1 はじめに

キャッシュコヒーレントな分散共有メモリをもつマルチプロセッサシステム CC-NUMA (Cache Coherent Non-Uniform Memory Access model) は、将来の大規模かつ高性能な並列計算機の構成方式の一つとして注目されている。CC-NUMA は、バス結合型の小規模並列計算機に比べて多数のプロセッサを接続でき、またこれらの小規模並列計算機で開発されたプログラムを容易に移植できる利点も持つ。現在 CC-NUMA は SGI の Origin[1]、Sequent の NUMA-Q[2] など、商用化が進んでいる。

文部省重点領域研究の一環として 1994 年から開発されている JUMP-1 は、数千プロセッサを越す超並列計算機上に効率の良い分散共有メモリ、同期、メッセージ転送を実現するためのテストベッドである。JUMP-1 は、一般的に用いられている共有メモリ型マルチプロセッサを数千プロセッサの規模にまで拡張することができるスケーラブルなアーキテクチャを目指しており、そのため本論文で述べる RDT(Recursive Diagonal Torus) [3][4][5] ネットワーク等、様々な特徴的な要素を用いて構成されている。

2000 年 3 月、16 クラスタ 64 プロセッサのハードウェアが完成したため、以後実機を用いて様々な評価を行なっていくこととなった。本論文では、JUMP-1 では、クラスタ間ネットワークである RDT ネットワークのマルチキャスト機構に関して、いくつかの性能向上メカニズムについて述べ、その評価結果を示す。

2 JUMP-1 の構成

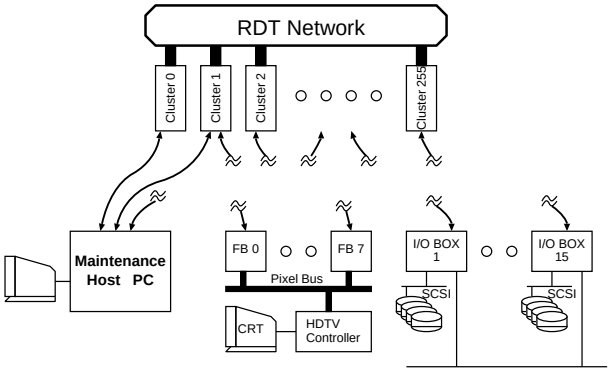

図 1: JUMP-1 の構成

JUMP-1 は最大で数千プロセッサ規模まで拡張可能なアーキテクチャを目標としており、そのクラスタ間は図 1 に示すように結合網 RDT で接続されている . RDT は二次元トーラスの Fat-tree 状の階層構造を持つ結合網で、RDT ルータチップ [6][7] により実現される。さらに、各クラスタは並列 I/O サブシステムを構成する高速シリアルリンク STAFF-Link[8] により、ディスクおよび画像データ用 Frame Buffer(FB)[9] と接続される。また、各クラスタはそれぞれメンテナンス用ホスト PC に接続され、システムのブートアップやデータの相互転送が行われる。

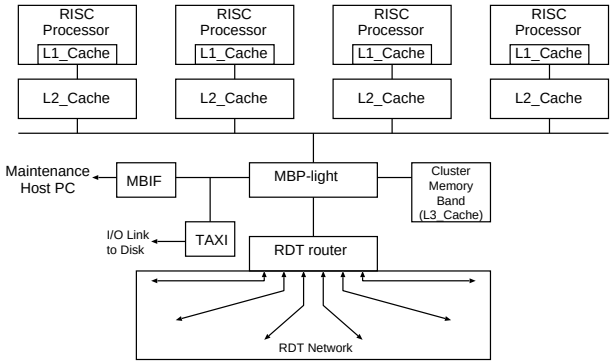

図 2: JUMP-1 クラスタの構成

図 2は、JUMP-1 クラスタの内部構成である。プロセッサには SuperSPARC+を用いており、4 プロセッサで 1 クラスタを構成する。各プロセッサは京都大学により開発された高性能の L2 キャッシュ [10] と Cluster Bus(CBus) を介して共有メモリ管理プロセッサである MBP(Memory Based Processor)-light へと接続されている。MBP-light には、分散共有メモリおよび L3 キャッシュとして用いられる 32MByte のクラスタメモリ、STAFF-Link を構成する TAXI チップ、RDT ルータチップやメンテナンスシステム用 FPGA である MBIF (Maintenance Bus InterFace) 等も接続されている。

