下，UVR 番地を MVM 番地に変換する手順を述べる．仮想資源名が持つ CRC 値から直接 i-番号を算出し，その番号から始まる i-node 内の識別子（前述の④）と UVR 番地上位部 (UVRID と CRC) を比較し，仮想資源に対応する i-node を検索する．検索した i-node 内には仮想資源を構成するセグメント，例えば，テキスト（コード），必要に応じて，利用者データやシステム・データ等の各種セグメントの先頭番地が 3 バイトの大きさで格納されている．最初の 2 バイトが MVM 空間におけるセグメント番号で，残りの 1 バイトはページ番号であり，これらを MVM 空間におけるベース番地として利用する．また，i-node 内の次の 2 バイトには，当該セグメントの長さが保持され，アクセス範囲の検査に用いる．以上の 5 バイトからなるセグメンテーション情報は各 PU 内のベース・レジスタに保持され，各々の制御機構（マイクロプログラム等）が実行時に利用する．

3.3 MVM-FRM 番地変換方式

MVM 番地の上位 3 ビットは，文献 6) にて既に報告した共有記憶内論理演算（一種のロジック・イン・メモリ）機能を指定するために用い，**表 1** のように，最上位 (A 31) ビットが “0” で番地変換を伴う仮想番

表 1 MVM 演算機能の内訳
Table 1 Specification of MVM operation.

MVM address		MVM operation	
A31～29	A28～0	read	write
0××	×	Read	Write
100	×	Read	Write
101	×	Direct Read	Direct Write
110	×	Test & Inc	AND
111	×	Test & Dec	OR

地へのアクセスであることを指定し，同ビットが“1”で KDSS-I との互換モードを示す．特に，AND や OR の操作は，共有データの授受における同期操作，例えば，データ駆動形のパケット転送における同期フラグのオン/オフ操作[7]等に頻繁に用いる．

MVM 番地を各共有記憶要素の番地（最大 256^3 バイト：16 M バイト）へと変換するテーブルは，PU と対の番地変換要素内にそれぞれ存在し，各 PU において登録・使用中の MVM ページとフレームに関する情報を保持する．**図 3** に示すように，各 PU ごとにセグメントとページ用の 2 種類のテーブルが存在し，2 段階の番地変換によって論理的な MVM 番地を各々の物理的な共有記憶要素番地へ写像する．これらのテーブルすべては，MVM 空間に配置された実記憶領域に存在し，各 PU ごとに通常の仮想記憶機構と類似のデマンド・ページング方式の番地変換に用いられる．

MVM 結合機構における仮想化機構と，従来の単一 CPU，もしくは，単一アクセス形の主記憶下における仮想化機構とは，以下に示す点において，設計上の違いが生じた．1 つ目の相違は変換テーブルの存在場所であり，2 つ目の相違は共有データのスワッピングに関してである．結果として，前者については，セグメントとページの 2 つの変換テーブルをハードウェアで各番地変換要素内に用意し，後者については，ページ・テーブル内の制御ビットに共有ビット（pc ビット 13：**表 2**）を追加して実行時のスワップアウトを防止した．以下にその理由を述べる．

並列計算機においては，共有領域に対する各 PU のアクセス時間を保証するために，PU の最大接続数に比例して，共有記憶の番地変換と接続バスのサイクル時間を等価的に短縮する必要がある．さらに，共有領域への書込み時にページ不在状態が発生すると，以後，そのページへ書込みを要求する PU がすべて待ち状態に陥ってしまうため，この問題も回避する必要がある．

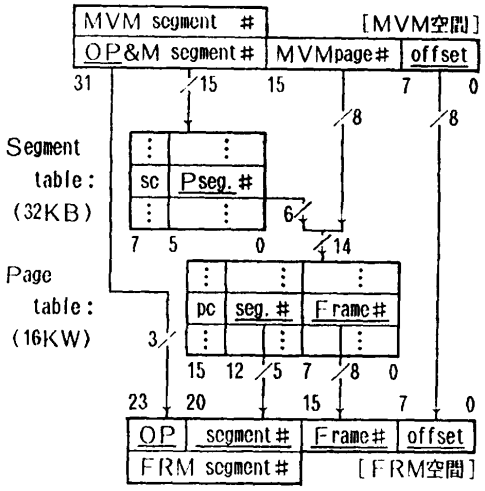

図 3 MVM-FRM 番地変換機構
Fig. 3 Address conversion mechanism for a PU from MVM address to FRM address.

