1 はじめに

ワークステーションクラスタ (WS クラスタ) は，複数の WS を高速結合網によって接続し，マルチジョブによる高速化に加えて単一ジョブの高速化をも実現するシステムである．並列計算機と異なり，価格に応じたスケールアップが容易で手軽に導入することができることから多くのシステムが商用化されている．しかし，現在の商用 WS クラスタの多くは共有メモリを持たないため，並列プログラムの開発，移植性の点で問題があり，WS クラスタ上で共有メモリを実現するためのさまざまな試みがなされている．

純粋にソフトウェアにより共有メモリを実現する SVM (Shared Virtual Memory) として，IVY[5] や SSS-CORE[12] がある．これらの方法はハードウェアに改変を加える必要はないが，OS を全面的に変更する必要がある．一方で，SHRIMP[7] や Typhoon[2][4] 等では，汎用インターフェースをもつ付加ハードウェアを搭載することにより性能の向上を図っている．

我々も簡単な付加ハードウェアにより，SUN WS クラスタ上に分散共有メモリを実現するシステム JUMP-1/3(SUN) を開発している [8][10]．JUMP-1/3 では，SUN WS の SBus に SBus 標準サイズのボード (DPM:DSM Protocol Management ボード) を接続することにより，OS にほとんど手を加えずにキャッシュ機能を持つ分散共有メモリ環境を実現する．DPM ボードには 2MByte の SRAM と，これを分散共有メモリとして利用するための管理を行なうマイクロプロセッサが搭載されている．DPM ボードは，一般の SVM と同様にボード上のメモリをページ単位で管理し，その一部を他のノードの DPM ボード上のメモリのキャッシュ領域として利用する．その一方で，キャッシュライン単位に用意されたタグを用いてアクセスされたラインが有効であるかを高速に判別する機構を持つ．また，WS 間のデータ転送はライン単位で行なう．WS 間の高速なパケット転送を可能とするため，JUMP-1 用に開発された RDT ルータチップ [6] を搭載した RDT ボードを DPM ボードの後部に接続し，高速大容量転送を行なうことができる．

しかし，JUMP-1/3 の開発を進め，実機のデータに基づくシミュレーションを行なった結果，以下の点が明らかになった．

- 結合網の転送遅延は，ノード数が 64 程度の場合は性能に大きな影響を与えない．このため，RDT ボードはオーバスペックである．
- JUMP-1/3 の当初のプロトコルは，共有ラインに対する書き込み時にプロセッサをストールさせる Sequential Consistency を用いていた．しかし，評価の結果，この方法は Write ミスのペナルティで性能を大きく低下させることがわかった．

そこで，この結果に基づき，今まで開発してきた JUMP-1/3 に以下の改造を加えた．

- RDT ボードを用いず，DPM ボード上に搭載した STAFF-Link により結合網を構成する．このことにより，容易かつ低コストで WS クラスタを構成することができる．
- DPM ボード上に搭載された DSP のソフトウェアにより Partial Store Ordering を実現した．

本報告はこの二つの改造を施したシステム JUMP-1/3-SL(STAFF-Link) について述べ，この実機による測定結果に関して報告する．

2 JUMP-1/3-SL の構成

2.1 JUMP-1/3 の構成とその問題点

JUMP-1/3 は図 1に示すように SUN WS 本体に 2 つのボードを接続することにより構成した．

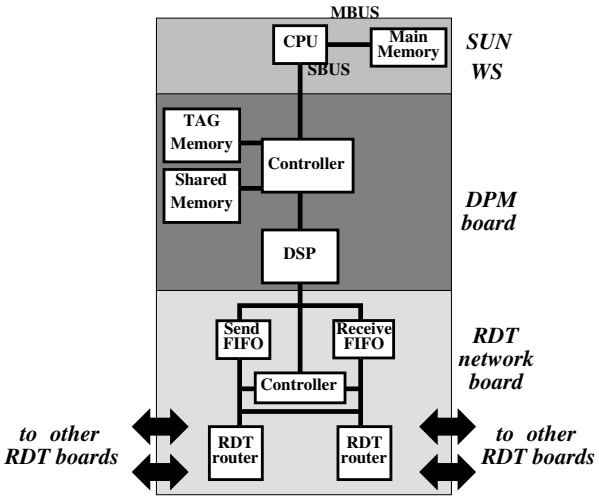

図 1: JUMP-1/3 のノード構成

このうち，DPM ボードは SBus 標準カード上に，分散共有メモリの実体，SBus インターフェース，プロトコル管理用プロセッサ (TI TMS320C40)，ネットワークインターフェースが搭載されている．