なお、本研究の対象である RDT ネットワークの様々な機能は、RDT ルータチップと MBP-light の協調動作によって実現される。

2.1 RDT ネットワーク

RDT は基本のトーラス構造の上に目の粗いトーラスを 45 度ずつ傾けながら再帰的に積み上げていくことにより、トーラス構造と階層構造の両方を満足させることを狙っている。図 3に示すように、基本となるトーラス上に上下に ± 2 ずつ離れた点同士を結んだ Rank 1 のトーラスを作る。さらに、この Rank 1 のトーラス上で上下に ± 2 ずつ離れた点同士を結んで Rank 2 のトーラスを作る。結果として、これは 8 ずつ離れたトーラスとなる。この操作を全ての点について上位トーラスが形成できないようになるまで繰り返す。このようにして形成されたものを完全 RDT と呼ぶ。しかし、完全 RDT は Rank 数の 4 倍のリンク数をノード当りに必要とすることから、現実的な結合網ではない。そこで、各ノードは基本となるトーラスの他に持つことのできる上位トーラスの数を 1 と定める。JUMP-1 では各 Rank に対し平等で、どの Rank へも 1 step の移動で利用可能な割り付けを採用している。JUMP-1 設計時に考えられていたノー

ド数 (16384 ノード) に対応した RDT を RDT(2,4,1)/α と呼び、この構成を図 4に示す。

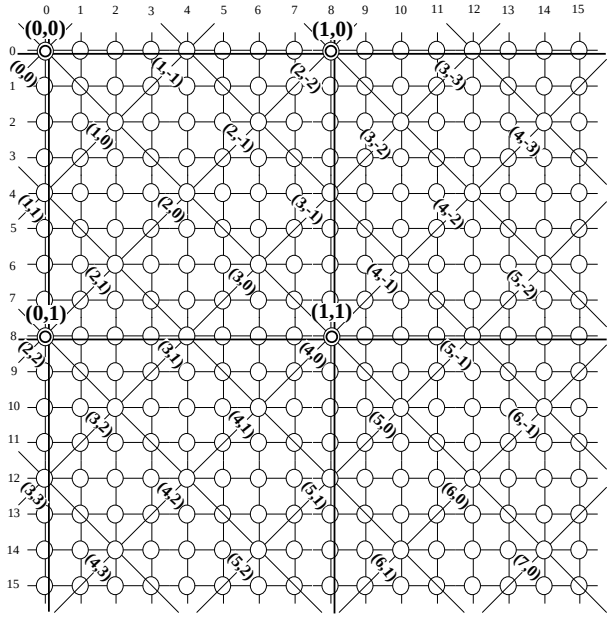

図 3: 完全 RDT の構成

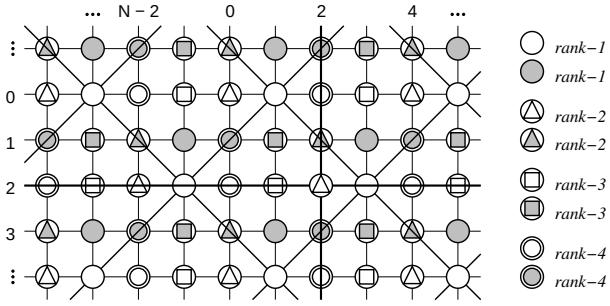

図 4: RDT(2,4,1)/α の構成

RDT の大きな特徴は、理論的な結合網である完全 RDT と実装上の結合網を最初から分離して考えている点にある。このことは、結合網がわかりにくくなるという欠点を持っているが、実装上の構成が柔軟になる点では利点となる。また、16384 ノードより小さなシステム、例えば 1024 プロセッサ (256 ノード) の場合、さきほどの RDT(2,4,1)/α の割り当てに対し、例えば Rank 4 を持つノードを Rank 2 か Rank 1 を持つノードへ、また Rank 3 を持つノードを Rank 1 を持つノードへと再割り当てすることで対応可能となる。もしシステムを拡張する場合でも同様な再割り当てを行うことにより、link の無駄は存在しない。そして、数百から数万と広いノード数の範囲でかなり効率の良い構成を取ることができる。