表 2 ページ制御—PC—ビットの内訳
Table 2 Specification of Page Control bits.

名 称	位 置	役 割
Referred	15	仮想記憶管理用で，過去に参照されたことを示す．
Changed	14	仮想記憶管理用で，過去に変更されたことを示す．
Shared	13	ページ入替えの対象にならない共有領域を示す．

そこで，MVM 結合機構の設計において，①近年，大容量化と低価格化が進められた CMOS スタティック・メモリ（アクセス時間 70～100 ns）を変換テーブルに用い，②そのテーブル 2 段から構成される番地変換要素をサイクル時間 200 ns の共有記憶要素ごとに用意し，③番地変換要素と共有記憶要素をパイプライン的に動作させ，④共有領域で実行時に書替えられる領域とそうでない領域をページ単位で区別するビットを用意し，前者をページ入替え不可能領域とする等の工夫を行い，前述の問題を解決するとともに安価な量産部品のみを用いて実効的な高速化を図った．

3.4 仮想記憶の保護と管理の機構

UVR 空間における記憶保護と管理の役割を受持つ機構は，PU 内の MVM 番地生成機構（主にマイクロプログラム）である．一方，MVM 空間における記憶保護と仮想化の両機構は，各々，図 3 の番地変換用のセグメント・テーブルとページ・テーブル内の各ビットにて制御される．

表 3 セグメント制御—SC—ビットの内訳
Table 3 Specification of Segment Control bits.

名　称	位置	役　割
User available	7	記憶保護用で，ユーザが利用できる領域を示す．
Write enable	6	記憶保護用で，書込み可能な領域を示す．

表 4 SC ビットによるアクセス制御
Table 4 Access control by SC bits.

UW	スーパバイザ			ユーザ		
	実行	読出し	書込み	実行	読出し	書込み
00	○	○	×	×	×	×
01	○	○	○	×	×	×
10	○	○	○	○	○	×
11	○	○	○	○	○	○

×：不可，○：可

セグメント・テーブルは，MVM 空間のセグメント番号 (M segment #) を中間的な番号 (P seg. #) に変換すると同時に，表 3 に示した上位 2 ビットのセグメント制御ビット (sc) によって，仮想記憶空間での保護機能を指定する．具体的には，表 4 に示すような実行，読出し，書込み時のアクセス制御機能をシステムとユーザの 2 つのモードにおいて提供する．

一方のページ・テーブルは，MVM 空間のページ番号 (MVM page #) と前出の中間番号から，物理的な共有記憶要素空間のセグメント番号 (segment #) とフレーム番号 (Frame #) へ変換すると同時に，表 2 に示したページ制御ビット (pc) の上位 2 ビットを，アクセス種別に応じてオンにする．すなわち，アクセスが発生したページの参照ビット (pc 15) をオンとし，特に書込みが行われた場合に限って変更ビット (pc 14) をオンにする．これらの情報は，共有ページを指定するビット (pc 13) とともに，ページ入替えを行うための LRU アルゴリズム実現において利用される．

4. MVM 結合方式の有効性

4.1 MVM 結合機構の採用効果

並列処理計算機の構成において，MVM 結合機構を採用した時の効果として，以下のことがいえる．

〔効果 1〕 複数 PU 間で共有する空間が主記憶内の任意領域であり，アドレス・ポインタ等の書替えだけで共有データの転送を行え，その高速化が容易であるとともに，プロセス生成時のコピー操作も削減できる．

〔効果 2〕 G と L の 2 つの相を番地変換要素と共有記憶要素のパイプライン動作で実現した結果，MVM 上の共有（広域）変数への書込みが，他の PU における MVM 上の局所変数への書込み，または，MVM からの読出し操作と競合することが全くない．

〔効果 3〕 MVM 内の論理演算機能により，1 メモリ・アクセスで共有領域中のフラグ類を操作でき，PU 間のデータ転送に伴う同期操作を高速化できる．

以上の効果に加え，MVM 結合機構上に 3 レベルの番地空間からなる共有仮想記憶機構を実現する場合，以下の効果を期待できる．

〔効果 4〕 UVR と MVM の両仮想空間が共有記憶領域に常駐しており，UVR 上のファイルやプロセス等の仮想資源へのアクセスを高速に行える．

〔効果 5〕 UVR 上の仮想資源は，i-list に登録した i-node 内の保護ビットと，所有者やグループ ID 等との組合せによって，不当なアクセスから保護され，さらに MVM 機構内の変換テーブルによって，セグメントごとにハード的保護を受けられる．

