計算機アーキテクチャ 119－23
（1996．8．28）

大規模共有メモリ型マルチプロセッサ実現のための マルチポートメモリ構成方式の提案

弘 中 哲 夫†

マルチプロセッサに於けるプログラミング・モデルはメモリシステムのアーキテクチャに大きな影響を受ける．ところが汎用のマルチプロセッサを広く普及させる上で重要なことは，コンピュータ・アーキテクチャに変化があっても基礎となるプログラミング・モデルには変化がないマルチプロセッサを実現することである．本稿では上記の問題を解決する方法として *DUMA*(Distributed Uniform Memory Access) 型共有メモリを新たに提案する．

Multiport Memory System for Large Scale Shared Memory Multiprocessors

TETSUO HIRONAKA†

Programming models are badly effected by the memory system architecture of the multiprocessor system. But to widen the use of multiprocessor systems, we need multiprocessor systems which programing model is independent with the architecture of the multiprocessor system. To solve such problems this paper we will present a new implementation method called *DUMA*(Distributed Uniform Memory Access) for implementing shared memory.

1. はじめに

今日広く普及しているパイプライン型スーパコンピュータはベクトル処理とパイプライン方式を採用している．この方式に基づくスーパコンピュータが今日このように広く普及したのは，高速な演算が可能であることだけでなく，非常に単純でかつ汎用性の高いプログラミング・モデルを提供していたということが大きな役割を果たしている．

これに対し次世代のスーパコンピュータとして着目されているマルチプロセッサ型の並列コンピュータは一部実用化されているがまだまだ広く普及していない．これは，現時点において確立したマルチプロセッサ・アーキテクチャが存在しないこと，および，マルチプロセッサを構築する上でPMS (Processor, Memory, Switch) 項目の多様な選択肢により，コンピュータ・アーキテクチャ上に実現されるプログラミング・モデルが多様に変化するためである．このように複数のことなるプログラミング・モデルが存在するということは，特定のマルチプロセッサ・アーキテクチャと特定応用プログラムとの結び付きを強くし，汎用のマルチプロセッサの普及を困難にする．

つまり，汎用のマルチプロセッサを広く普及させる上で重要なことはコンピュータ・アーキテクチャに変化があっても基礎となるプログラミング・モデルには変化がないマルチプロセッサを実現することである．これまでの試みとして国内外で疎結合型のマルチプロセッサ上に，付加装置やソフトウェアを追加することにより分散共有型の共有メモリを実現したり，密結合型のマルチプロセッサ上でソフトウェア的にメッセージパッシング・インタフェースを実現することが広く行われてきた．しかしながら大規模なマルチプロセッサを構築する上ではまだオーバヘッドが大きく，単純で理解しやすいプログラミング・モデルを実現するまで至っていない．本稿では上記の問題を解決する方法として *DUMA*(Distributed Uniform Memory Access) 型共有メモリを新たに提案する．

2. メモリシステムの現状

マルチプロセッサのメモリシステムは実行されるユーザ・プログラムからどのようにメモリが見えるかにより分類すると大きく3つに分けることができる．共有メモリシステム，分散メモリシステム，分散共有メモリシステムである．これらのメモリシステムにはそれぞれ，次のような特徴がある．なお，システム例として挙げられたマルチプロセッサ・システムの構成可能プロセッサ数

† 広島市立大学 情報科学部
Faculty of Information Sciences,
Hiroshima City University
E-mail: hironaka@ce.hiroshima-cu.ac.jp