2.2 RDT ルータチップ

JUMP-1 で用いられている RDT ルータチップ [6, 7] の構成を図 5に示す。

RDT では Rank 0 のリンクに 4 本、上位 Rank のリンクに 4 本、JUMP-1 の設計当時に MBP(-light) はクラスタ内に二つあったためにそれぞれに 1 本づつ必要である。よって、全体は 10 入力 10 出力のクロスバが基本になる。さらに、後に述べる応答パケット収集機能が加わるため、クロスバは 11 入力 10 出力となる。また、転送のビット幅は 1 チップ当たり各リンクについて 18bit である。JUMP-1 では 2 個のルータチップを bit slice に用いることにより、全体で 36bit 幅を実現する。ピンを有効に利用するため、JUMP-1 の通信はすべて双方向線により行う。

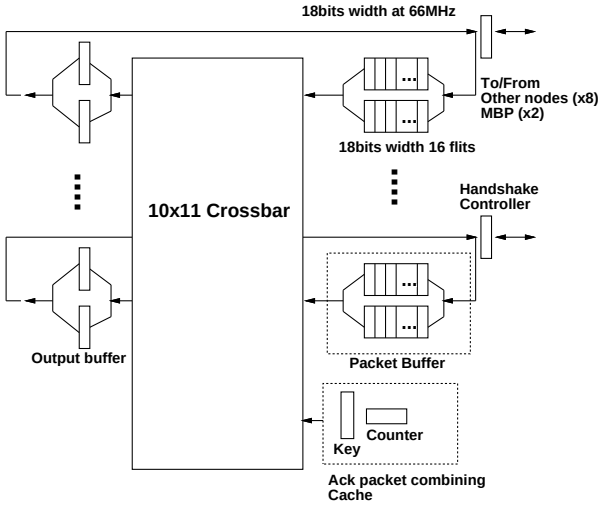

図 5: RDT ルータチップの基本構成

チップには日立の 0.5μm BiCMOS ゲートアレイである HG22S125 を利用している。このゲートアレイは高速である上、ECL 入出力により転送線を直接駆動することができる。

最大周波数	60 MHz
消費電力	19.4 W
gate 使用率	63 %
pin 数	299
package	PGA

表 1: RDT router の実装諸元

JUMP-1 でのネットワークパケットは最大 16 フリットであり、3 フリットは 2 つのチップに対して共通の内容のヘッダとなる。また、通常用いられるパケットは最大 16 フリットの可変長のマルチキャスト用パケット (multicast packet) となる。他にも、3 フリットのみの応答パケット (acknowledge packet) や MBP-light とルータ間の制御に

用いられる command packet、status packet、shootdown packet、setup packet 等が存在する。マルチキャストパケットのヘッダにはそれぞれの階層でのマルチキャスト用のビットマップが格納されている。このビットマップは、次に使用するビットマップがルータ内で常に自動的に先頭になるようにシフトされる。

転送方式は asynchronous wormhole 方式 [11] であり、各入力にはパケット 1 個分を格納することのできるバッファを 2 組持ち、それぞれが 1 つの channel を構成する。この channel を使い分けることにより、deadlock-free な階層マルチキャストを実現することができる。RDT ルータチップでは 1 対 1 転送は階層マルチキャストの一部として実現され、固定かつ deadlock-free、FIFO 性の保証された転送となる。

RDT router の実装上の特徴を以下に示す。

- クロスバと双方向線の同時 1 クロックのアービトレーションによる高速な経路スイッチ
- 可能な転送から順に行う効率の良いマルチキャスト
- SM 法や LARP 法、LPRA 法の三種類の階層マルチキャスト [12] のサポート
- 応答パケットのルーティングと収集用 cache
- FIFO 性を保証した shootdown と setup の装備

2.3 MBP-light

2.3.1 MBP-light の全体構成

図 6に MBP-light の全体構成を示す。MBP-light は、RDT ルータチップの制御を行う **RDT Interface**、クラスタメモリと共有バスの制御を行う **MMC**(Main Memory Controller)、プロトコル制御を行うコアプロセッサ **MBP Core** の三つの部分から構成される。

MMC は CBus とのインタフェース、およびクラスタメモリを構成する SDRAM やタグメモリとしての SRAM などの制御を行っている。RDT インタフェースは RDT ルータとのパケット送受信や応答パケット制御等を行っている。MBP Core は 16 bit データ長、21 bit 命令長のプロセッサで、コアプロセッサ上のソフトウェアにより MMC や RDT Interface 間でのデータ転送やプロトコル処理などを行なう。