これらをより有効なものとするために，MVM 結合機構における PU（したがって，プロセス）間の結合能力について議論する必要がある．以下，MVM 結合方式が，従来の単一バス共有記憶結合方式や MRPM 結合方式に比較して，格段に優れていることを示す．

4.2 MVM 結合機構の性能と価格

PU 間の結合能力を評価するモデルとして，文献 7) に示したデータ駆動形の並列処理モデルを用いる．すなわち，Qs 台の論理的な QSV プロセッサを Pu 台の物理プロセッサへ均等に割付け，その時の QSV プロセッサ間におけるデータ転送時間を，単一バス・共有記憶結合，MRPM結合，改良形MRPM結合，MVM 結合の 4 方式各々について求め，比較する．

条件として，多重読出し機能を保有していない単一バス・共有記憶結合方式が極端に悪く評価されることを避けるために，各 QSV プロセッサの処理プログラムは各 PU 内のローカル・メモリに存在するものと仮定する．また，データを長さ B バイトのパケットとし，その転送に用いる同期フラグも共有記憶上に 2 ビット用意する[7]．以下，1 台の QSV プロセッサから共有記憶への書込みに要する時間の総和を T_w，同様に読出しに要する時間の総和を T_r とし，すべての QSV プロセッサが一通りパケットを転送するために必要な時

間を全転送時間として求める．

ただし，t_c：共有記憶のサイクルタイム（400 ns）

I：QSV 入力端子数

O：QSV 出力端子数

B：パケット長（バイト数）

ph：同期フラグのヒット率，$ph \leqq 1$

Qs：QSV プロセッサ数

Pu：物理プロセッサ数

〔従来方式[7]〕

単一バス・共有記憶結合と MRPM 結合の両方式に共通な T_w, T_r, および，両方式におけるそれぞれの全転送時間 T_s と T_m を以下に示す．

$$T_w = t_c \cdot \{L + 2 \cdot (I + M)\}$$

$$T_r = t_c \cdot \{L \cdot I + (I + M)/ph\}$$

ただし，L は1パケットの総語数で，M は QSV プロセッサにおける出力側フラグの総語数である．

$$L = 1 + Int\{(B-1)/2\}$$

$$M = 1 + Int\{(O-1)/8\}$$

Int：整数化の関数とする．

全転送時間 T_s と T_m は，

$$T_s = Qs \cdot (T_w + T_r)$$

$$T_m = Qs \cdot T_w + T_r \cdot mp$$

ただし，$mp = 1 + Int\{(Qs-1)/Pu\}$

〔改良方式〕

従来方式の T_w において，係数2が式中に存在したが，これは，共有領域上の AND や OR といった演算を2サイクルにて実行していたことによるものである．そこで，MRPM の改良方式として，読出しと修飾，および，書込みを単一サイクル（t_c）内に行えるダイナミック RAM を採用し，その際の書込み時間 T_{wx} を求める．そして，改良形 MRPM 結合方式の全転送時間 T_{mn} と，本論文にて設計した MVM 結合方式の全転送時間 T_{mv} を以下に示す．

$$T_{wx} = t_c \cdot \{L + I + M\}$$

全転送時間は，

$$T_{mn} = Qs \cdot T_{wx} + T_r \cdot mp$$

$$T_{mv} = \mathrm{Max}\{Qs \cdot T_{wx}, (T_{wx} + T_r) \cdot mp\}$$

ただし，M, L, T_r, mp は従来方式に等しい．

図4に，$I = O = 16$ ポート，$B = 64$ バイト，$Qs = 320$ プロセスの問題（例えば TSS プロセスが320個など）を仮定し，$Pu = 64$ 台とした時の同期フラグ・アクセス回数（$1/ph$）に対する全転送時間の変化を，T_s, T_m, T_{mn}, および，T_{mv} 各々について示す．