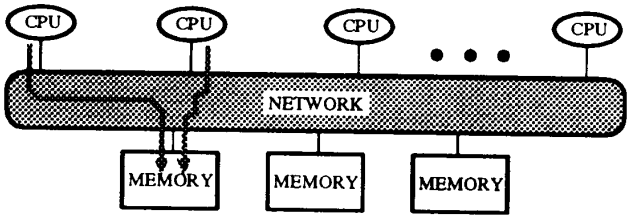

図1 共有メモリ方式

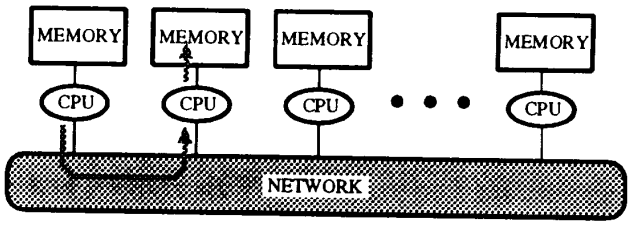

図2 分散メモリ方式

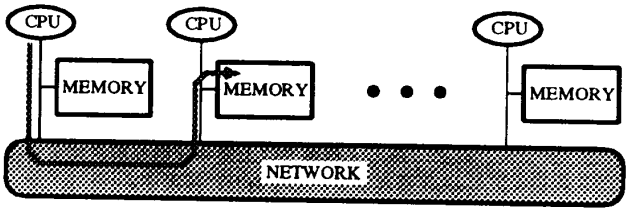

図3 分散共有メモリ方式

は文献 [DMES96] のプロセッサ数を主に用いた.

(1) 共有メモリシステム：図1に示すようにすべてのプロセッサで均等にメモリを共有する方式である. 共有メモリは, すべてプロセッサから単純なロード/ストア命令を用いてアクセス可能である. このような共有メモリシステムではプロセッサ間データ転送を明示的に行う必要がない. このようなマルチプロセッサ・システムとしては 32 プロセッサ構成が可能な NEC の SX-4 [NHI+95] などが存在する.

(2) 分散メモリシステム：図2に示すようにプロセッサ毎に専有できるローカルメモリを持つ方式である. 個々のプロセッサが持つローカルメモリはロード/ストア命令を用いて直接相互のローカルメモリをアクセスできない. 他プロセッサのローカルメモリへのアクセスが必要な場合は, メッセージをローカルメモリを所有するプロセッサに送って操作してもらう必要がある. つまり, 分散メモリシステムで実行されるユーザ・プログラムはプロセッサ間でデータの受渡しを行う場合, 明示的なデータ転送操作を用いる. このようなマシンとして 3680 プロセッサ構成が可能な Intel の Paragon シリーズ [Int91], 1056 プロセッサ構成が可能な Thinking Machines の CM-5 [HT93] や 1024 プロセッサ構成が可能な日立の SR2201/1024 [Hit95], 512 プロセッサ構成が可能な IBM の SP2 [AMM+95] などが存在する.

(3) 分散共有メモリシステム：分散メモリシステムと同様にプロセッサ毎にローカルメモリを持つ方式である. ただし, 純粋な分散メモリシステムと異なり, 図3に示すように論理的には他のプロセッサのローカルメモリを自プロセッサのローカルメモリ同様にロード/ストア命令を用いて直接操作可能になっている. ユーザプログラムに対する共有メモリシステムとしての論理的なイメージはハードウェアのサポート, または, オペレーティング・システムやコンパイラのサポートにより, 暗黙的にメモリアクセスを他のプロセッサへのメッセージへ変換することで実現している. 分散共有メモリシステムでは, ユーザ・プログラムからは明示的なプロセッサ間データ転送操作を用いない. このようなマシンとしては 1024 プロセッサ構成が可能な Cray の T3D [MK95], 166 プロセッサ構成が可能な富士通の VPP500 [KLKVA95] などが存在する.

これら3つのメモリ・システムを実現するマシンについて Dongarra らの調査した Top500 Supercomputers - Worldwide [DMES96] より現状を判断すると, 各メモリシステムでサポートできるプロセッサ数の関係は,

共有 < 分散共有 < 分散

であり, 各メモリシステムを実現する実際のマルチプロセッサ・システムの最大演算性能は,

共有 < 分散共有 < 分散

となっているようである. ここで取り上げたマルチプロセッサ・システムはいずれも Top500 Supercomputers - Worldwide [DMES96] の上位に掲載されていることから発表年代の差はあれ, いずれのも各社最高の技術を用いて開発されたシステムである. したがって, 上記の傾向は各社の設計方針からだけでなく, 高い演算性能実現する上での難易度などからもこのような傾向が現れるのだと考えられる. つまり, LINPACK 等の大型数値演算において現状のアーキテクチャや設計技術では, 世界最高性能を出す計算機では分散メモリ・システムであり, 純粋な共有メモリシステムを用いた 100 プロセッサ構成以上の大規模な共有メモリシステムは現状ではインプリメントするのが困難である. これは, 現在実現されている共有メモリシステムの多くがバス結合型マルチプロセッサであるため, 相互結合網がボトルネックになり大規模なマルチプロセッサ構成をとることができないためである.