2.3.2 RDT Interface の構成

RDT Interface の構成を図 7に示す。RDT Interface は、パケットの送受信を行う Packet handler、応答パケットの収集を行う Ack Collector、応答パケットの自動生成を行う Ack Generator、Send/Receive Unit および MBP Core ・RDT インタフェース間のパケット受渡しに用いられる PBR(Packet Buffer Register) から構成される。RDT ルータとのパケットの送受信は Send/Receive Unit によって行なわれ、送受信するパケットの制御は他の 3 モジュールが全て独立に動作して行なわれる。

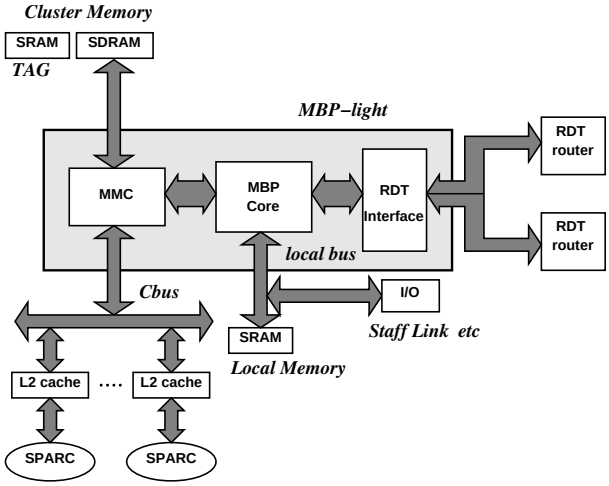

図 6: MBP-light の全体構成

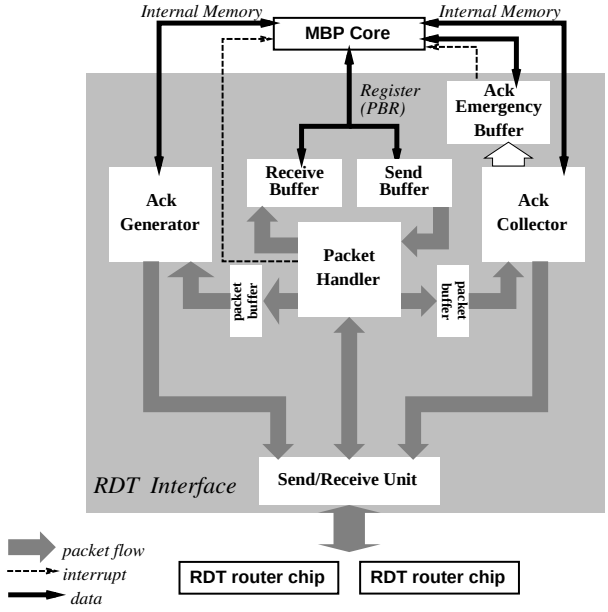

図 7: RDT Interface の構成

3 性能向上のためのメカニズム

RDT では、性能向上のために下に示すような様々なメカニズムが用意されている。

- 縮約階層ビットマップ方式である RHBD 方式を利用した階層マルチキャストのサポート

- MBP-light の Ack Generator によるマルチキャストパケットに対する応答パケット (acknowledge packet) の自動生成
- RDT ルータチップによる Rank 0 の応答パケットの自動収集
- MBP-light の Ack Collector による上位 Rank の応答パケットの自動収集

ここでは、今回評価を行なった、MBP-light による Ack パケットの自動生成、RDT ルータチップによる応答パケットの自動収集、MBP-light による応答パケットの自動収集について簡単に述べる。

3.1 MBP-light の Ack Generator による応答パケットの自動生成

マルチキャストに対応した応答パケットの収集は少しでも短時間で終了した方がよい。そのために、応答パケット収集が必要なマルチキャストパケットを受け取った node は速やかに応答パケットを生成し、送信する必要がある。Ack Generator では Net Cache と Ackmap Cache の 2 つの Cache を用いて、ハードウェアにより応答パケットの自動生成機能を実現する。

