Vol. 36 No. 6　　情　報　処　理　　June 1995

解　説

重点領域研究：超並列原理に基づく情報処理基本体系

5. 超並列処理計算機 JUMP-1 のアーキテクチャ†

富　田　眞　治††

1. はじめに

1976年に発表されたCRAY-1に始まるスーパコンピュータは科学技術計算分野の研究開発を大きく変革してきた．CRAY-1に代表される今日のスーパコンピュータは1次元ベクトル処理に基礎を置いたパイプライン方式を採用している．この方式では高速性，連続性，汎用性，単純性および単一性を特徴として，科学技術分野で普及してきた．しかし，1 TFLOPS(Tera FLOPS：毎秒10^{12}回の浮動小数点演算)の性能が要求される今日，ランニングコストとしての電力問題，集中記憶システムから派生するメモリ供給系の性能限界が顕在化してきた．

一方，マイクロプロセッサの高速化はとどまるところを知らず高速化の一途を辿っている．このような状況を反映して，次世代のスーパコンピュータとしてマルチプロセッサ型の並列コンピュータが注目され，一部実用化されている．

マルチプロセッサ型のスーパコンピュータはPMS(Processor, Memory, Switch)項目で以下のように分類できる．

（1） プロセッサ

要素プロセッサとしてスカラプロセッサ(スーパスカラ，VLIWを含む)を採用し，これらをきわめて多数(数千～数万)結合するスカラ並列プロセッサ(SPP)とベクトルプロセッサを多数(数百～数千)結合するベクトル並列プロセッサ(VPP)に分類できる．

（2） メ　モ　リ

各要素プロセッサから直接アクセスできる共有メモリ空間が存在する共有メモリ方式と各要素プロセッサからはそれ固有のメモリにしかアクセスできない私有メモリ方式(メッセージ交換方式)に分類される．いずれの方式でもプロセッサが多数存在するので，メモリは物理的には分散配置される．前者はさらにキャッシュメモリのコヒーレンス制御をハードウェアで行う方式とキャッシュメモリの制御をソフトウェアに任せる方式がある．

（3） ネットワーク

各要素プロセッサを相互に結合するネットワークの幾何学的形状によって分類される．ネットワークは大別して，静的網と動的網に分類され，前者にはリング系，トーラス系，トリー系，ハイパキューブ系，シャフル系などがあり，後者にはクロスバ系，多段結合網系などがある．

並列コンピュータはこれらの分類項目を反映して多様なシステムが開発され，現時点において確立された単一アーキテクチャは存在しない．このシステムの多様性が，CRAY-1の単一アーキテクチャを採るパイプライン型スーパコンピュータと比較して広く普及しない一因となっている．これは特定の並列アーキテクチャと特定応用との結び付きが強く，汎用並列コンピュータを構築しにくいことが一因であるが，近い将来，数種の並列アーキテクチャに収斂する必要がある．したがって，より汎用性の強い高性能な並列コンピュータの研究が必要であり，JUMP-1研究もその一つである．

2. JUMP-1 のシステム構成[1)~3)]

図-1 にシステム構成を示す．図のようにクラスタ構成を採用しており，実験機としては128クラスタの構成をとっている(4000クラスタまで拡張可能である)．各クラスタは，**図-2** のように4台のSuper SPARC Plusマイクロプロセッサ(動作周波数は60 MHz．各1 MBの2次キャッシュメモリ付き)，2台の主記憶装置(各64

† Architecture of the Massively Parallel Computer, JUMP-1 by shinji TOMITA(Kyoto Univ.).

†† 京都大学大学院工学研究科

MB)，2台の知的メモリ機構(以後，MBP (Memory Based Processor)と呼ぶ)，数本の高速シリアルI/Oリンク(以後，STAFF-Linkと呼ぶ)から構成される．これらのクラスタが階層トーラス網(以後，RDT網と呼ぶ)を通して結合されている．

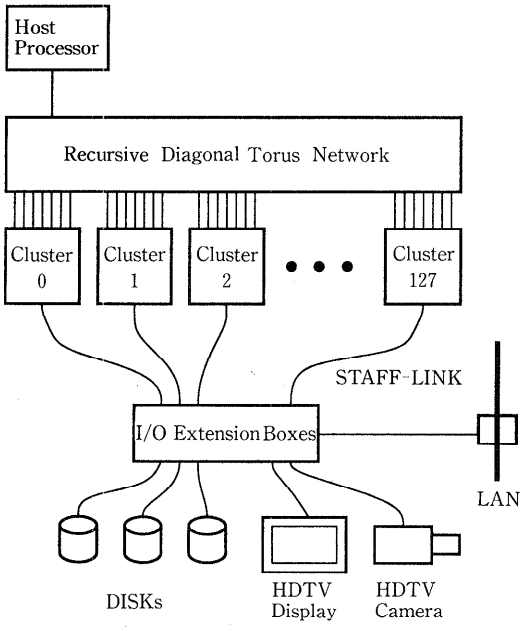


図-1 JUMP-1 全体構成

JUMP-1は以下の特徴を有している．

(1) キャッシュコヒーレンス制御機構を内蔵した共有メモリ方式

図-2のクラスタ内はスヌープ方式，クラスタ間は後述の疑似フルマップ方式を採用し，データ属性ごとにコヒーレンスプロトコルの変更を可能とし，無駄な通信の削減を図っている．