これに対して分散共有メモリシステムは分散メモリシステムにはまだ及ばないながらも, 富士通の VPP500 や Cray の T3D などに見られるように分散メモリシステムに迫るプロセッサ数と演算性能が実現され始めている. しかしながら, 現状の分散共有メモリシステムには共有メモリシステムと比べいくつかの問題点がある. 以下の章ではこの問題点を議論する.

3. 現状の分散共有メモリシステムの問題点

現在実現されている分散共有メモリシステムは図3が示すように，分散メモリシステム同様にプロセッサとメモリを対にしたものを基本構成単位とし，これを複数相互結合網により連結することでマルチプロセッサシステムを実現している．分散共有メモリシステムが分散メモリシステムと大きく異なる点は，各プロセッサがそれぞれ相互に各プロセッサ毎に持つメモリを互いに直接アクセス可能にすることで共有メモリを実現していることである．しかしながら，相互結合を結合網を介したメモリアクセス（リモート・メモリアクセス）と，相互結合網を介さないメモリアクセス（ローカル・メモリアクセス）の2種のメモリアクセスにわかれたため，メモリシステムの性能に次のような問題が生じる．

- ローカル・メモリアクセスとリモート・メモリアクセスではメモリアクセス・レイテンシーが大きく異なる．
- 相互結合網のバンド幅によってはローカル・メモリアクセスとリモート・メモリアクセスのデータ転送バンド幅が大きく異なる．
- 各プロセッサ毎に高速なアクセス（ローカル・メモリアクセス）ができるメモリ容量が固定的に決められており，実行するプログラムが各プロセッサ毎に必要とする容量に合わせてこれを増減できない．共有メモリシステムでは，ローカル・メモリアクセスとリモート・メモリアクセスといったメモリアクセスの違いがないのでこのような問題は生じない．

上記のような問題はマルチプロセッサ・システムの演算性能に関係するだけでなく，プログラミング・モデルにも大きく影響を与える．例えば，ローカル・メモリアクセスとリモート・メモリアクセスのアクセス時間が大きく違えば，分散共有メモリシステムであっても結局リモート・メモリにあるデータをアクセスする時，プロセッサの性能を完全に引き出すのであればリモート・メモリのデータをローカル・メモリにコピーして使うことになる．このようにリモート・メモリのデータをローカル・メモリにコピーする操作を多用すると，結局分散メモリシステム上で演算を行っているのと何ら変わらない状況となる．

また，分散共有メモリシステムはマルチプロセッサ・システムを構成する上で次の問題を持つ．

- プロセッサとメモリが対になった物を基本構成単位としているので，プロセッサ数がNならばメモリバンクの数も同じ数Nとなる．このため，複数のプロセッサが同じリモートメモリにアクセスする可能性をメモリバンク数をN以上に増やすことで低減できない．
- プロセッサ間で対称性を保ちつつマルチプロセッサ・システムのメモリ増設を行う場合，プロセッサ数に比例した容量を単位としてメモリ増設を行う必要がある．例えば，1024プロセッサ構成のマルチプロセッサシステムにメモリを増設する場合，1024 × M(kbyte) といった単位でメモリ増設を行わないとプロセッサ間の対称性を保てない．

これらの問題を解決する共有メモリ型マルチプロセッサを実現するための新しいマルチポートメモリ構成方式を以下の章で提案する．

4. 結合網とメモリの一体化

分散共有メモリ方式は3章で述べたようなさまざまな問題点を共有メモリシステムに比べ持つ．本稿ではこれらの問題点を解決するために結合網とメモリが一体化されることで実現される *DUMA*(Distributed Uniform Memory Access) 型共有メモリを新たに提案する．

DUMA 方式は，結合網を構成するスイッチと主記憶を構成するメモリの一部を一つの基本構成単位 *RRAM*(Routing RAM) とし，これを用いてさまざまな結合網（メッシュ，トーラス，ハイパーキューブ，オメガ網などなど）を構成する．なお，*DUMA* では各 *RRAM* 上のメモリはすべてシステム全体で一意の物理アドレスを用いてアドレシングされ，共有メモリとしてすべてのプロセッサが均等にアクセスできる．図4にメッシュ網型に *RRAM* を組み合わせた *DUMA* 型共有メモリシステムの構成例を示す．

DUMA 型共有メモリシステムを構成する各 *RRAM* は図5に示すように Switch 部と RAM 部から構成される．図5を用いて *RRAM* の動作を説明する．アドレスとデータからなる入出力ライン (a) のいずれかよりメモリアクセスが行われた場合，自 *RRAM* が持つメモリに該当アドレスが存在するか否かで次のように動作する．