一方，RDT ボードは RDT ルータチップ 2 個 (36bit 幅のリンクを 18bit2 組にビットスライスして用いる)，DPM ボードとのデータの送受信用の FIFO，およびこのコントロールを行なう QuickLogic 社の FPGA から構成されている．パケットの生成および分解は，DPM ボード上の DSP によって行なわれ，RDT ボードには，RDT ルータと RDT-DSP 間のインタフェースが実現されている．RDT ルータは ECL 入出力を持ち，直接 2m 長の同軸ケーブルをドライブして他のノードとの間でデータ転送を行なうことができる．RDT ルータチップは本来 60MHz のクロックで動作するが，ここでは Quicklogic の動作速度および DPM ボードの処理速度を考慮し，25MHz で動作させている．クロックおよびリセット信号は専用のボードから同軸で供給され，PLL 素子を用いて位相を合わせている．

JUMP-1/3 では，DPM ボードによって管理される共有メモリは DPM ボード上のメモリに限定され，WS 上のメモリはそ

れぞれのノードのローカルメモリとなる. もちろん, WS 上のメモリを用いてソフトウェアによる SVM を実現することは可能であり, ここにテキストなどの読みだし専用の領域を割り付ければ, DPM ボードと合わせて効率の良い SVM を持つ WS クラスタを実現できる.

JUMP-1/3 の分散共有メモリ構成法は, 超並列計算機 JUMP-1 用に提案された方法 [11] に基づいている. すなわち DPM ボード上のメモリはホーム領域とキャッシュ領域に分割され, このうちホーム領域が分散共有メモリを構成する. タグはホーム領域とキャッシュ領域の両方に対しキャッシュライン毎に保持する. ある WS が別の WS のホーム領域をアクセスした際には, そのデータを自分のキャッシュ領域にコピーして利用する. メモリの共有単位はページ (4KByte) 単位であるが, false sharing を回避するために, アクセスの起こったキャッシュラインのみを転送する.

さて, JUMP-1/3 は上記の構成で, 各部の動作の検証を行なうと共にシミュレーションによる評価を行なった結果, 以下の問題点が明らかになった.

実装上の問題: RDT ボードが, SUN WS の筐体の外側に付加する形になり, さらにケーブルが大量に必要なことから, 全システムをラック中に格納する必要が生じる. 4 ノードのシステムの外観を図 2に示す. 図中に示すようにケーブルの接続は複雑で困難であると共に, RDT ルータチップの動作に必要な–5V, –2V の電源, GND レベル合わせ, クロック配分等の配慮も必要になる. このため JUMP-1/3 は机上の WS に付加ボードを差して気軽に構成できるシステムとはかけ離れた形態になっている.

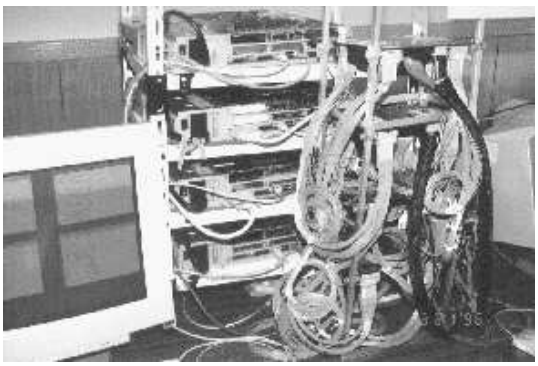

図 2: JUMP-1/3

性能上の問題: JUMP-1/3 の実測データに基づきシミュレーションにより評価を行なった [10]. この結果, ある程度の台数効果は得られるものの, CPU の稼働率は低いことがわかった. この原因を解析した所, 書き込みミス時のストールによる待ち時間が, すべてのアプリケーションでロス時間の半分以上を占めることがわかった.

2.2 JUMP-1/3 SL

まず, JUMP-1/3 の実装上の問題を解決するために, JUMP-1/3 の DPM ボード上に実装されたシリアルリンクである STAFF-Link を用いて WS クラスタを構成することにした. また, 性能上の問題を解決するため, DPM 上のプロトコルプロセッサのソフトウェアにより Partial Store Ordering を実現し, 書き込みミスが直接プロセッサのストールを招かないように改良した. このシステムを JUMP-1/3-SL と呼ぶ.

