2003－ARC－153 (5)
2003／5／8

並列計算機 JUMP-1 の MBP-light の命令セットアーキテクチャの評価

鈴 木 紀 章† 天 野 英 晴†

並列計算機 JUMP-1 の分散共有メモリ管理プロセッサ MBP-light の命令セットアーキテクチャの評価を、実機を用いて行った。その結果、MBP-light に導入された Buffer-Register Architecture は、ホームクラスタで 5.64%、リモートクラスタで 6.27%の性能向上を達成した。特殊命令では、クラスタアドレスのハッシュ値を求める命令によって 2.80%の性能向上を得る事ができたが、他の特殊命令はあまり活用されていない。これらは、DSM 管理プログラムの主要な動作が、ローカルメモリ上のパケットキューの操作である事に起因する。従って、標準 RISC 命令の利用率が高く、特にロード、ストア命令の比率が高い。このため、命令メモリとデータメモリを分離することができた場合、約 33%の性能向上が達成できることが分かった。

Performance Evaluation of Instruction Set Architecture of MBP-light in JUMP-1

Noriaki Suzuki† and Hideharu Amano†

The instruction set architecture of MBP-light, a dedicated processor for the DSM (Distributed Shared Memory) management of JUMP-1 is analyzed with a real prototype.

The Buffer-Register Architecture proposed for MBP-core improves performance with 5.64% in the home cluster and 6.27% in a remote cluster. Only a special instruction for hashing cluster address is efficient and improves the performance with 2.80%, but other special instructions are almost useless. It appears that the dominant operations in the DSM management program were handling packet queues assigned into the local cluster. Thus, common RISC instructions, especially load/store instructions, are frequently used. Separating instruction and data memory improves performance with 33%.

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、中，大規模な並列計算機の代表的な構成方法であり、多数のプロセッサを接続可能であると共に、共有メモリを利用したプログラム開発も容易である。このため，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 年から開発された 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[3]。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、このために様々な新しい構成を採用している。特に、分散共有メモリ制御用チップ MBP-light は、RISC プロセッサを内蔵しており、ソフトウェアの変更により様々なキャッシュコヒーレンスプロトコルを実現することができる。

JUMP-1 は，2000 年 3 月に 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、実機を用いた評価が可能になった。本論文では、JUMP-1 の分散共有メモリ制御用チップ (MBP-light) の命令セットアーキテクチャについて、実機を用いて評価する。第 2 章で JUMP-1 の全体構成を紹介し、MBP-light の構成を第 3 章で、MBP-light の命令セットアーキテクチャについて第 4 章で述べる。第 5 章で実機評価とその結果について述べる。

2. JUMP-1 の構成

JUMP-1 は数千プロセッサ規模への拡張を可能とするために結合網 Recursive Diagonal Torus(RDT)[4] で相互接続されたクラスタ構造を持つ。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、メッセージマルチキャストと応答収集機能を持つルータチップを用いて実現する。さらに、各クラスタは並列 I/O サブ

† 慶應義塾大学 理工学研究科
Department of Computer Science, Graduate School of Keio University

システムを構成する高速シリアルリンク STAFF-Link[9] により、ディスクおよび画像データ用 Frame Buffer(FB) と接続する。また、各クラスタに対してメンテナンス用ホスト PC を設け、システムのブートアップやモニタを行う。

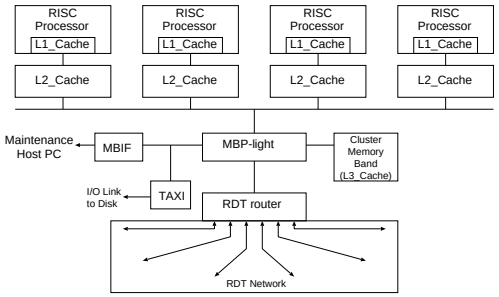

図 1 JUMP-1 クラスタの構成

図 1 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[5] 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に実装された FPGA の MBIF (Maintenance Bus InterFace) 等も接続されている。

2.1 JUMP-1 の分散共有メモリ管理

JUMP-1 と、それ以前の CC-NUMA である DASH / FLASH[10]、Alewife[11]、NUMA-Q[2] との相違点は、分散共有メモリ管理手法である[6]。これまでの CC-NUMA では、キャッシュライン単位で管理が行われ、リモートノードのデータは各プロセッサのキャッシュに転送される。キャッシュコヒーレンスプロトコルには単純な無効化プロトコルが使用され、結合網はメッシュやリングであり、1 対 1 のデータ転送によってディレクトリ管理が行われる。