このグラフと式の計算値から次のことがいえる．

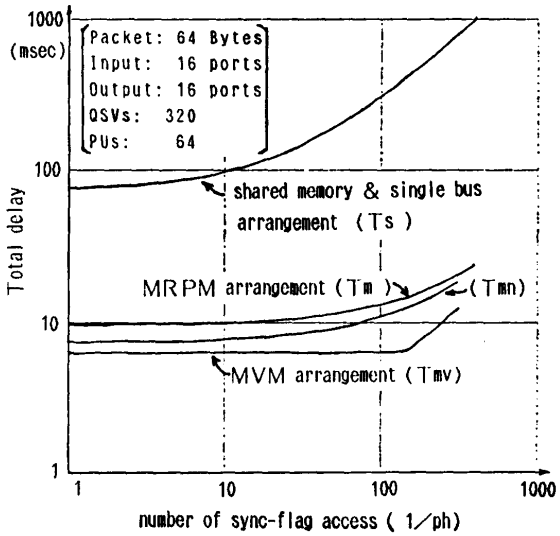

図 4　同期フラグ・アクセス回数と全転送時間の関係
Fig. 4　Total delay for packet communication vs. access rate of synchronization flag.

① 多重読出し方式の全転送時間は，T_m, T_{mn}, T_{mv} のいずれの場合も，同期フラグのアクセス回数によって，大幅に変化しない．

② 単一バス・共有記憶結合の全転送時間 T_s は，ヒット率 ph の低下に伴って大幅に増加し，例えば，$1/ph$ が200の時に約 533 ms となり，多重読出し方式よりも 32〜64 倍程度遅くなる．

③ MRPM 結合に対し，改良形の性能比は，最良で1.3で，ph の低下に伴って1に収束する．

④ MRPM 結合に対し，MVM 結合の性能比は，ph の常用域（すなわち，全 QSV プロセッサが書込みを完了するまでの総時間と，1台の PU が読出しに必要とする総時間とが釣合う点までの領域）で 1.5〜2.3 倍まで向上する．

⑤ MVM 結合の全転送時間 T_{mv} は，ph の常用域まで一定で，約 6.4 ms と非常に短く，この時の総転送量は 320×16×64 バイトであることから，51 MB/秒の実効転送速度が得られている．

加えて，各 QSV プロセッサの処理プログラムをローカル・メモリに配置したことから，以下のことがいえる．

⑥ 広域変数のみならず，処理プログラム等も共有記憶に配置して実行させた場合，MVM 結合方式は，他方式よりも格段に優れた性能を示し，広域変数の総書込み速度が 5 MB/秒以下の範囲では，t_c（400 ns）以内に 64 台の PU が各々

2バイトをフェッチできることから，320 MB/秒の主記憶と等価になる．

MVM 結合機構における主要部品は各 PU ごとに用意する容量 2 MB のダイナミック RAM，容量 64 kB の CMOS スタティック RAM，および，市販ゲート IC 等である．今後，VLSI 技術の成果である 1 Mbit のダイナミック RAM と 256 kbit CMOS スタティック RAM の普及等により，これらの価格は継続的に低下するものと考えられ，MVM 結合機構を実現するための経済的環境が整いつつある．

5. むすび

本報告では，技術蓄積の多い従来のソフトウェア構築手法をそのまま適用できることを目標にして，従来にないマルチ μ-cpu 構成の汎用並列計算機 (GP^3) を考案し，その構成法と基本設計を示した．具体的には，PU 間の新しい密結合機構として，MVM 結合方式と2段階の番地変換方式を提案し，これらにおける仮想化と記憶保護に関する基本機構の詳細設計，および，その有効性について論じた．そして，性能について考察したところ，汎用に耐える十分な密結合性を示し，本研究の目標であった安価，かつ，高速な汎用の並列計算機を構成する一方式論を提示できたといえる．

現在，GP^3 は机上設計から実装段階へ移行しており，それとともに，より詳細な性能評価と GP^3 のソフトウェア開発を支援する目的で，KDSS-I 上に MVM 機構のエミュレータを作成している．

また，本論文の投稿直後に，本文で提案した MVM 結合方式に類した研究も発表[14]されるなどしており，今後，多重アクセス形の仮想記憶を用いた汎用システムに関し，各方面からの幅広い研究が重要となろう．

謝辞 本研究を進めるに際して，貴重な御意見を頂いた慶応義塾大学の天野英晴博士に感謝する．また，PU の設計に関し，御支援を頂いた東京芝浦電気(株)の木下常雄氏を始めとする設計担当諸氏，ならびに，JBA の関係諸氏に深謝する．本研究は昭和60年度文部省科学研究費の助成を受けて行った研究の一部である．