(2) 通信オーバヘッドの削減を図れるMBP機構の導入による細粒度並列処理方式

メモリ共有型超並列コンピュータではメモリは分散配置される．したがって近接地に存在するメモリと遠隔地に存在するメモリへのアクセス時間(レイテンシ)は異なる．したがって，メモリアクセスや通信処理を演算処理とオーバラップさせることによって高速化を図ることができる．

メモリアクセスと通信処理を分担するのがMBPであり，以下のような知的な機構を有している．

①レイテンシ削減

メモリレイテンシに対処する方式としてはレイテンシ隠ぺい(hiding)と削減(shortening)がある．前者はマルチスレッドによるレイテンシ時間の有効利用が考えられるが，これは要素プロセッサ自体の設計と深く関係しており，JUMP-1のように商用プロセッサを使用する場合は実現困難

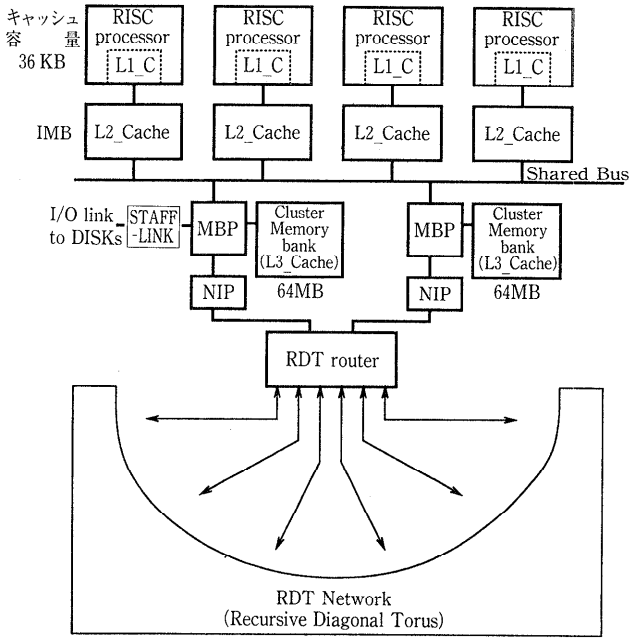


図-2 クラスタ内構成図

である．JUMP-1では，キャッシュメモリに対するハードウェア管理機構を強化し，データプリフェッチ，データ属性に基づいたプロトコル制御などでレイテンシ削減を図っている．

②無駄な通信の削減

無駄な通信を削減することも重要であり，各データへの同期ビットの付与，FIFO機構の装備などの工夫もなされている．

JUMP-1では，このような手法を取り入れることによって，細かな処理単位での並列処理が可能となり，細粒度並列処理に向いた方式を実現している．

（3） スケーラブルで直径の小さい階層トーラス網

相互結合網に要求される要件として，次数，直径，総スイッチ数，ルーティング容易性，拡張性，3次元実装性，耐故障性などがある．また，応用サイドから見ると，流体力学などでの隣接通信，多体問題などでの遠隔通信(放送)および全対全通信が必要となる．隣接通信向き応用には2次元トーラスは不可欠であり，JUMP-1でもトーラスを相互結合網のベースに考えた．このトーラスをベースに遠隔通信や全対全通信の高速化が図れる階層トーラスの一種であるRDT(Recursive Diagonal Torus)方式を採用した．

（4） 高速な入出力機構

同軸ケーブルを用いた100 Mbps以上での通信が可能なSTAFF-Link (Serial Transparent Asynchronous First-in First-out Link)を開発した．AMD社のTAXIチップを中心に構成している．図-1に示したように，各クラスタからのSTAFF-Linkを集め，ファイルシステムやHDTV対応の画像入出力などが可能である．

3. メモリアーキテクチャ

JUMP-1では，演算処理を実行するプロセッサの他にメモリアクセス・通信を分担する知的なメモリ処理機構(MBP)を導入し，細粒度並列処理を実現している．本章ではMBPの主要な機能である，仮想共有メモリ，キャッシュコヒーレンス機構，高速同期機構などを中心にメモリアーキテクチャの特徴について述べる．