これらの手法は限られたプロセッサ数では有効であるが、プロセッサ数が多い場合には大きなキャッシュやディレクトリのメモリ容量を必要とする等、大きなコストを要する。また、複数のプロセッサが同一のキャッシュラインを保持する場合、1 対 1 転送を用いた無効化プロトコルでは、ネットワークを混雑させてしまう。

これらの問題を解決するため、JUMP-1 では下記の手法を採用している。

(1) 各プロセッサは、3 レベルの TLB を用いてグローバルな仮想アドレス空間を共有する。ディレクトリはページ毎に管理され、データ転送はキャッシュライン単位で行われる。クラスタメモリの一部は L3 キャッシュとして機能する。

(2) 無効化プロトコル、アップデートプロトコル等、様々なキャッシュコヒーレンスプロトコルを利用可能である。

(3) Reduced Hierarchical Bitmap Directory(RHBD) 方式[12] を用いることによって、ディレクトリのビット数を削減する。また、RDT ネットワークは階層的な構造を持つため、RHBD のビットマップをパケットヘッダに利用することによって、各階層でのディレクトリアクセスを行わずに、パケットをマルチキャストすることができる。

これらのメカニズムは、MBP-light 上で動作する、京都大学が実装した DSM 管理プログラムによって実行される[7]。

3. MBP-light の構成

MBP-light は分散共有メモリ制御用コントローラチップであり，約 20 万ゲートの 0.4μm CMOS ASIC である．プロトコル制御を行う専用のコアプロセッサ MBP Core に，RDT ルータチップとのパケットの交信を制御する RDT Interface，クラスタメモリと共有バスの制御を行う MMC(Main Memory Controller) の二つのハードウェアモジュールを接続した構成を持つ．

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う．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おり，応答パケットの自動生成や自動収集機能を持つ[8]．MMC は CBus とのインタフェース，およびクラ

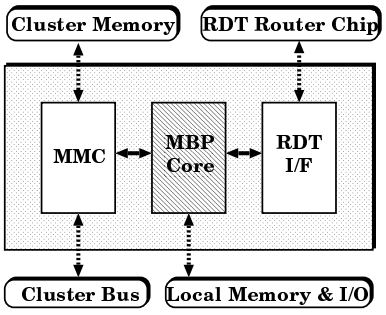

図 2 The Structure of MBP-light

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.

3.1 MBP Core

3.1.1 Buffer-Register Architecture

高速に行う必要がある処理は、RDT interface や MMC によってハードウェア処理されるため、MBP Core が扱うのは DSM プロトコルの一部である。その処理は、パケットを受信後 GPMT(Global Process Management Table) のアクセスを行い、アドレス変換を行った後に送信パケットを組み立てる事である。JUMP-1 のパケットヘッダはパケットの数フリットを占めており、パケットのデータ部にあるタグは、プロトコル制御等に用いられる。従って、全てのパケットバッファをレジスタとして扱う事ができれば便利であるが、バッファのビット幅は 68bit であり、これらを汎用レジスタとして扱う事は難しい。

そこで MBP Core では、16 個の 16bit 幅の GPR(General Purpose Register) と、112 個の 68bit 幅の PBR(Packet Buffer Register) を備えている。PBR 内のデータは、パケットとして直接 MMC や RDT Interface を通して送受信することができ、またレジスタとしてのアクセスも可能である。これらの特徴から、我々はこのアーキテクチャを *Buffer-Register Architecture* と呼んでいる。

3.1.2 PBR(Packet Buffer Register)

PBR はパケットを扱うための 68bit 幅の専用レジスタであり、汎用 PBR64 本、RDT パケットの送信バッファとなる To RDT PBR 24 本、受信バッファとなる From RDT PBR 24 本の 3 種類に分かれる。

To RDT PBR、From RDT PBR は、MBP Core と RDT Interface で共有している。これらは 3 エントリのパケット FIFO として機能し、パケットは 8 個の連続した PBR に格納される。To RDT PBR に格納されたデータは、MBP Core の特殊命令によって直接 RDT ネットワークへ送出する事ができる。FIFO 動作をする From RDT PBR のトップエントリに格納されたデータは、MBP Core から直接扱う事ができ、1 サイクルで次のエントリへ切り替える事が可能である。