- 自 *RRAM* が持つメモリに該当するアドレスが存在する場合

  (1) 入出力ライン (a) から Switch 部に入力されたアドレスは Switch 部で自 *RRAM* が担当するアドレスであるか否かチェックし，担当するアドレスであれば (b) の RAM 部にアドレスとデータを入力する．

  (2) (b) の RAM 部でメモリに対するリード / ライトが行なわれ，必要に応じてリード / ライトの結果が Switch 部に返される．Switch 部ではリード / ライトアクセスを行ったプロセッサへのアドレスを付加し，ルーティング・アルゴリズム (結合網により異なる) により (a) から適当な入出力ラインを用いてメモリアクセスを行ったプロセッサに結果を返す．
- 自 *RRAM* が持つメモリに該当するアドレスが存在しない場合

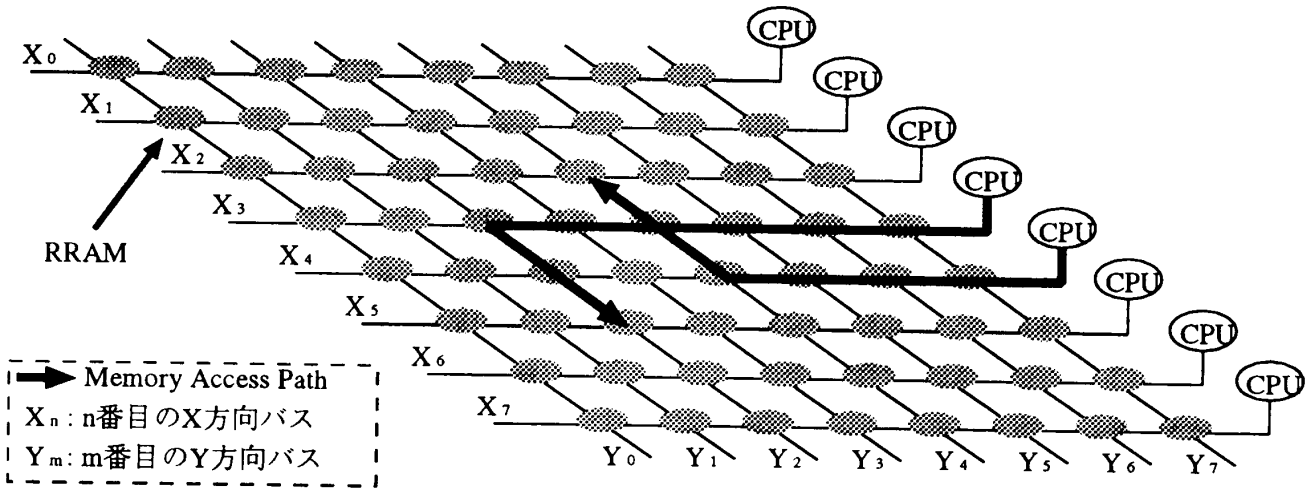

図4 *DUMA* 型共有メモリシステム

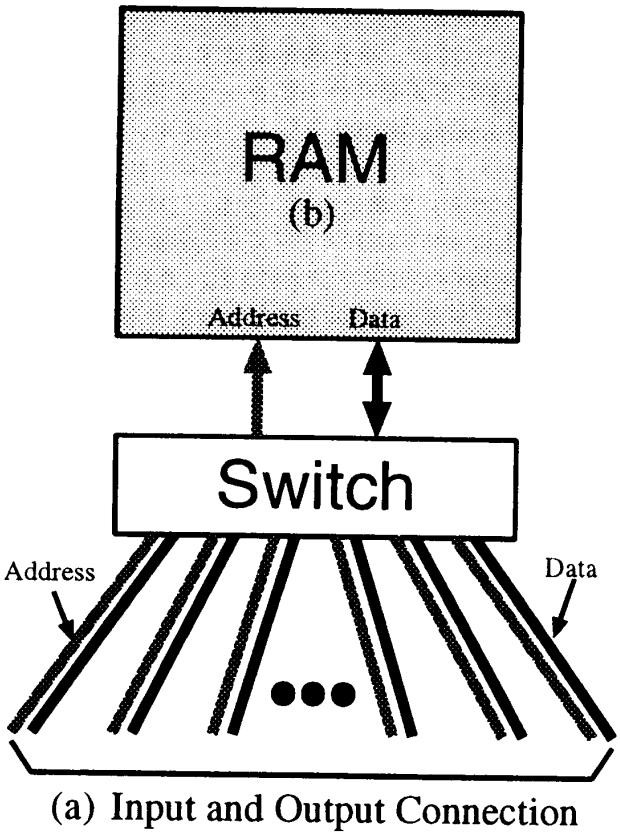

図5 *RRAM* の概念図

(1) 入力ライン (a) から Switch 部に入力されたアドレスは Switch 部で自 *RRAM* が担当するアドレスであるか否かチェックし，担当するアドレスでなければルーティング・アルゴリズム (結合網により異なる) により該当アドレスを担当する *RRAM* またはプロセッサへ接続される入出力ラインへルーティングする．

5. メッシュ網型 *DUMA* 型共有メモリシステム

より具体的に *DUMA* 型共有メモリシステムの性質を調べるため，図 4に示すようなメッシュ網を用いた *DUMA* 型共有メモリシステムを詳細化する．図 4のようなメッシュ網を用いた *DUMA* 型共有メモリシステムを実現する場合，メッシュ網の中でのメモリアクセスをルーティングするルーティング方式により大きく 2 つに分類できる．一つは，メモリアクセスを行うプロセッサからメモリアドレスを担当するノードまでの間にある通信路上のノードをすべて一度に確保してからアクセスを行う方式 (*TYPE1* と定義) である．もう一方は，メモリアクセスを行うプロセッサからメモリアドレスを担当するノードまでの間にある通信路上のノードをすべて一度に確保するのでなく，現在位置に隣接するノードのみを確保する方式 (*TYPE2* と定義) である．図 6，図 7にそれぞれのルーティング方式でノードに使用される *RRAM* の構成図を示す．以下それぞれ，*TYPE1*，*TYPE2* のルーティング方式において *RRAM* の動作を述べる．