3.1 仮想共有メモリ

図-2に示したように各クラスタには128 MB

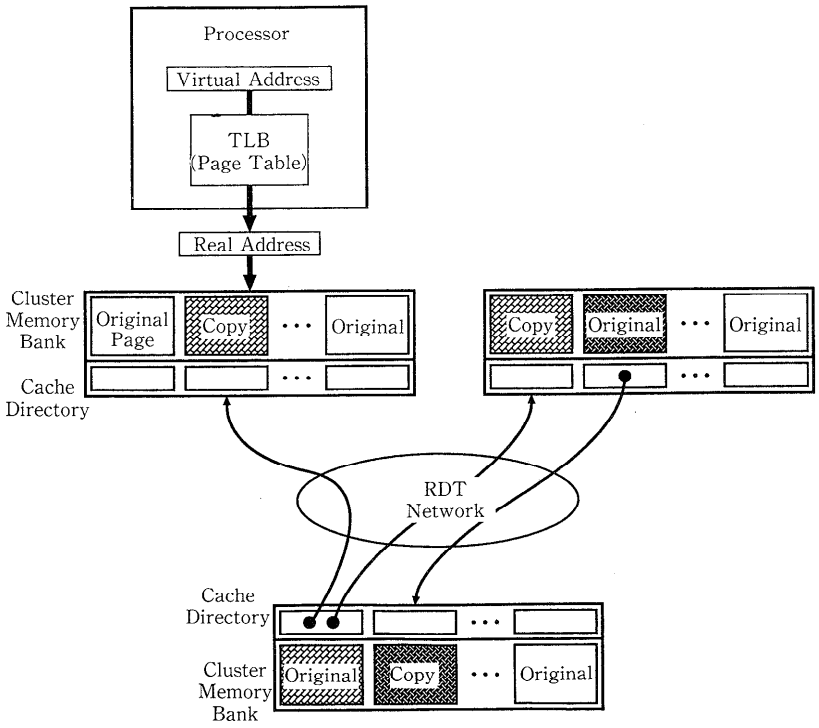


図-3 仮想共有メモリ

のクラスタメモリが装備されている．クラスタメモリは独自(original)空間とコピー(copy)空間に分割されている．分割の仕方は動的に変更可能なように構成されている．あるクラスタメモリのコピー空間には他のクラスタの独自空間内のページをキャッシングできるようになっている．これを**図-3** に示す．独白ページにはそれがキャッシングされるクラスタを示すディレクトリ部が付随している．これは次に述べる疑似フルマップ方式で制御される．

3.2 クラスタ間キャッシュコヒーレンス

（1） 疑似フルマップ方式

バス以外の一般構造のネットワークを有するシステムでは，キャッシュコヒーレンス方式として，ディレクトリ方式が採用される．この方式にはフルマップ，リミテッド，チェインの三つの方式が提案されている．フルマップ方式が最も高速ではあるが，ディレクトリのメモリ容量がプロセッサ(クラスタ)台数とメモリ容量の積に比例するため，超並列処理では実際的ではない．リミテッド，チェイン方式では性能に問題点がある．したがって，JUMP-1 ではフルマップの特徴を残しつつ経済的に実現可能な疑似フルマップ方式を考案した．**図-4** に方式の概略図を示す．図では各クラスタ(2 進木の葉ノードに対応)が 2 進木構造(中間ノードを T で表す)でコヒーレンス制御されている(実際の JUMP-1 では，全クラスタは後述の RDT ネットワークを用いた 8 進木によってコヒーレンス制御されている)．各ディレクトリは 2 進木の階層に対応したエントリ(図では T 0〜T 4)を有しており，そのエントリが 1 である場合にはその 2 進木ノードの枝につながるクラスタ内に少なくとも一つのコピーが存在することを示す．

当該ディレクトリへの変更操作(たとえば，無効化や更新操作など)は 1 の立っている 2 進木ノードの枝につながるすべてのクラスタに対して放送機構を用いてなされる．図の例で，A のコピーが B，C，D に存在する場合には，ノード T 3 の右枝につながるクラスタ(Q と B)，ノード T 1 の右枝につながるクラスタ(C，D など 8 クラスタ)に変更操作が放送される．変更操作の完了通知はこの逆の方向で流されるが，ネットワークが階層構造になっているので，途中で結合操作(Combining)がなされ，トラヒックを減少できる．しかし，非常に遠距離にある領域の一つのクラスタにだけキャッシングされている場合(A のコピーが E にのみ存在する場合)は非常に大きなオーバヘッドとなる．このため，このような場合には直接キャッシングされるクラスタ番号を示すようになっている(図で MD=1 のとき)．すなわち，一部リミテッド方式も取り込んだ方式となっている．

（2） 疑似フルマップ方式の評価

一般にディレクトリ方式を用いたキャッシュコヒーレンス制御では，以下の点を考慮しなくてはならない．

●共有ブロックの大きさ：すなわち，通常のキャッシュライン(64 B 程度)とするのかページ(4 kB 程度)とするのかの選択がある．共有データの大きさを小さくすると，ディレクトリのエントリ数が増大するが，共有される確率が減り(すなわち，偽共有(False Sharing)が減り)，制御が高速化される．

●書込み時のポリシ：データに対して書込みが生じた場合に，他のキャッシュに対して無効化(Invalidation)するか更新(Update)するかの選択がある．これは応用にも大いに依存する．無効化が有効な場合は，書き込まれたプロセッサでのみしばらくの間そのデータが使用される場合である．このような場合には，共有度も 1 となり，コヒーレンス制御にともなうネットワーク負荷も小さくなる(更新方式では，書込みを行う当該プロセッサの最終書込みのみが重要なのに，すべての書込み時に更新情報が必要とされ，オーバヘッドが多い)．一方，更新方式が有利な場合は，頻繁