また、各 PBR のデータは、MBP Core の命令によって、直接クラスタメモリとの間の転送を行う事ができる。

4. MBP-light の命令セットアーキテクチャ

表 1 は、MBP Core の命令セットを分類したものである。MBP Core には、標準的な命令の他に、RDT Interface や MMC 等を扱ういくつかの命令がある。また、プロトコル処理を高速化する特殊命令を扱う事もできる。

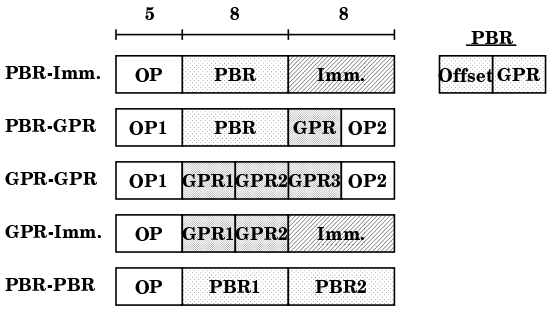

図 3 命令セットフォーマット

表 1 命令セットの分類

Class	Type
WGG	operate between GPR and GPR
LMA	access to local memory
BRANCH	branch
WGI	operate between GPR and Imm.
BPI	operate between PBR and Imm.
RDT	control RDT Interface
MPP	transmit from PBR to PBR
WPG	operate between PBR and GPR
MMC	control of MMC
TJ	table jump
INT	control interrupt
IMA	access to internal memory
SPE	special instructions
NOP	no operation

4.1 基本 RISC 命令

基本 RISC 命令は 3 オペランドを持つ命令で、WGG 命令 (レジスタ間演算命令)、WGI 命令 (レジスタ-直値間演算命令)、LM 命令 (ローカルメモリアクセス命令)、BRANCH 命令に分類される。GPR のうち、R0 は常に 0 であるため、MOVE 命令は存在しない。また、BRANCH 命令は単純なフラグを元に分岐する。

4.2 PBR 操作命令

PBR 操作命令は、WPI, WPG, MPP 命令に分類される。WPI 命令は、8bit の直値データと PBR 内の 8bit との間で演算を行う命令であり、パケットヘッダを操作する場合等に有効である。WPG 命令は GPR と PBR 内の 16bit との間での演算命令であり、PBR、GPR のどちらをデスティネーションにとることもできる。MPP 命令は 8bit、16bit、68bit(PBR 全体) 単位で PBR 間のコピーを行う命令である。

4.3 パケット、キャッシュライン転送命令

RDT Interface に対してパケットの送信、受信を指示する命令として、MBP Core は PRDT、GRDT 命令を

持つ。クラスタメモリとのデータ転送のために MMC を操作する命令にはいくつかの種類があるが、代表的なものとして、PBR とクラスタメモリ間でのデータ転送を行う TGM 命令がある。

4.4 特 殊 命 令

表 2 特殊命令

Instruction	Call
TJRQ	Table Jump with External Requests
TJ	Table Jump
HSHC	Hash Cluster Address
HSHG	Hash GPR
HSHN	Hash Net Address
GUN	Get Unique Number
BITS	Bit set
BITR	Bit reset
BITC8	Bit Count (8bit)
TJRI	Table Jump interrupt

MBP Core は高速な DSM 処理のための特殊命令を持っている。MBP Core の主要な役割の一つに、アドレス変換がある。高速なテーブル参照のために、MBP Core はローカルメモリを利用したテーブルジャンプ命令をサポートする。また、高速な割り込み処理のために、外部要求によってテーブルジャンプを行うこともできる。また、ハッシュ命令、ビット操作命令等も備えている。

5. 評　　価

5.1 評 価 環 境

MBP-light の命令セットアーキテクチャについて、DSM 管理プログラムのコンパイル結果と、実機で行列積プログラムを実行した場合に MBP Core が実行する DSM 管理プログラムのコードを元に検討する。

コンパイラ出力による検討では、MBP Core で実行される共有メモリ管理プログラム全体を MBP-light 用 C コンパイラでコンパイルし、アセンブリ出力に含まれる命令から検討を行なう。

行列積プログラム実行時の評価は、4 クラスタの JUMP-1 システム上でプログラムを実行した際に、MBP Core がどの命令を実行するかをカウントしたものである。