動作は通信路上のノードとして動作する *RRAM* と，通信先であるメモリアドレスを担当するノードとして動作する *RRAM* に分けて説明する．なお，プロセッサは図 4が示すように X 方向バスにのみ取り付けられており，プロセッサからアクセスを行うノードへのルーティングは X 方向の移動が終了した後に Y 方向の移動を行うものとする．また，メモリアクセスが行われたノードからプロセッサへのルーティングはプロセッサからアクセスを行うノードへのルーティングと同じ経路を逆方向にたどって行われると仮定する．

- *TYPE1* ルーティングのための *RRAM* の動作
  - 通信路上のノードとして動作する *RRAM*

    通信路上のノードとして動作する *RRAM* は

X方向バスからY方向バスへ乗り換えを行うノード以外ではX方向バス，Y方向バスへのインタフェースである図6の(b)(c)はハイインピーダンス状態でありX方向バス，Y方向バスに影響を与えない．X方向バスからY方向バスへ乗り換えを行うノードでは同じY方向バスを共有する*RRAM*間でY方向バスの調停を行い，Y方向バスが確保できたら図6の(b)(c)を短絡することでX方向バスからY方向バスへ乗り換えを実現する．

- 通信先ノードとして動作する*RRAM*
  通信先ノードとして動作する*RRAM*はメモリアクセスがX方向バスから来るのかY方向バスから来るのかで動作が異なる．X方向バスからアクセスされる場合は，図6の(b)(a)が短絡され，メモリアクセスが実現される．Y方向バスからアクセスされる場合は，図6の(a)(c)が短絡されることでメモリアクセスが実現される．

- *TYPE2*ルーティングのための*RRAM*の動作
  - 通信路上のノードとして動作する*RRAM*
    通信路上のノードとして動作する*RRAM*はX方向バス，または，Y方向バス上のLatchが空いていれば，次のデータをLatchし，X方向バス，Y方向バスでパイプライン転送を実現する．X方向バスからY方向バスへ乗り換えを行うノードでは図7の(b)(c)が短絡され，X方向バス上のLatchは使用されず，Y方向バス上のLatchでX方向バスのデータをLatchすることでX方向バスからY方向バスへ乗り換えを実現する．
  - 通信先ノードとして動作する*RRAM*
    通信先ノードとして動作する*RRAM*はメモリアクセスがX方向バスから来るのかY方向バスから来るのかで動作が異なる．X方向バスからアクセスされる場合は，図7の(b)(a)が短絡される．この時X方向バスのLatchは使用されない．Y方向バスからアクセスされる場合は，図7の(a)(c)が短絡される．この時Y方向バスのLatchは使用されない．