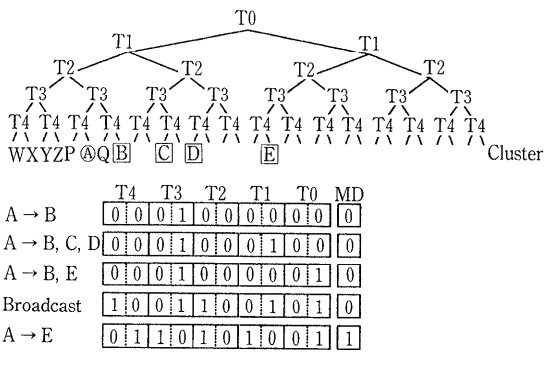


図-4 疑似フルマップ

に書込みプロセッサが変更される場合や書込みデータがすべてのプロセッサで利用される場合である．**図-5** は一次方程式の反復解法 $X \leftarrow AX$（X は n 次元ベクトル，A は $n \times n$ 行列）の計算例であり，計算された X の値がただちにすべてのプロセッサで使用されるので，更新方式が有利である場合を示す．

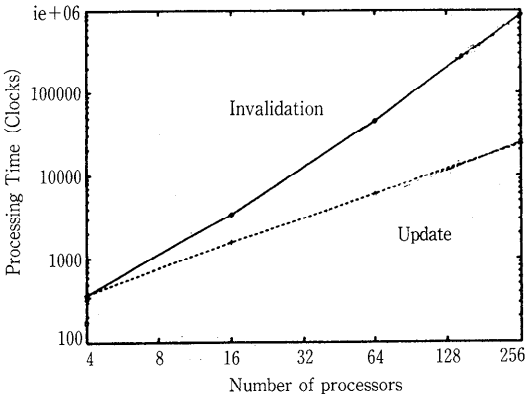
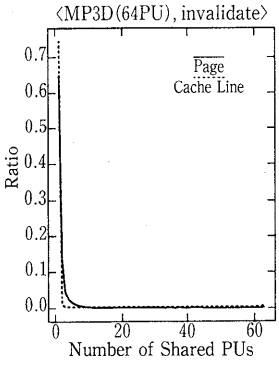


図-5 各種プロトコルによる性能差（クロック数）

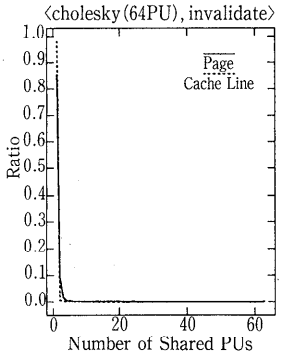
リミテッド，チェイン方式と比較した際，疑似フルマップ方式の評価をする際に以下の点が重要となる．

●ディレクトリ容量

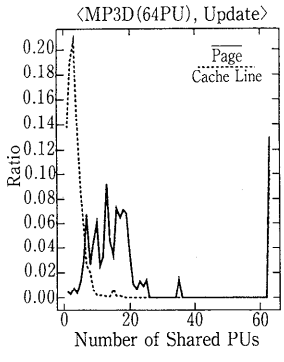
共有度（共有ブロックを共有するクラスタ数）が問題となる．共有度は共有ブロックの大きさおよび無効化・更新方式に依存する．**図-6** に 64 台のプロセッサで，スタンフォード大学のベンチマーク SPLASH の MP 3 D，Cholesky を実行させた場合の無効化・更新，共有度，共有サイズとそれらの頻度の関係を示す．これらが相互に大きく関連していることが分かる．いまプロセッサ台数を 16384 台の超並列処理の場合を考える．無効化方式でキャッシュラインサイズで管理をすると，



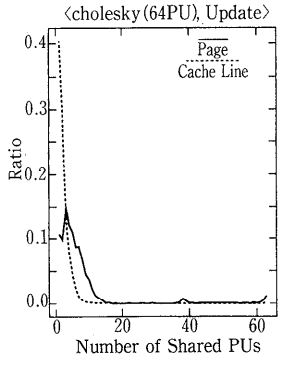
（a） MP 3 D：64 PU（Invalidate 型キャッシュプロトコル）での共有度



（b） Cholesky：64 PU（Invalidate 型キャッシュプロトコル）での共有度



（c） MP 3 D：64 PU（Update 型キャッシュプロトコル）での共有度



（d） Cholesky：64 PU（Update 型キャッシュプロトコル）での共有度

図-6 共有ブロックサイズ，無効化/更新方式と共有度との関係

リミテッド方式では共有度は2程度で十分であり，ポインタビット数は28ビットとなる．これに対して疑似フルマップでは36ビット必要であるので，メリットは少ない．しかし，リミテッド方式で，ページ単位の場合には，共有度は4程度(56ビット)となり，疑似フルマップが優位に立つ．また，更新方式の場合には共有度は非常に大きくなり，疑似フルマップが優位になる．