共有メモリ管理プログラムは大規模で多数の条件分岐を内包しており、完全な形で実行される命令をカウントする事は非常に困難であるため、ここでは行列積プログラム実行の際に発生するパケット処理の種類と数をカウントし、各パケットの処理ルーチン内においては例外的な処理が行なわれないものとして実行する命令を数えることによって命令数のカウントを行なった。

また、ローカルメモリの容量の関係からプログラム全体のパケットのログを取ることはできないため、最後の 100 パケットについてのデータを使用した。

5.2 各命令の比率

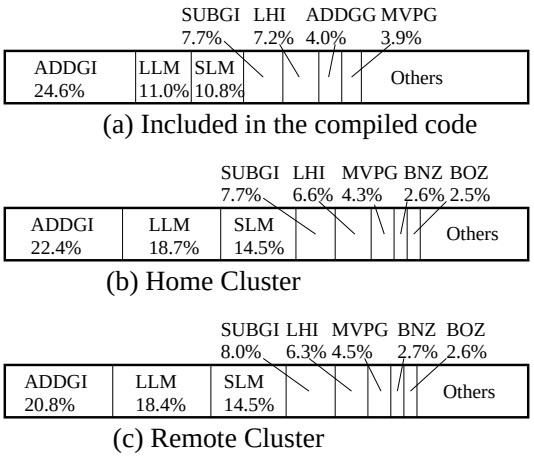

図 4 各命令の比率

ホームクラスタとリモートクラスタでは、同じ DSM 管理プログラムを実行するが、実行される処理は異なる。ここでは、ホームクラスタ (クラスタ 0) と、リモートクラスタの 1 つであるクラスタ 1 について図 4 に示す。

これらの結果から、PBR を扱う命令や特殊命令よりも、基本 RISC 命令の比率がかなり高い事が分かる。

表 3 ローカルメモリアクセスに関する比較

	1 Clock Stall	Without Stall	Impr.
Home	64471	48394	33.2%
CL1	43130	32448	32.9%

また、ローカルメモリアクセス命令の比率も高く、ホームクラスタで 33.2%、リモートクラスタで 32.9%を占める。これは、DSM 管理プログラムがパケットを退避する際にローカルメモリを使用している事に起因する。

MBP-light では、パッケージのピン制約の関係で命令メモリとデータメモリを分離しておらず、ローカルメモリアクセスには 1 クロックのストールを伴う。表 3 は、1 クロックのストールがある場合とない場合を比較したものである。この結果から、命令メモリとデータメモリの分離によって、30%超の性能向上が得られ

る事が分かったため、十分なピン数を確保できる場合は命令メモリとデータメモリを分離すべきである事が分かった。

5.3 PBR 操作命令

表 4 PBR 操作命令

Ins.	Compiled code	Home	Remote
ADDPI	0	0	0
SUBPI	0	0	0
SLLPI	0	0	0
SRLPI	0	0	0
ANDPI	10	0	38
ORPI	0	0	0
XORPI	0	0	0
LPI	40	52	3
CPI	0	0	0
ADDPG	9	56	42
SUBPG	0	0	0
ANDPG	41	454	330
ORPG	10	0	39
XORG	0	0	0
MVBPG	252	185	319
MVPG	909	2099	1476
MVBPP	46	2	0
MVPP	144	134	166
MVLPP	158	298	205

表 4 は、PBR 操作命令の数を、DSM プログラムのバイナリ中に含まれるもの、ホームクラスタやリモートクラスタで実行された物について調査したものである。これらの命令を使用する場合、特殊関数呼出しとしてプログラム中に記述されるため、使用される頻度が低く、あまり実行されていない。PBR のコピーを行う命令やビット操作系の命令は比較的利用されているが、数値、論理演算系の命令の多くはあまり利用されていない。

表 5 は、PBR に関する命令を他の命令で置き換えた場合の必要サイクル数を示す。これらは、以下の条件を元に算出した。

- パケットバッファは内部メモリとしてマッピングされる
- 内部メモリはバイトアクセス可能である
- 内部メモリは 1 サイクルでアクセス可能である

ADDPG、ANDPG、ORPG 命令は、デスティネーションが PBR か GPR かによって必要サイクル数が異なり、GPR がデスティネーションの場合は 2 サイクル、PBR の場合は 3 サイクルである。ここでは、簡単のため 2.5 サイクルとして定義する。