JUMP-1/3-SL は, SBus カードサイズの DPM ボード単体に簡単な STP(シールド付きツイストペア) ケーブルによるシリアルリンクを付加するだけでシステムを構成可能であることから JUMP-1/3 のように専用のラックや電源を必要としない. DPM ボードは STAFF-Link を 3 セットを持っており, WS クラスタの接続は, リング等の簡単な直接網となる. このため, 机上で用いられている WS を接続して小規模な WS クラスタを構成することが可能である.

2.2.1 DPM ボードの構成

JUMP-1/3-SL の DPM ボードのブロック図を図 3に示す. 基板レイアウト自体は, JUMP-1/3 用 DPM ボードと同一であるが, JUMP-1/3 では未実装であった STAFF-Link インタフェース部が実装されている.

DPM ボードはプロトコル制御用プロセッサ, 分散共有メモリ部, SBus インタフェース部, STAFF-Link 制御部から構成される. 以下, 構造と機能を述べる.

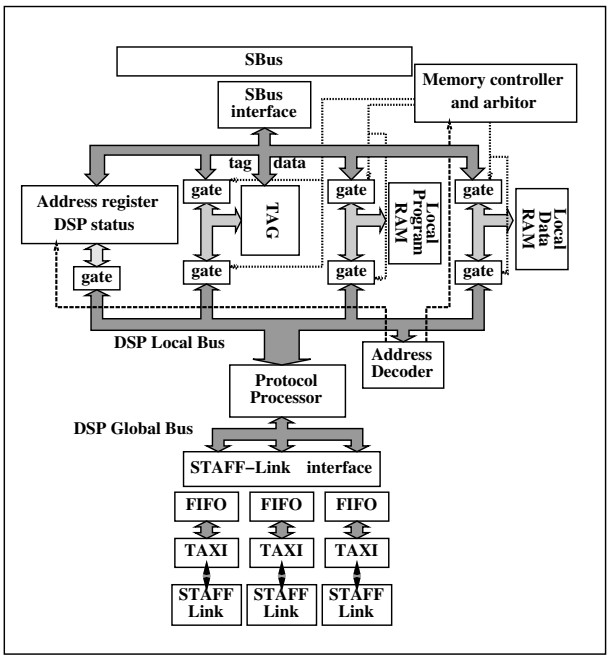

図 3: DPM ボードのブロック図

プロトコル制御用プロセッサ:　プロトコル制御用プロセッサとして，汎用 DSP(Digital Signal Processor)(TI TMS320C40) を採用した.TMS320C40 は以下の特長を持つ.

- 2 組のバスを持ち，独立にアクセス可能である. この構成は SBus 側とネットワーク側で頻繁にデータをやりとりする DSM プロトコル制御に適している.
- 内蔵の DMA 機能により高速転送が可能である.
- 内蔵のタイマにより，ネットワークの転送性能などのパフォーマンスの計測が可能である.

TMS320C40 は，バスサイクル 25MHz，内部周波数 50MHz で動作する. クロックは，DPM ボード上に実装され，SBus からのクロックとは独立である. プロトコル制御プロセッサの命令やアドレス変換テーブルを格納するため，LPR(Local Program RAM) が 2Mbyte 設けられている. LPR はアクセス 20nsec の高速 SRAM で実装され，内容はシステムの立ち上げ時に，WS から SBus を経由して書き込まれる.

分散共有メモリ部:　分散共有メモリとして用いられる L-DR(Local Data RAM) を 2Mbyte 持つ. 先に述べたように JUMP-1/3 同様 JUMP-1/3-SL でも，OS の改変を避けるため，分散共有メモリとして用いることができるのは LDR のみである. このうち 1Mbyte はホームメモリとして，残りの 1Mbyte は他ノードの LDR のキャッシュ領域として利用される. LDR は，プロトコル管理用プロセッサからアクセスされると共に，SBus を介して WS からもアクセスされる.

LDR の各ラインに対しては, 別に 4bit ずつタグメモリを設ける. JUMP-1/3-SL では，プロトコル制御プロセッサのソフトウェアにより様々なプロトコルが実現可能であり，タグメモリの利用法も様々であるが，3bit タグ (shared/exclusive，owner/not owner，valid/not valid) を用いる方法 [10] が最も標準的である. WS が LDR にアクセスを行なうとタグは自動的に参照され，Read Hit の場合，アクセスされたデータはプロトコル制御プロセッサを介することなしに，WS に送られる. その他の場合は，DSP に対して処理を要求する割り込みがかかる. LDR，タグメモリ共にアクセス 20nsec の高速 SRAM を用いている.