●ネットワークを流れるトラヒック量

疑似フルマップでは遠距離に小数の共有クラスタが存在すると，無駄なデータ転送が多くなり，これがネットワーク負荷を増大させ，性能低下が生じる．この評価データの収集はシミュレーションではまだ行っていない．実機上での大規模な実測が必要となろう．

3・4 高速同期・通信機構

メモリ共有型マルチプロセッサの汎用化と高速化を図るためには同期や通信のオーバヘッドを極力減らし，細粒度の並列処理が可能でなくてはならない．また，キャッシュメモリを装備していることが前提となっており，キャッシュメモリを活かした方式とすることが望まれる．本節ではJUMP-1で採用した2，3の特徴ある方式について言及する．

(1) キャッシュメモリ

JUMP-1のキャッシュメモリは図-2に示すように，SPARCプロセッサ内の1次(L 1)キャッシュ，チップ外に置かれた2次(L 2)キャッシュ，クラスタメモリによる3次(L 3)キャッシュから構成される．1次キャッシュは実アドレスキャッシュ(実アドレス32ビット)で命令用20 kB，データ用16 kBの容量がある．またブロックサイズは命令用64 B，データ用は32 Bであり，ライトスルーで制御される．プロセッサ外へはライトスルーであるので，ライト情報が取り出せ，また任意ブロックの無効化が外部より可能である．これらの機能を使用して2次キャッシュを構成する．

2次キャッシュは1 MBの統合キャッシュであり，ライトバックで動作する．ブロックサイズは32 Bである．ブロックに付加されているタグはINV(Invalid)，EX-Dty(Exclusive Dirty)，LS-Cln(Locally Clean)，LS-Dty(Locally Dirty)，GS-Cln(Globally Clean)の5状態である．Localはクラスタ内に共有が閉じていることを示す．また，2次キャッシュにはIストラクチャのfull/emptyビットが8バイトごとに付随している．

2次キャッシュの特徴は以下のとおりである．

●ページ属性として無効化と更新があり，データ属性に応じたプロトコル選択が可能である．

●実アドレスの上位5ビットをコマンドとして使用して，先の無効化・更新プロトコルと組み合わせて使用することにより，以下のような様々な機能を実現できる．

① プリフェッチと注入

前者はデータを必要とするプロセッサがあらかじめ，データをキャッシュにロードできる機構を指す．後者は外部より強制的にキャッシュへのデータロード(injection)を指す．注入の例として，FIFOキューの先頭が消費された場合に強制的にキューの末尾に次のデータをクラスタメモリからロードできる．また，注入を利用すると後述のRead-Allを実現できる．

② 更新データのマージ

更新プロトコルの際にライト時ごとに更新するのは通信量の増大となるので，いくつかのライトをまとめる．

③ 早期共有解除

更新プロトコルの際に，一定時間利用されないブロックは共有状態を解除して(early replace)，更新のための無駄な転送を削除する．

④ 自浄機構

EX-Dtyの状態にあるブロックを使用されなくなると，いち早くクラスタメモリに書き戻し，Clean状態とする(self clean-up)．このようにすると，最新情報がメモリに存在し，他のプロセッサからのリードミスヒット時の高速化につながる．[7]

(2) キャッシュコヒーレンスプロトコルの動的切り替え

図-5に示したようにコヒーレンスプロトコルは対象とするデータ属性によって異なるように制御すれば無駄な制御情報の転送をなくすことができ，高速化を図ることができる．データ属性としては，INV(無効化)，UD(アップデート)のほか，RA(リードオール，あるプロセッサがデータを読むときに，すべてのプロセッサでデータの取込みを行う)，などが考えられる．これらのデ

ータ属性と最適プロトコル選択はコンパイル時に行う必要があり，コンパイラの能力が十分高い必要がある．また，JUMP-1では実行時にも動的に変更可能なように設計されている．

（2） Iストラクチャと FIFO 機構

プロセッサ間の通信をプロデューサとコンシューマモデルで考えると，1対1通信，1対多通信，多対1，多対多の場合が考えられる．これらの高速化について以下の方式を提案している．

1対1通信の場合には，同期ビットとデータが必要である．これら二つの情報が異なるメモリ領域に格納されている場合には，ネットワークによっては，二つの情報の到着順序に逆転現象が生じる場合がある．このような場合には，データが正しくコンシューマに到着したことを確認した上でプロデューサは同期ビットを設定する．コンシューマは同期ビットの設定を知ってはじめてデータを読める状態となる．この過程は大きなオーバヘッドとなる．

JUMP-1ではIストラクチャを使用して，このオーバヘッドの回避を図っている．すなわち，同期ビットとデータが不可分なものとして(データにタグビットを付けた1語として)，転送される．プロデューサは同期ビットを0に設定して，メモリにデータを書き込む．コンシューマは同期ビットが0となるまで待つ．プロデューサは同期ビットが1となるまで次の転送を待つ．コンシューマはデータを受け取ると，同期ビットを1とする．異なるプロセッサ間(N個)で1対1通信がN個なされる場合には，各プロデューサが次のデータの転送をするためには，各コンシューマからの同期ビット1を受け取らねばならない．このためネットワークにはN個のメッセージが流れる．