- Net Cache

  Ack Generator では応答パケットの自動生成のために、512 エントリの Net Cache を 2 組備えている。クラスタ (ノード) 内のキャッシュラインの状態に従って、Net Cache の状態を作成しておく。そして、マルチキャストパケット到着時に Net Cache を調べ、マルチキャストパケット中のアドレスがヒットした場合、登録しておいた状態を応答パケットに付加することができる。ただし、この Net Cache に対するエントリの作成や追い出しなどの管理は MBP Core 上のソフトウェアで行なう必要がある。

- Ackmap Cache

  Ack Generator では Ackmap Cache を用意して、応答パケットに返送する情報として Ackmap (マルチキャストのどのリンクに対応した応答パケットかを示す情報) を持たせる。もし Achkmap Cache にヒットし、かつエントリが有効である場合には、そのエントリを読み出して応答パケット用の情報を設定する。Ackmap Cache は 256 ノード分 (8 bit) のエントリを確保しており、256 ノード以内の場合には必ず Ackmap Cache にヒットする。ただし、256 ノード以上のシステム構成では、MBP Core のソフトウェアで Ackmap Cache の登録や追い出しなどの管理を行なう必要がある。

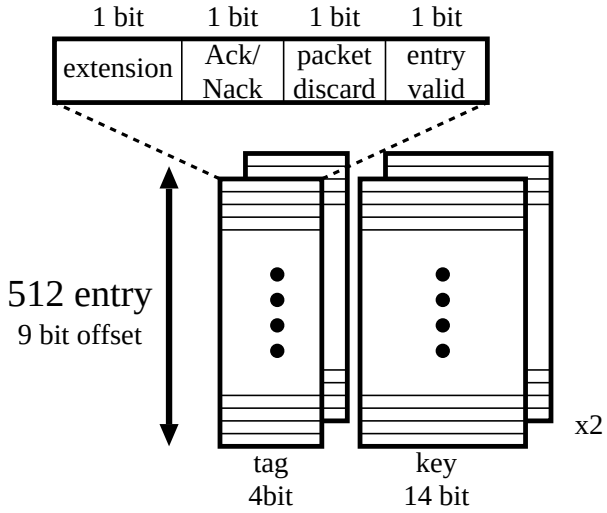

図 8: Net Cache の構成

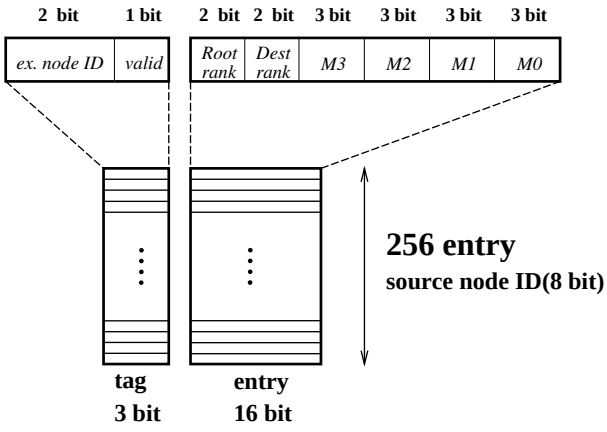

図 9: Ackmap Cache の構成

3.2 RDT ルータチップによる応答パケットの自動収集

キャッシュコヒーレンス維持のために階層マルチキャストを行なった場合を考える。一貫性制御プロトコルによっては、全てのノードがそのパケットを受けとった事を発信元のノードが知る必要がある。JUMP-1 においては、この操作を応答パケットを発信元に送り返すことにより行なっている。しかし、パケットが送られたノードが一対一通信で応答パケットを発信元に返すとネットワークの負荷が増大してしまう。そこで、マルチキャストした階層をそのまま利用して、応答パケットを収集する方法が提案されている。

この収集は各階層のマルチキャスト発信元の MBP-light が行なうが、マルチキャストが頻繁だと応答パケットの収集により、ルータ-MBP-light 間のラインがボトルネックになる可能性がある。このため、ルータチップでは応答収集のためのパケット 1 個分の Cache を持つ。この Cache は基本トーラス (Rank 0) を用いて行なう末端ノードに対する応答パケットに対してのみ有効である。