SBus コントローラ　SBus コントローラは，SBus を介して WS 側から LPR，LDR，タグメモリをアクセスするための制御を行なうと共に，タグ参照と Hit 時のデータ転送，プロトコル制御プロセッサのアクセスとの調停を行なう点で DPM ボードのハードウェア的な中央制御機構の役割を担っている. SBus コントローラは高速アクセスが必要なため，QuickLogic 社の FPGA QL24x16B1 上に実装されているが，これはアンチヒューズ型でダイナミックな変更を行なうことができない. そこで制御に柔軟性を持たせるため，DPM ボード上には，補助的に Xilinx 社の FPGA XC3190 を搭載している. XC3190 は，SBus を介して内部の配線情報を転送することが可能で，LDR，LPR のアドレスラッチを行なう他，内部構造の設定によっては，QL24x16B1 に代わって柔軟な制御を行なうことができる. ただし，柔軟性と引きかえに動作速度は犠牲になる.

STAFF-Link　STAFF-Link は JUMP-1/3-SL で新たに実装された部分である. STAFF(Serial Transparent Asynchronous First-in First-out) -Link[9] は，並列システムの入出力インタフェースや，狭域，広域ネットワーク用に開発された高速シリアルリンクである. STAFF-Link は，パラレル / シリアル変換と高速データ転送を行なう機能を持つ LSI(TAXI チップ [1]) に，送信用 / 受信用の 2 つの FIFO，さらに FIFO が溢れないようにハンドシェーク操作を行なう通信コントローラから構成される. このような構成により，リンクの両端には仮想的に双方向の FIFO が形成され，透過な通信路を構成することができる. ハードウェアの性能を最大限に用いた場合，140Mbps の転送レートを実現することができる.

この DPM ボードでは STAFF-Link が 3 本装備され，コントローラには Xilinx 社の FPGA XC3164 を用いている. SBus コントローラに用いた FPGA 同様，このコントローラのコンフィギュレーションも，SBus 経由で行なうことが可能である.

DPM ボードの外観を図 4に示す. SBus カードサイズに収めるため，表面実装部品を多用している. 上辺の 3 つの金属性のコネクタが STAFF-Link 用で，WS の背面から外に突き出される. ここに STP ケーブルを接続する.

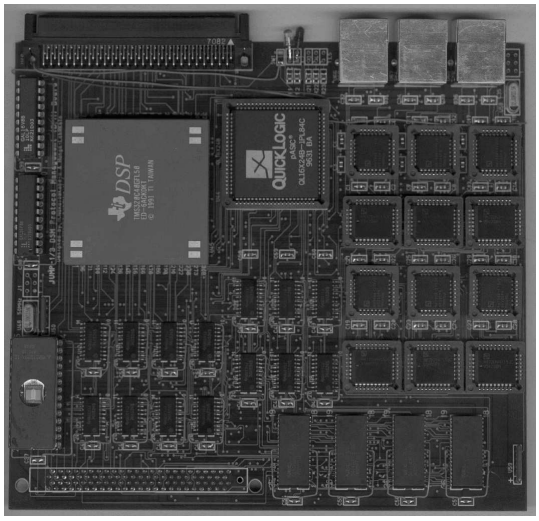

図 4: DPM ボード

3　Partial Store Ordering の実現

JUMP-1/3 の分散共有メモリプロトコルは，超並列マシン JUMP-1 の分散共有メモリ管理法に基づいている. ただし，ページの共有情報であるディレクトリはフルマップ方式を用いて管理する.これは，共有情報をビットマップとして保持する方

式で，システムのサイズが大きくなると，必要なビット数 (メモリ量) が増加してしまうが，JUMP-1/3 では最大 64 台を前提としているため，問題はない. 共有情報の管理はホームノードでのみ行う. そのためキャッシュミスした場合はすべてそのページのホームノードに要求が送られることになる. ディレクトリを含めて，ソフトウェアやプロトコル管理に必要なページに関する情報はすべて GPMT(Global Page Map Table) と呼ばれるページテーブルに格納される.