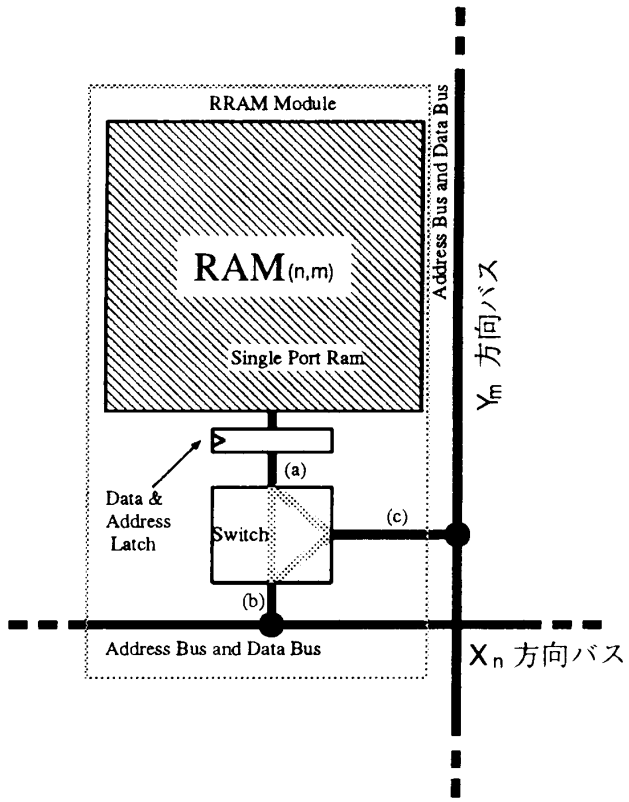

図6 *TYPE1*ルーティングのための*RRAM*

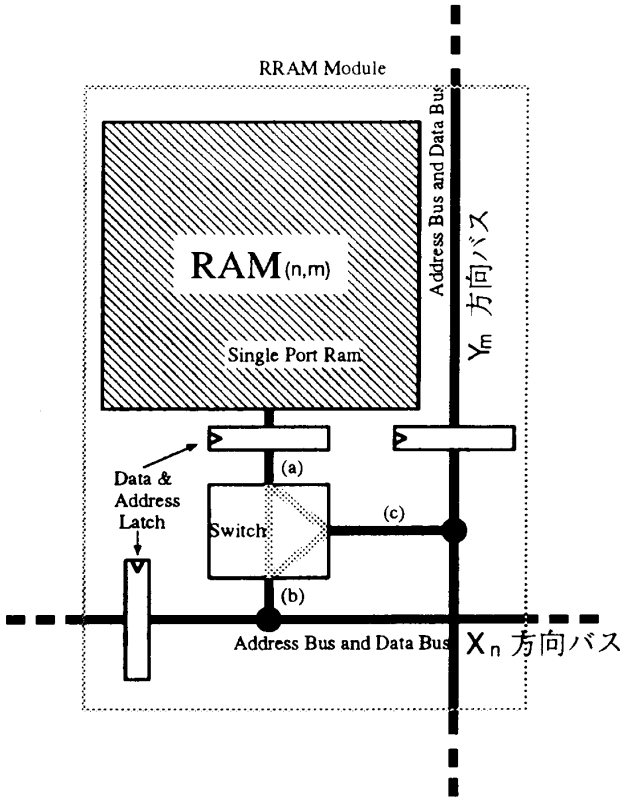

図7 *TYPE2*ルーティングのための*RRAM*

6. アドレスの割り付け

*RRAM*を用いた*DUMA*型共有メモリシステムではどのような方法でアドレス空間を各*RRAM*に割り付けるかにより性能が大きく変化する．ここでは図4のメッシュ網を用いた*DUMA*型共有メモリシステムを例にとり，可能ないくつかのアドレス空間の割り付け方法を検討する．アドレス空間の割り付けの方針として大きく分けてインタリーブ方式とバンク方式がある．なお，

ここで $RRAM_{(n,m)}$ をn番目のX方向バスとm番目のY方向バスの交点にある $RRAM$ とする.