末端ノードに対するマルチキャストが行なわれた時、Cache が空いていれば、そのパケットの識別子が Cache のキーに、マルチキャスト用ビットマップ中の 1 の数が Cache のカウンタにセットされる。木構造の末端ノードから到着した応答パケットはパケットの識別子をキーとして Cache を検索し、ヒットすればカウンタを 1 減らす。この検索及びカウンタの減算は同時にパケットが到着した場合でも、遅れなしで並列に実行することができる。カウンタの値が 0 になると、応答パケット収集 Cache は上位階層のノードに対して応答パケットを送ると共に、エントリを空にする。

識別子がヒットしなかった場合や、エントリが空でないのに新たなマルチキャストが行なわれた場合、応答パケットおよびマルチキャストパケットは MBP-light に送られ、応答パケットの収集は MBP-light で行なわれる。RDT ルータは、ハードウェアの制約により、木構造の末端ノードにのみ、それも 1 パケット分しか応答パケット収集 Cache を持たない。しかし、通常応答パケットはマルチキャスト到着時に末端ノードから即時に返されるので、この方法は少量のハードウェアで最も大きな効果が期待されるものである。

3.3 MBP-light の Ack Collector による応答パケットの自動収集

マルチキャストパケットを発生したノードはその終了の確認のために、応答パケットを収集しなければならない。RDT ルータチップは基本トーラス (Rank 0) 上の応答パケットを 1 エントリ分だけ収集する機能を持つ。しかし、ルータチップ内の収集用エントリが溢れた場合の基本トーラスでの収集と上位トーラスでの収集は、MBP-light 中の RDT Interface の役目となる。

そのため MBP-light の RDT Interface 内には、応答パケットの収集を行なうための Ack Collector と呼ばれる機構が実装されている。Ack Collector は基本トーラス用と上位トーラス用に Ack Cache と呼ばれる Cache を保持している。図 10に示すように、Ack Cache は 128 エントリと 2 エントリからなる表を持ち、全体として特殊な 2 way で set associative な Cache として機能する。そして、この Cache を用いることで、それぞれの階層における応答パケットの収集を完全にハードウェア制御で行なう。

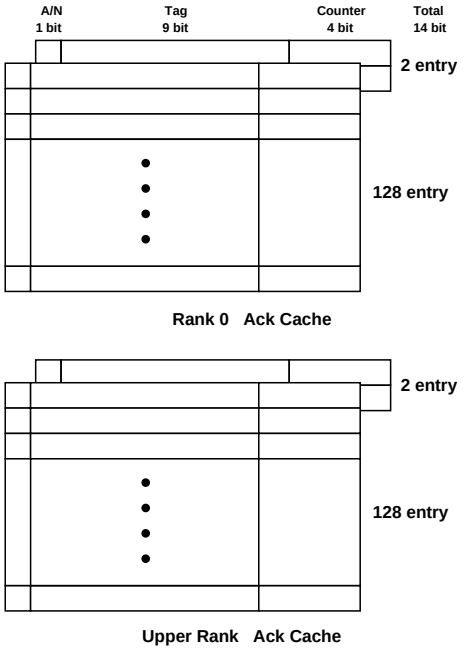

図 10: Ack Cache の構成

4 評価

今回は、RDT の様々な高速化メカニズムのうち、応答パケットの自動返送機構、RDT ルータチップによる応答パケットの自動収集機構、MBP-light の RDT Interface による応答パケット自動収集機構について評価を行なった。

評価に用いたのは、4 クラスタ (ノード)16 プロセッサ構成のシステムで、RDT ネットワークの結線は図 11のような、単純なトーラス構造となっている。

4.1 応答パケット自動生成機構の評価

今回の応答パケット自動生成機構の評価は、自動生成機構を使用した場合と、自動生成機構と同様の操作を MBP-light の Core プロセッサによるソフトウェアで行なった場合について、それぞれサイクル数を測定するという方法で行なった。

評価の際、まずマルチキャストパケットを図 11の Sender から W に向けて送信し、この送信処理開始時にクロック

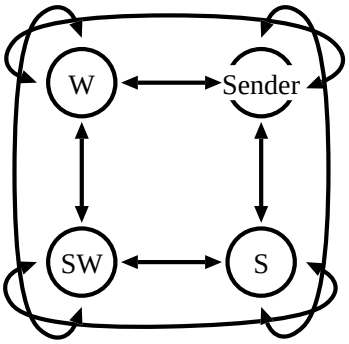

図 11: 4 クラスタにおける RDT 結線図

カウンタの動作を開始させる。W のノードでは応答パケットを Ack Generator で生成するか MBP-light の Core プロセッサのソフトウェア処理によって生成し、Sender に送り返す。そして、Sender ノードの MBP Core に応答パケットが届いた時、カウンタの値を読み出す。