JUMP-1/3 のメモリ管理は，JUMP-1 同様 OS の仮想記憶のページ管理機構を利用している. DPM ボード上の共有メモリのページは，SPARC の通常のメモリと同様に MMU により管理される. プロセスからは，WS 上の通常のメモリ (ローカルメモリ) と JUMP-1/3 の全 WS の DPM ボード上の共有メモリは，同一論理アドレス空間に見える. プロセスが共有メモリをアクセスした場合，そのページが DPM ボード上に存在すれば MMU により，論理アドレスが DPM ボード上のメモリの物理アドレスに変換される. 次にこのアドレスをアクセスすると，SBus コントローラがタグメモリをアクセスし，そのノードが有効なラインを持っているかどうかを調べる. 無効な場合は，DSP はミスしたアドレスから, LPT(Local Page Table) によりページ番号を調べ，GPMT を参照する. LPT には，そのノードがキャッシュしているページ番号が格納されている. GPMT にはホームクラスタの WS の番号やホームノードにおける物理アドレスの番号などが格納されており，さらにホームクラスタにはそのページを共有するディレクトリがフルマップ (最大 64bit) で格納されている. この GPMT を参照することによって，DSP はそのページのホームノードでの DPM ボード上のアドレスを知ることができる. このように JUMP-1/3 では，2 段階のアドレス変換を行っている.

JUMP-1/3 の分散共有メモリの管理手法の詳細については，文献 [10] を参照されたい. ここでは，JUMP-1/3-SL で新たに導入した PSO の実現についてのみ解説する.

3.1 PSO (Partial Store Ordering)

JUMP-1/3 での性能の低下の大きな原因は全ての Write Request に対して，応答が返って来るまでストールさせた点にある. このため，JUMP-1/3-SL では，同期時を除いて Write Request を一定数連続して出すことができるプロトコルを用いる. このプロトコルは，SPARC Version 8 の命令セットで定義している PSO の実現を目的としている.

PSO は以下の順序のみを保証するコンシステンシィモデルである [3].

- $Read \rightarrow Read$,$Read \rightarrow Write$
- Store Barrier 等同期命令，Read，Write の順番

すなわち同期命令をはさまない Write 同士 ($Write \rightarrow Write$)，Write の後の Read($Write \rightarrow Read$) の順番は保証されない.

3.2 Transaction Buffer (TB) の動作

PSO の実現のためには，書き込んだデータを保存しておくための Transaction Buffer (TB) が必要である. ここでは，プロトコル制御用プロセッサのソフトウェアにより LPR 上に 16 エントリ分の TB を実現した. TB の構造を図 5に示す. 図に示すように，1 ライン分のデータ領域，ラインアドレスおよびラインの各ワードの有効 / 無効を示すビット列を持つ.

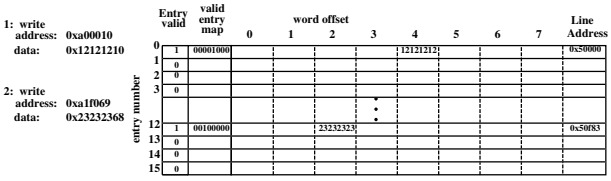

図 5: Transaction Buffer の構成

TB の参照はキャッシュラインアドレスの下 4bit で参照し，別のラインによりエントリが埋まっていれば，その場所からサーチする. 利用可能なエントリがない場合，そのアクセスはブロックされる. TB は以下の作業を行なう.

- 書き込まれたデータを記憶する. 同一ラインに対する複数の書き込みが連続して行なわれた場合，それらをすべて上書きして記憶する.
- 他のノードから，読み出されたラインのデータに対して，書き込まれたデータを上書きする.
- 有効なデータの読み出しアクセスに対しては直接応答する.

無効化型プロトコルにおいて，Dirty の状態のキャッシュに対する書き込みは LDR にデータを書き込むだけで，プロトコル制御プロセッサを介さずにすぐに WS の SPARC に制御を返すことができる.

TB が動作するのは，(1)Write Miss した場合，(2)Shared ラインに対して Write Hit した場合の 2 つの場合である. JUMP-1/3 のプロトコルは Fetch-on-Write 型である [10] ので，両方の場合について，基本的には以下のように動作する. ただし，JUMP-1/3-SL においては STAFF-Link の転送には FIFO 性が保証されている.

1. ローカルノードに書き込みが起きると，プロトコル制御プロセッサに割り込みがかかる. プロトコル制御プロセッサは，TB にエントリを作成し，書き込まれたデータを記憶する. そして WS の SPRAC に対しては Acknowledge(ackwr) を返し，処理を先に進めることを許す. 一方，ホームメモリに対して EREQ(Exclusive Read Request) メッセージを送る.