これに対し，N個のコンシューマへのデータ到着を次に述べるバリア同期機構で高速に検知し，それを各プロデューサ，コンシューマに放送する手法がある．このとき，次にはプロデューサは同期ビットを1とすることでコンシューマへデータ到着を知らせる．したがって，データ転送ごとに同期ビットの役割が逆転する(最初のデータ転送では0，次の転送では1，その次では0，でデータの利用可能状態を示す)．

1対多通信では，一つのプロデューサからのデータは多数のコンシューマで利用される．データ受領に対するカウンタ機構とデータを保持するプロデューサからコンシューマへの放送機構で高速化は達成できるが，JUMP-1では先のビット逆転方式で対処している．

多対1の通信の際に FIFO 機構[6]が使用されることが多い．すなわち，多数のプロデューサから転送されたデータをコンシューマ側で FIFO キューに一時的に蓄えて使用する．この FIFO キューの制御には，先頭要素の格納位置，次データを格納する末尾要素の格納位置などのポインタ類があり，これらを排他制御の下で管理する必要がある．これをプロデューサとコンシューマの各プロセッサで行うと非常に大きなオーバヘッドとなる．したがって，FIFO という構造体をカプセル化して，その操作をすべて MBP で行うと，無駄な転送をなくすことができる．

多対多は以上の方式を組み合わせて実現している．

（3） バリア機構[9]

通常のバリア機構では，そのプロセッサがバリアに到着したことを他のプロセッサへ通知する機能と，すべてのプロセッサがバリアに到着するまで待つ機能が一つのバリア命令で実現されていた．これを二つの操作に分割し，オーバヘッドを減少させる手法にエラスティックバリアがある．基本的には，上記の操作を別々の二つの操作で行わせる方式である．二つの操作間に実行できる命令群があれば，オーバヘッドを減らすことができる．JUMP-1では MBP の機能としてこの機構をサポートしている．

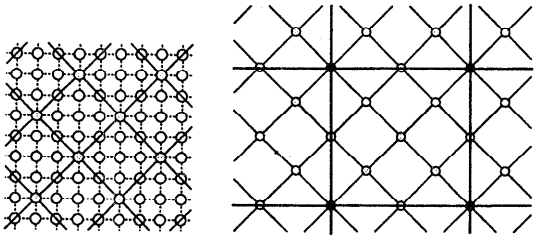
4. ネットワークアーキテクチャ

4.1 ネットワーク構成方式[4]

図-7(a)に示すように RDT は次のように階層トーラス方式で構成される．

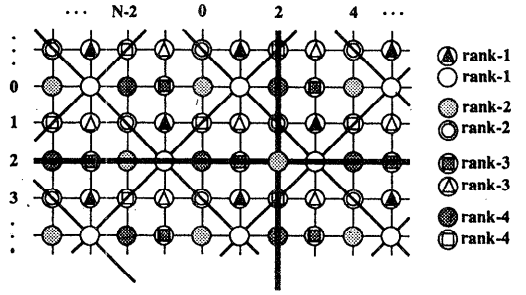
● Rank-0 トーラスは通常の2次元トーラスである．

● Rank-$(i+1)$ トーラスは，Rank-(i) トーラスのノードを$(X, Y)i$とすると，$(X+2, Y)i$，$(X-2, Y)i$，$(X, Y+2)i$，$(X, Y-2)i$に位置するノードを結合するように構成する．すなわちトーラスの粗さを8倍(面積にして)にして，45度回転させると，Rank-$(i+1)$ トーラスが構成さ

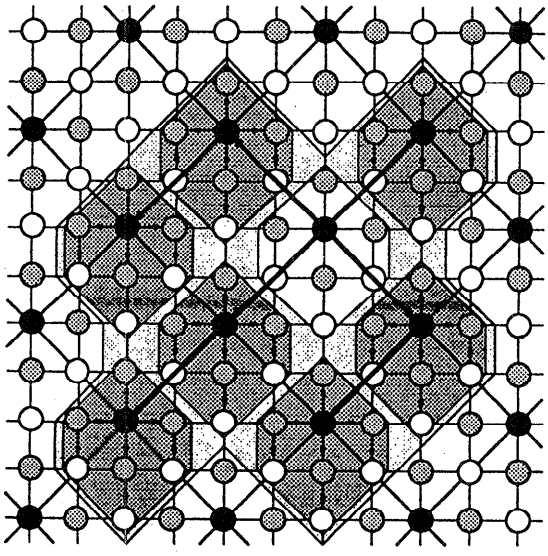


（1） rank-o 上の rank-1 トーラス　（2） rank-1 上の rank-2 トーラス

（a） 上位トーラスの形成例



（b） RDT の構成



（c） ブロードキャストのテリトリ（キャッシュコヒーレンス制御などに利用．8 進木構成となっている）

図-7 RDT の構成

れる．

● Rank-0 トーラスの各ノードにすべての Rank-(i) トーラスが結合されれば，理想的であるが，コスト的あるいは実装的に実現が困難である．したがって図-7(b)に示すように，Rank-0