自動生成	51 サイクル
ソフトウェア処理	91 サイクル

表 2: 応答パケット返送に要するサイクル数

レジスタ初期化等	9 サイクル
Ackmap Cache 読みだし	5 サイクル
応答パケットの作成	22 サイクル
その他	1 サイクル

表 3: ソフトウェア処理による処理内容の内訳

この結果を、表 2に示す。このように、応答パケットを自動生成した場合は、ソフトウェア処理した場合に比べて 40 サイクル短縮されており、ハードウェア処理が有効であることが分かる。

4.2 RDT ルータチップによる応答パケット自動収集機構の評価

ルータチップの自動収集機構の評価では、宛先が 1 箇所、2 箇所、3 箇所のそれぞれについて、自動収集機構を使用した場合と同様の処理を MBP-light の Core プロセッサのソフトウェアで行なった場合について、それぞれサイクル数を測定した。マルチキャストパケットの宛先で応答パケットを返送する際には、Ack Generator の自動返送機構によるハードウェア処理を利用している。サイクル数のカウント開始は、Sender ノードによるマルチキャストパケットの送信処理開始時としている。自動収集機構を利用した場合は収集が終わったことを示す応答パケットが MBP-light の Core に届いた時をカウント終了とし、ソフトウェア処理の場合はそれぞれの宛先からの応答パケットが全て届いた事を確認する処理が終了したところでカウント終了としている。

宛先が W のみ	自動収集	51 サイクル
	ソフトウェア	64 サイクル
宛先が W, S	自動収集	51 サイクル
	ソフトウェア	84 サイクル
宛先が W, S, SW	自動収集	63 サイクル
	ソフトウェア	104 サイクル

表 4: 応答パケット自動収集に要するサイクル数

パケット登録時	エントリが空かどうか	6 サイクル
	エントリ登録	5 サイクル
パケット収集時	エントリ読み出し、一致検出	6 サイクル
	カウンタ減算、値の格納	7 サイクル
	その他	2 サイクル

表 5: ソフトウェア処理による処理内容の内訳

この結果を表 4に示す。まず、自動収集機構を利用した場合について見てみると、宛先が 1 箇所 (W のみ) の場合と 2 箇所 (W, S) の場合ではサイクル数に差はなく、複数の宛先からの応答パケットの収集が同時に行なえていることが分かる。また、51 サイクルという値は、第 4.1節で示した応答パケットの自動返送の際のサイクル数と同一であり、オーバヘッドがないことが分かる。また、宛先が 3 箇所 (W, S, SW) の場合は 63 サイクルであるが、第 4.1節と同様のテストで、宛先を SW に変更した際には 63 サイクルとなる。これは、パケット転送のホップ数の増加のためであるが、このことから、宛先が 3 箇所となった場合でもオーバヘッドなしに収集処理を行なう事ができることが分かる。

一方、ソフトウェア処理した場合、宛先が 1 箇所の時はそれほど差がみられないが、宛先が増えるごとに 20 サイクル増加していることが分かる。JUMP-1 では、各階層でのマルチキャストの最大数は 8 であるので、宛先が多い場合にはソフトウェア処理によるオーバヘッドは大きなものとなる。

4.3 MBP-light による応答パケット自動収集機構の評価

MBP-lignt による応答パケット自動収集機構の評価は、RDT ルータチップによるものと同様の条件で行なった。ただし、MBP-light の Ack Cache は 128 エントリと 2 エントリの特殊な 2 way set associative であるが、Ack Collector

の動作をソフトウェア処理するプログラムでは簡単のため、128 エントリの表のみを作成して処理している。

宛先が W のみ	自動収集	67 サイクル
	ソフトウェア	70 サイクル
宛先が W, S	自動収集	75 サイクル
	ソフトウェア	107 サイクル

表 6: Ack 自動収集に要するサイクル数

パケット登録時	エントリが空かどうか	6 サイクル
	エントリ登録	7 サイクル
	その他	2 サイクル
パケット収集時	エントリ読み出し、一致検出	12 サイクル
	カウンタ減算、値の格納	11 サイクル
	その他	2 サイクル