2. EREQ を受け取ったホームノードは，GPMT の参照によりそのラインの状態により以下の操作を行なう.
   - ラインの状態が Exclusive であればなにもしない.

- ラインの状態が Dirty であれば，Owner のノードからラインを書き戻すと共に Owner を無効化する．
- ラインの状態が Shared であれば，共有するキャッシュを無効化する．

その後，ラインの状態を Dirty として，ERPLY(Exclusive Read Reply) メッセージと共にラインのデータを返す．

3. ERPLY を受け取ったローカルノードは，受け取ったラインのデータに対して TB の内容を上書きした後，LDR 上のキャッシュ領域に格納する．最後に TB をクリアする．

上記の基本動作を図 6に示す．ここではローカルノードのキャッシュの状態は Invalid，ホームノードは Exclusive 状態の場合である．ここで，PSO では，ローカルノードが ERPLY を待っている間，様々な状況が発生する可能性がある．

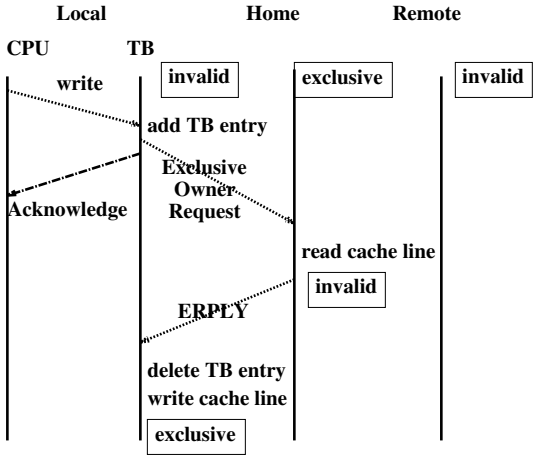

図 6: 無効化型プロトコル基本動作

- TB 上のラインに対してデータの書き込みが起こった場合:
  TB 上に上書きし，WS の SPARC に対しては即座に Acknowledge 信号を返す．
- TB 上のラインに対してデータの読み出しが起こった場合:
  TB 上に有効なデータが存在すれば，それを読み出し，WS の SPARC に対しては即座に Acknowledge を返す．TB 上に格納されているラインと同一ライン上だが有効データは存在しない場合，ERPLY メッセージにより有効なデータが到着するまで待つ．ローカルノードから連続して write が 2 つと read が 1 つ送られた場合のメッセージのやりとりを図 7に示す．
- TB 上のラインアドレスに対して結合網から無効化メッセージ (IR) が到着した場合:
  TB はこれを無視して応答メッセージ (ACKIR) のみ送信する．結合網の FIFO 性から，この無効化メッセージは他のノードの書き込み要求によるものか，自分のノードの EREQ によるものかどちらかである．いずれにせよ，自分の書き込み要求により，後で同一ラインのコピーは全