参考文献

1) Hockney, R. W. and Jesshope, C. R.: *Parallel Computers: Architecture, Programming and Algorithms*, Adam Hilger Ltd., Bristol (1981).
2) Multiprocessing Technology, *IEEE Comput.*, Vol. 18, No. 6 (1985).
3) Olson, R.: Parallel Processing in a Message-Based Operating System, *IEEE Softw.*, Vol. 2, No. 4, pp. 39-49 (1985).
4) Yura, E., Yoshikawa, R., Nara, Y., Kimura, T. and Aiso, H.: An Approach to Parallel Processing for Continuous Dynamic System Simulation with Microprocessors, *Proc. of 2nd USA-JAPAN Computer Conference*, pp. 172-177 (1975).
5) Yoshikawa, R., Kimura, T., Nara, Y. and Aiso, H.: A Multi-microprocessor Approach to High-speed and Low-cost Continuous-system Simulation, *AFIPS Conference Proceedings*, Vol. 46, pp. 241-246 (June 1977).
6) Nakagawa, T., Kumata, I., Hasegawa, T., Matsumoto, T., Abe, K., Kobayashi, N. and Aiso, H.: A Multi-microprocessor Approach to Discrete System Simulation, *Proc. of 20th COMPCON*, pp. 350-355 (1980).
7) 中川 徹，小林信裕，相磯秀夫：データ駆動型離散系シミュレータ KDSS-I，電子通信学会論文誌，Vol. J 65-D, No. 3, pp. 386-393 (1982).
8) 天野英晴，工藤知宏，若月哲郎，斎藤千鶴子，相磯秀夫：一般疎行列専用並列計算機 $(SM)^2$-II のプロトタイプ，第 30 回情報処理学会全国大会講演論文集，pp. 121-122 (1985).
9) Thompson, K. and Ritchie, D. M.: UNIX Programmer's Manual. Seventh Edition. Bell Laboratories, Murray Hill, New Jersey.
10) Sanguinetti, J. and Kumar, B.: Performance of a Message-based Multi-processor, *12th Ann. Intl. Conf. on Computer Architecture*, pp. 424-425 (1985).
11) Matelan, N.: The FLEX/32 Multicomputer, *12th Ann. Intl. Conf. on Computer Architecture*, pp. 209-213 (1985).
12) Rodgers, D. P.: Improvements in Multiprocessor System Design, *12th Ann. Intl. Conf. on Computer Architecture*, pp. 225-231 (1985).
13) Katz, R. H., Eggers, S. J., Wood, D. A., Perkins, C. L. and Sheldon, R. G.: Implementing a Cache Consistency Protocol, *12th Ann. Intl. Conf. on Computer Architecture*, pp. 276-283 (1985).
14) 陣崎 明，八星禮剛：ブロードキャストネットワークによる分散型単一階層仮想記憶システム，電子通信学会技術研究報告，CPSY 86-20, pp. 1-11 (1986).

(昭和61年5月26日受付)
(昭和62年3月25日採録)

中川　徹（正会員）

昭和27年生．昭和51年慶應義塾大学工学部電気工学科卒業，昭和56年同大学院博士課程修了．同年豊田工業大学に勤務，昭和58年同大学工学部講師，現在，同助教授．工学博士．主に，並列処理・分散処理のシステム・アーキテクチャに関する研究に従事し，連続系／離散系などの並列シミュレーション，および，異機種間高速 LAN の方式設計に興味を持つ．また，個人向けの計算機における母国語プログラミングにも興味を持つ．ACM，電子情報通信学会，日本シミュレーション学会各会員．

北川　一（正会員）

昭和38年京都大学工学部数理工学科卒業，昭和43年同大学院工学研究科博士課程修了，同年4月京都大学工学部助手，昭和47年4月京都大学大型計算機センター助教授，昭和58年4月より豊田工業大学制御情報工学科教授，現在に至る．京都大学工学博士．主として，システム性能評価，オペレーティング・システム，コンピュータ・ネットワークなどの研究開発に従事．電子情報通信学会会員．

相磯　秀夫（正会員）

昭和7年生．昭和30年慶応義塾大学工学部電気工学科卒業．昭和32年同大学院修士課程修了．大阪大学工学部助手を経て，通産省工業技術院電気試験所（電子技術総合研究所）に勤務．トランジスタ計算機の研究開発に従事．その間昭和35年から1年半米国イリノイ大学計算機研究所に留学．昭和46年慶応義塾大学工学部電気工学科教授，現在に至る．工学博士．著書「計算機アーキテクチャ」（岩波講座）ほか．電子情報通信学会，IEEE，ACM 各会員．