表-1 RDT の直径と次数

ネットワーク ＼ プロセッサ台数	4096 のとき	65536 のとき
2D トーラス	32(4)	128(4)
3D トーラス	24(6)	48(6)
ハイパキューブ	12(12)	16(16)
DeBruijn	12(4)	16(4)
RDT	8(8)	12(8)
	直径(次数)	

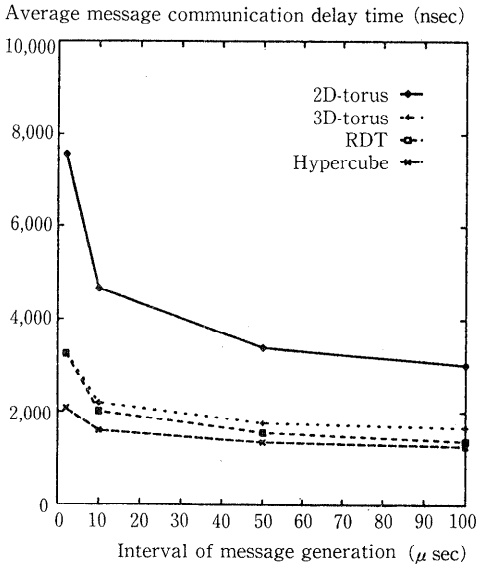


図-8 Virtual Cut-through を使用した場合の平均通信遅延時間

トーラスの各ノードには Rank-(i) トーラスのどれか一つのトーラスが結合されるように構成する．このようにすると，Rank-0 トーラスのノードからのリンク数，すなわち次数は 8 となる．

RDT はその階層構造を利用したマルチキャスト(ブロードキャスト)機能(図-7(c)参照)，ベクトルルーティングと呼ぶ容易なルーティング方式，トーラス/トリー/ハイパキューブ，シャフル網などの高速エミュレーション能力を有している．また，表-1 に他の代表的な結合網との直径と次数の比較を示す．

4.2 ネットワークの評価[8)]

相互結合網の性能評価は動的なトラフィックに対してなされる必要があるので，INSIGHT と呼ぶシミュレータを開発してシミュレーションによる簡単な評価を行った．比較対象は 2 次元トーラス，3 次元トーラス，ハイパキューブであり，

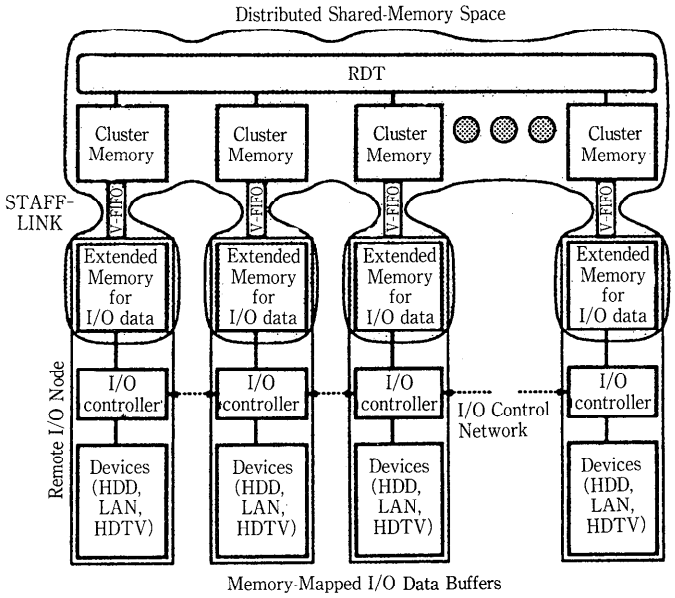


図-9　JUMP-1の共有入出力バッファ

各々4096ノードの場合について行った．INSIGHTではStore/Forwards，Wormhole，Virtual Cut-throughなどのフロー制御，メッセージ長，ネットワーク周波数，チャネル幅などをパラメータに設定して種々のネットワークのシミュレーションが可能である．

図-8にフロー制御で最も性能の良かったVirtual Cut-through方式についてメッセージ発生間隔，平均通信遅延時間，ネットワーク形状の関係を示す．RDTは性能的には3次元トーラスとハイパキューブの中間的な特性を有することが分かる．

5．入出力アーキテクチャ[5)]

超並列処理における入出力方式は重要なテーマであるにも関わらず，これまでほとんど研究がなされてこなかった．JUMP-1の入出力機構はRDTの高速データ転送能力とMBPの柔軟な制御を生かした方式となっており，以下の特長を有している．

（1）　高速STAFF-Link

従来の高速データ転送方式では，フラットケーブルを用いたパラレル通信が一般的であった．しかし，この方式では入出力装置との距離が短い，コネクタなどの配置や実装が煩雑となる，などの欠点がある．光データ転送やATM網の普及などの要因もあり，高速シリアル転送可能な各種デバイスが利用可能となってきたので，JUMP-1ではモトローラ社のTAXIチップを使用したSTAFF-Linkを提案し，試作することにした．最高転送能力は220 Mbpsであり，同軸ケーブルを使用する．STAFF-Linkは送受信ノード側に一つずつ通信ブロックが置かれている．送信側ノードでは送信FIFOが一杯になるまでデータを転送できる．受信側ノードは受信バッファが空になるまでデータを受け取る．また，受信バッファのキューがバッファ長の半分以上となると送信側の通信コントローラに対して送信中断を要求する．いずれにせよ送信側から見ると仮想的なFIFOキューによって通信がなされているように見える．