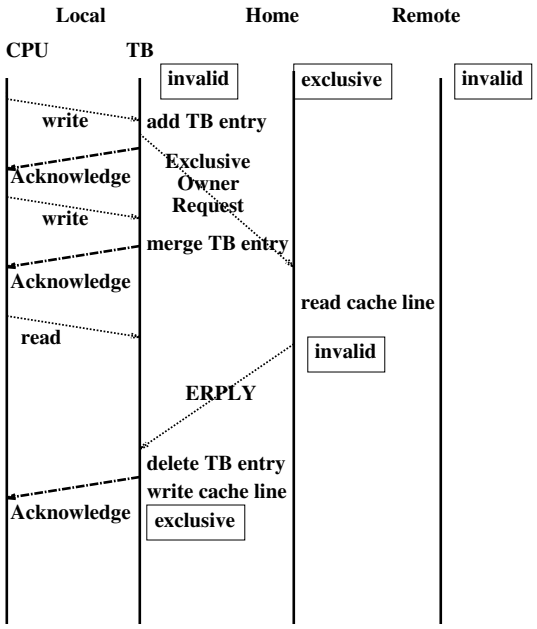

図 7: 連続した write，read に対するプロトコル動作

て無効化されるため，データの一貫性は維持される．他ノードの書き込みが同時に起こった場合のメッセージのやりとりを図 8に示す．

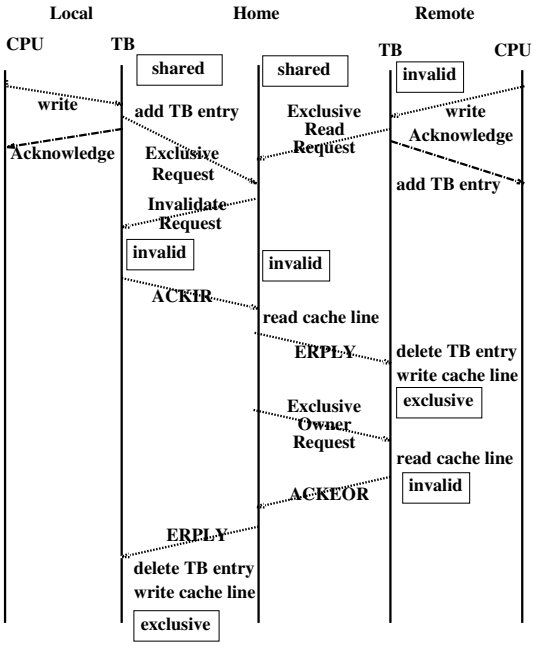

図 8: ホームでの write の競合時のプロトコル動作

4 評価

現在，JUMP-1/3-SL は，DPM ボード 2 枚が稼働している状態で，共有メモリを構成してアプリケーションを動かすには至っていない．ここでは，実測データに基づき，改造の効果について示す．

4.1 STAFF-Link の転送速度

JUMP-1/3 で標準的なパケットサイズである 32bit × 16flit のパケットを送受信およびルーティングする時間を測定した. STAFF-Link はシリアルリンクであるが，転送用チップにより 9bit 単位のデータ幅に変換される. パケット構成は，ルーティング用情報のヘッダ 9bit の後はデータ長に相当する flit 数が 8bit 単位に折り畳まれて格納される. このパケットの送受信時間を表 1に示す. ここで，受信時間の括弧内の数字はハンドシェークをソフトウェアで行なった場合の転送時間である. また，参考のため JUMP-1/3 で用いていた RDT ボードを用いた場合の転送時間を示す.

表 1: パケット転送時間 (ns)

結合網	送信	受信	他ノードへの転送
STAFF-Link	3380	2280 (9960)	3240
RDT	700	800	100

RDT ボードを用いる場合，LDR からボード上の FIFO にソフトウェアで転送を行ない，さらにボード上の QuickLogic 社 FPGA を用いて転送を行なうため，JUMP-1 で直接 RDT ルータチップを用いる場合に比べ，必要な時間が大きくなっている. STAFF-Link による転送は，Point-to-Point の転送ならばこれに比べて 3-5 倍程度で済んでいることがわかる. また，複雑な構造の RDT 用のパケットを作る時間を考えると，STAFF-Link に切替えたことによる損失は少ないと考えられる. STAFF-Link は他ノードへの転送を行なう場合にも，パケットを受信してヘッダを解析しただけで，他のノードにフォワーディングする Virtual Cut Through 方式を実現できる. しかし，これはソフトウェアが介在する必要があるため，ルータ内で Asynchronous Wormhole ルーティングを行なう RDT ルータに比べると差が大きい. しかし，テーブル参照等により読みだしミス時にパケットを発生するのに必要な時間を考えると，1 ノード当たり 3.4μsec 程度の遅延は，システムサイズが小さければ大きく不利になることはないと考えられる.

4.2 TB 操作に要する時間

プロトコル制御プロセッサのソフトウェアで TB を実装し，その操作時間を測定した. TB の操作時間は，次の二つの場合で異なってくる. 一つは，既に entry が存在しており新たなデータと，valid entry map を書き変えるだけの場合であり，もう一つは，TB に entry が存在せず，TB に entry を加えてさらにデータと valid entry map を作成する場合である. それぞれの場合について実行時間の測定結果を示す.

この結果は，TB をラインアドレスの下位 4bit で参照したバッファに書き込むことができた場合の数値である. 別のラインによってエントリが埋まっており，他の TB を参照する場合，一度参照する毎に 560ns を余分に必要とする. しかし，一般的なアプリケーションでは，多くの場合，連続した Write が連続したアドレスに対して行われるので，このペナルティを受ける可能性はさほど高くない.

表 2: TB を操作する時間 (ns)

	存在する	存在しない
割り込みの発生までの時間	252	252
TB の操作に要する時間	3480	3600
アクノリッジを返す時間	252	252
合計	3984	4104

JUMP-1/3 で Sequential Consistency を用いていたときには 1word の書き込みに平均で 40μsec 以上かかっていたのに対し，10 分の 1 以下の時間で書き込みが終了することができる. そのために，Write ミスのペナルティが小さくなり，プロセッサのストール時間が大幅に減少することが期待できる.