表 7: ソフトウェア処理による処理内容の内訳

この場合のサイクル数は表 6のようになった。MBP-light の Ack Collector で応答パケットの自動収集を行なった場合は、RDT ルータチップで収集した場合と異なりオーバヘッドが生じる。ただし、ソフトウェア処理した場合は宛先が増えると収集に要するサイクル数はかなり増加するが、ハードウェア処理では小規模にとどまっている。また、ソフトウェア処理では 128 エントリの表のみを用いたが、実際の Ack Cache と同様の処理を行なった場合は更にオーバヘッドが拡大することが予想される。

5 おわりに

本論文では、JUMP-1 のクラスタ間結合網である RDT ネットワークのマルチキャスト機構に関して、いくつかの性能向上メカニズムを紹介した。また、その効果について、MBP-light の Core プロセッサによるソフトウェア処理と比較した評価を行ない、オーバヘッドが短縮される事を示した。

しかし現在では、まだこれらの機構を容易に利用するためのライブラリは実装されていない。よって、実アプリケーションに基づくこれらの機構の有効性の検証も行なわれていない。

従って、今後は RDT の性能向上メカニズムを利用するためのライブラリ整備を行ない、DSM プログラムに組み込んでいく必要がある。また、ライブラリや DSM プログラムの整備が完了した際には、実アプリケーションを用いた性能向上メカニズムの有効性の評価も行なっていきたいと考えている。

6 謝辞

JUMP-1 開発グループのメンバーの皆様に心からお礼を申しあげます。特に、東京大学の平木敬教授、京都大学の富田眞治教授と五島正裕助手、豊橋技科大学の中島浩教授、東京農工大学の中條拓伯助教授、熊本大学の久我守弘助教授に感謝致します。

参考文献

[1] James Laudon and Daniel Lenoski. The SGI Origin 2000: A CC-NUMA Highly Scalable Server. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pp. 241–251, June 1997.

[2] T. Lovett and R. Clapp. STiNG: A CC-NUMA Computer System for the Commercial Marketplace. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pp. 308–317, May 1996.

[3] Yule L. Yang, et al. Recursive Diagonal Torus: An interconnection network for massively parallel computers. In *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing*, pp. 591–594, December 1993.

[4] Yule L. Yang and Hideharu Amano. Message Transfer Algorithms on the Recursive Diagonal Torus. In *Proceeding of the 1994 International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 310–317, December 1994.

[5] 楊愚魯, 天野英晴, 柴村英智, 末吉敏則. 超並列計算機に向き結合網: RDT. 電子情報通信学会論文誌, pp. 118 – 128, February 1995. J78-D-I.

[6] Hiroaki Nishi and ohters. The JUMP-1 Router Chip: A Versatile Router for Supporting a Distributed Shared Memory. In *Proceedings of the 15th International Phoenix Conference on Computers and Communications*, pp. 158–164, March 1996.

[7] Hiroaki Nishi, et al. The RDT Router Chip: A Versatile Router for Supporting a Distributed Shared Memory. *IEICE transaction on Information and Systems*, Vol. E80-D, No. 9, pp. 854–862, September 1997.

[8] H. Nakajo, S. Ohtani, T. Matsumoto, Kohata M., K. Hiraki, and Y. Kaneda. An I/O Network Architecture of the Distributed Shared-Memory Massively Parallel Computer JUMP-1. In *Proc. of the 1997 International Symposium on Supercomputer*, 1997.

[9] 小畑正貴, 中條拓伯. 超並列計算機 JUMP-1 におけるハイビジョン画像表示システム. 情報処理学会研究報告, 計算機アーキテクチャ研究会, pp. 17–23, 10 月 1994.

[10] 五島正裕ほか. 細粒度プロセッサ間通信をサポートする高機能キャッシュシステム. 情報処理学会研究報告, 計算機アーキテクチャ研究会, pp. 121–128, 8 月 1993.

[11] 天野英晴. 並列コンピュータ. 昭晃堂, 1996. ISBN 4-7856-2045-5.

[12] Tomohiro Kudoh, et al. Hierarchical Bit-map Directory Schemes on the RDT Interconnection Network for a Massively Parallel Processor JUMP-1. In *Proceedings of the International Conference on Parallel Processing*, Vol. I, pp. 186–193, August 1995.