- インタリーブ方式：ワード単位またはキャッシュライン単位で連続するアドレスを異なる $RRAM$ に割り付ける方法である. 具体的には連続するアドレスをワードまたはキャッシュライン単位で $RRAM_{(0,0)}$, $RRAM_{(0,1)}$, $RRAM_{(0,2)}$, ..., $RRAM_{(1,0)}$, $RRAM_{(1,1)}$, ... という具合に複数の $RRAM$ へ連続するアドレス空間をインタリーブする. 十分に高速なネットワークにおいて, このように配置をすることでパイプライン的なアクセスが可能になる. このような構成にすることで分散共有メモリシステムで問題であった, ローカル・メモリアクセスとリモート・メモリアクセスのレイテンシー差を軽減できる. なお, この方式は $TYPE1$ ルーティング方式と組み合わせることで最小のアクセス・レイテンシーを実現できる.
- バンク方式：ワード単位やキャッシュライン単位といった小さな単位で $RRAM$ にアドレス空間を割り振るのでなく, 比較的大きな塊のアドレス空間をまとめて個々の $RRAM$ に配置する. 非常に大きいレイテンシーを持つネットワークを用いる場合において一つのコネクションで転送するデータ量を増加させることができ, スループットを上げることが可能になる. なお, この方式は $TYPE2$ ルーティング方式と組み合わせることで最大のスループットを実現できる.

7. おわりに

本論文では, 新しい共有メモリアーキテクチャである $DUMA$(Distributed Uniform Memory Access) 型共有メモリを方式を提案した. 本方式では結合網とメモリを一体化することで分散共有メモリ型マルチプロセッサにおいて問題となるローカル・メモリアクセスとリモート・メモリアクセスのレイテンシーの差を軽減し, バス型共有メモリシステムで問題となるメモリバンド幅を不足を解決することでより大規模なマルチプロセッサシステム構築を可能にする. 本方式ではプロセッサ台数にスケーラブルなメモリバンド幅を提供しつつ, 簡単な共有メモリモデルに基づくプログラミング・モデルを提供することができる.

今後の課題は計画としては, 要素 $RRAM$ の詳細設計および, 要素部品化の検討, および, $DUMA$ 型メモリアクセス性能の評価を行う予定である.

謝辞

日頃ご討論戴く広島市立大学 情報科学部 情報工学科の児島彰助手, 高山毅助手, 藤野清次教授に感謝致します.

参考文献

[AMM+95] T. Agerwala, J. L. Martin, J. H. Mirza, D. C. Sadler, D. M. Dias, and M. Snir. SP2 system architecture. *IBM Systems Journal*, Vol. 34, No. 2, pp. 152–184, 1995.

[DMES96] Jack J. Dongarra, Hans W. Meuer, and eds. Erich Strohmaier. Top500 supercomputer sites, 1996. http://parallel.rz.uni-mannheim.de/top500.html.

[Hit95] Ltd. General Purpose Computer Division Hitachi. 超並列コンピュータ「hitachi sr2201」, 1995. http://www.hitachi.co.jp/Prod/comp/hpc/jpn/sr1.html.

[HT93] W. Daniel Hillis and Lewis W. Tucker. The CM-5 connection machine: A scalable supercomputer. *Communications of the ACM*, Vol. 36, No. 11, pp. 31–40, November 1993.

[Int91] Paragon XP/S product overview. Intel Corporation, 1991.

[KLKVA95] Sachio Kamiya, Pierre Lagier, Werner Krotz-Vogel, and Noboru Asai. Two programming paradigms on the vpp system. In Masaaki Shimasaki and Hiroyuki Sato, editors, *Proc. International Symposium on Parallel and Distributed Supercomputing*, pp. 35–44, Fukuoka, Japan, September 1995. Kyushu University, Computer Center of Kyushu University.

[MK95] Hiroto Matsuhashi and Takehiko Kato. Hybrid adaptive parallel programming model on cray mpp system and its applications. In Masaaki Shimasaki and Hiroyuki Sato, editors, *Proc. International Symposium on Parallel and Distributed Supercomputing*, pp. 51–60, Fukuoka, Japan, September 1995. Kyushu University, Computer Center of Kyushu University.

[NHI+95] N. Nishi, S. Habata, M. Inoue, H. Matumoto, and Kondo. Sx-4 architecture for scalable parallel vector processing. In Masaaki Shimasaki and Hiroyuki Sato, editors, *Proc. International Symposium on Parallel and Distributed Supercomputing*, pp. 45–50, Fukuoka, Japan, September 1995. Kyushu University, Computer Center of Kyushu University.