5 おわりに

WS クラスタ上に分散共有メモリを実現するシステム JUMP-1/3 に改良を加えた JUMP-1/3-SL に関して述べた. 結合網を RDT から STAFF-Link に変更することにより，SBus ボードに対してシリアルリンクを接続するだけで簡単に WS クラスタが構成できるようになった. これによりパケット送受信時間は 3-5 倍程度に大きくなるが，システムが小規模であれば損失はさほどに大きくならないことが予想される.

また，JUMP-1/3 における性能低下の大きな部分が書き込み時のストールによるものであることを考え，TB(Transaction Buffer) を導入し，PSO(Partial Store Ordering) を実現した. TB の実現はプロトコル制御プロセッサのソフトウェアで行なうため，Write 時には約 4μsec ストールが起きるが，ホームノードからの応答を待つ必要がなくなるため，大幅な性能の向上が期待できる.

現在，JUMP-1/3-SL は基板が 2 枚動作するにすぎない. さらに開発を進め，実際に分散共有メモリを構成して評価を行なうと共に，大規模システムについての評価をシミュレーションにより行なう予定である.

謝辞

プロトコル制御プロセッサのプログラム開発環境は TI 社の University Program を利用しました. 深く感謝します. また，この研究の一部の費用は，並列・分散処理研究推進機構により援助されたものです.

参考文献

[1] *Advanced Micro Devices, Inc.: Am7968/Am7969-175 TAXI-175 Transmitter/Receiver Data Sheet and Technical Manual*, 1992.

[2] Mark D.Hill, James R.Larus, and David A.Wood. Tempest: A Substrate for Portable Parallel Programs. In *COMPCON '95*, pp. 327–332, Spring 1995.

[3] SPARC International Inc. *The SPARC Architecture Mannual Version 8*. Prentice-Hall, Inc., 1991.

[4] Steven K.Reinhardt, James R.Larus, and Dabid A.Wood. Tempest and Typhoon: User-Level Shared Memory. In *21st ISCA*, pp. 325–336, 1994.

[5] K. Li. A Shared Virtual Memory System for Parallel Computing. In *Int. Conf. on Parallel Processing, St. Charls, IL*, pp. 94–101, August 1988.

[6] Hiroaki Nishi, Katsunobu Nishimura, Ken ichiro Anjo, Hideharu Amano, and Tomohiro Kudoh. The JUMP-1 Router Chip: Versatile Router for Supporting a Distributed Shared Memory. In *Proc. of 15th IPCCC*, pp. 158–164, 1996.

[7] Edward W.Felten, Richard D.Alpert, Angelos Bilas, Matthias A. Blumrich, Douglas W. Clark, Stefanos N. Damianakis, Cezary Dubnicki, Liviu Iftode, and Kai Li. Early Experience with Message-Passing on the SHRIMP Multicomputer. In *23rd ISCA*, pp. 296–307, 1996.

[8] 安生健一朗, 西宏章, 董小社, 吉山晃, 工藤知宏, 中條拓伯, 天野英晴. 超並列計算機用結合網 RDT のルーティング制御評価用システム:JUMP-1/3. 電子情報通信学会技術報告, August 1996.

[9] 中條拓伯, 中野智行, 松本尚, 小畑正貴, 松田秀雄, 平木敬, 金田悠紀夫. 分散共有メモリ型超並列計算機 JUMP-1 におけるスケーラブル I/O サブシステムの構成. 情報処理学会論文誌, Vol. 37, No. 7, pp. 1429–1439, July 1996.

[10] 安生健一朗, 中條拓伯, 小野航, 工藤知宏, 山本淳二, 西宏章, 木透徹, 天野英晴. 分散共有メモリを持つ WS クラスタ:JUMP-1/3. 並列処理シンポジウム JSPP'97 論文集, May 1997.

[11] 松本尚, 平木敬. Memory-Based Processor による分散共有メモリ. 並列処理シンポジウム JSPP'93 論文集, pp. 245–252, May 1993.

[12] 松本尚, 駒嵐丈人, 渦原茂, 竹岡尚三, 平木敬. 汎用超並列オペレーティングシステム :SSS-CORE — ワークステーションクラスタにおける実現 —. 情報処理学会オペレーティングシステム研究会報告, Vol. OS73-20, pp. 115–120, August 1996.