（2）　共有入出力バッファ

入出力の際には入出力バッファを通してデータ転送がなされる．JUMP-1ではこの入出力バッファをクラスタの共有メモリ空間の一部にマッピングする方式を採用している．したがって図-9に示すように，クラスタメモリが拡張された構成になっている．共有メモリへのアクセスを行うことによって，クラスタ間で入出力装置を共有することができる．

たとえば，HDTVを用いて画像出力する場合には，フレームメモリが分割(行方向ブロック分

割，ドット領域分割などが可能)され，それらが各クラスタメモリにマッピングされる．また，フレームメモリは2組用意されており，一方が画像出力の際には他方が次画像の生成用に使用され，リアルタイム処理が可能となっている．

6. おわりに

JUMP-1は現在4つのチップ(MBP，キャッシュ制御，ルータ，バス制御)を開発中である．多数の大学の共同開発であり，種々の困難な側面もあるが，協力体制を維持して開発が進められている．1995年度末にはハードウェアシステムの稼働を予定しており，JUMP-1上での種々の応用プログラムが実働する予定である．

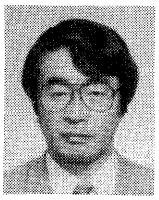
JUMP-1にはこれまでに述べたように，多数のアイデアが盛り込まれており，一部オーバスペックと見られる箇所も多数存在する．また，コンパイラ研究者，OS研究者との密接な連携も必要な箇所が多数存在する．規模の小さなベンチマークのシミュレーションでは実システムの挙動は捉えることはできないので，大規模な実応用プログラムを多数実行させてアーキテクチャ，コンパイラ，OSをシステムとして総合的に評価する必要がある．これこそが実マシンを大学で開発する意義ともいえる．このような評価の後はじめてマルチプロセッサ型並列計算機のアーキテクチャの単一化が図られ，真の普及が達成されよう．

謝辞 JUMP-1は以下の方々の共同研究として開発されている．ここに日頃の努力に対して感謝の意を表する．東京大学田中英彦教授，平木敬助教授，松本尚助手，佐藤充氏，慶応大学天野英晴助教授，東京工科大学工藤知宏助教授，京都大学中島浩助教授，森眞一郎助手，五島正裕氏，神戸大学金田悠紀夫教授，中條拓伯助手，岡山理科大学小畑正貴助教授，九州工業大学末吉敏則助教授，久我守弘講師，柴村英智助手，九州大学村上和彰助教授．

参考文献

1) 田中英彦編：文部省重点領域研究「超並列処理原理に基づく情報処理基本体系」シンポジウム予稿集1～5(1992～1994)．
2) Hiraki, K. et al.: Overview of the JUMP-1, an MPP Prototype for General Purpose Parallel Computations, Proc. of ISPAN, pp. 427-434 (1994)．
3) 平木 敬ほか：超並列プロトタイプ計算機JUMP-1の構想，情報処理学会計算機アーキテクチャ研究会資料，102-10(1993)．
4) 工藤，好村，福嶋，西村，楊，天野：超並列計算機JUMP-1のクラスタ間結合網RDTにおける階層マルチキャストによるメモリコヒーレンス維持手法，JSPP 95, pp. 257-264(1995)．
5) 中條拓伯ほか：分散共有メモリ型超並列計算機JUMP-1のディスク入出力サブシステム，JSPP 95, pp. 67-74(1995)．
6) 五島正裕ほか：Virtual Qneue：超並列計算機向きメッセージ通信機構，JSPP 95, pp. 225-232(1995)．
7) 森眞一郎ほか：Self Clean-up Cacheの提案，JSPP 95, pp. 265-272(1995)．
8) 柴崎，久我，末吉：超並列計算機のための相互結合網シミュレータ，情報処理学会論文誌，Vol. 35, No. 4, pp. 589-599(1994)．
9) 松本 尚：Elastic Barrier一般化されたバリア同期機構，情報処理学会論文誌，Vol. 32, No. 7, pp. 886-896(1991)．

(平成7年3月6日受付)



富田 眞治 (正会員)

1945年生．1968年京都大学工学部電子工学科卒業．1973年同大学院博士課程修了．工学博士．同年京都大学工学部情報工学教室助手．1978年同助教授．1986年九州大学大学院総合理工学研究科教授，1991年京都大学工学部情報工学科教授，現在に至る．計算機アーキテクチャ，並列処理システムなどに興味を持つ．著者「並列計算機構成論」「計算機システム工学」「並列処理マシン」「コンピュータアーキテクチャI」など．電子情報通信学会，IEEE，ACM各会員．