表 5 PBR 操作命令の他命令による置き換え

Ins.	Cycles	Ins.	Cycles
ANDPI	3	MVBPG	1
LPI	2	MVPG	1
ADDPG	2.5	MVBPP	2
ANDPG	2.5	MVPP	2
ORPG	2.5	MVLPP	10

表 6 PBR 操作命令の有無によるサイクル数の比較

	With PBR	Without PBR	Impr.
home	64471	68106	5.64%
CL1	43130	45836	6.27%

表 6 は、PBR 操作命令がある場合とない場合の必要サイクル数を比較したものである。今回の場合、ホームクラスタで 5.64%、リモートクラスタで 6.27%の性能向上を達成している。しかし、Buffer-Register Architecture のコストは DMA 機能付きのオンチップキャッシュ等に比べて小さいものの、DSM 管理プログラムの性能改善には大きく寄与していない事が分かった。

5.4 特 殊 命 令

表 7 は、特殊命令の数を、DSM プログラムのバイナリ中に含まれるもの、ホームクラスタやリモートクラスタで実行された物について調査したものである。

特殊命令は、コンパイラが直接扱うことができないため、対応する特殊関数呼出しによりこれらの命令を扱う。表 7 に示すように、今回の実装では、HSHC、GUN、BITS、BITR、BITC8 が利用されているが、HSHC 以外の命令はほとんど利用されていない。これは、他の特殊命令がハッシュ値の競合等の例外的な場合に主に用いられる事に起因する。

表 8 は、ハッシュ系、ビット操作系の特殊命令を他の命令で置き換えた場合の必要サイクル数である。”Reg.” は、置き換えの際に必要となる、レジスタ数のオーバヘッドを示す。表 9 は、これらの特殊命令が存在した場合としなかった場合のサイクル数を比較したものである。この結果、ホームクラスタで 2.80%、リモートクラスタで 2.83% の性能向上となった。

表 7 特殊命令

Ins.	Compiled code	Home	Remote
HSHC	35	279	204
HSHG	0	0	0
HSHN	0	0	0
GUN	3	0	0
BITS	2	0	0
BITR	2	0	0
BITC8	12	4	0
CRDT	0	0	0
LUDR	0	0	0
SUDR	0	0	0

表 8　特殊命令の他命令による置き換え

Ins.	Cycles	Reg.	Ins.	Cycles	Reg.
HSHC	7	0	BITS	3	0
HSHG	4	0	BITR	4	0
HSHN	9	0	BITC8	34	2
GUN	5-84	2			

表 9　特殊命令の有無による必要サイクル数の比較

	With	Without	Impr.
Home	64471	66281	2.80%
CL1	43130	44354	2.83%

これら結果から、現在の DSM プログラムの実装では HSHC 以外の特殊命令の効果は限定的である事が分かった。

5.5　アーキテクチャの検討

MBP-light は DSM 管理プログラムの実装前に作成されたため、DSM 管理プログラムに最適な命令セットを備える事ができなかった。

MBP-light は、チップサイズの制限からオンチップに大規模なメモリを実装する事はできず、チップのピン数にも制限がある。ただし、事前に DSM プログラムの動作を把握できていた場合、

- パケットバッファとしても利用可能な、命令とデータを分離した特殊なキャッシュ
- パケットキューを取り扱う命令

を備えるアーキテクチャをとる事により、より高い性能を得られた可能性がある。

6. おわりに

本論文では，JUMP-1 の MBP-light の命令セットアーキテクチャについて、実機による評価を行った.

Buffer-Register Architecture を採用する事によって、ホームクラスタで 5.64%、リモートクラスタで 6.27%の性能向上を達成した。また、特殊命令のうちハッシュ値を求める命令は、有効で、2.80%の性能向上を達成したが、他の命令はあまり有効に機能していない。DSM 管理プログラムの主な処理はパケットのキュー管理であるため、ローカルメモリの load/store 命令の利用が頻発する。MBP-light では内部メモリやピン数の制限から命令/データメモリを分離していないが、これらを分離する事ができた場合、33%の性能向上を達成することができる。また、パケットキューを取り扱う命令を準備する事で、より性能向上できる可能性がある。

参 考 文 献

1) J. Laudon and D. Lenoski, “The SGI Origin: A cc-NUMA Highly Scalable Server,” The 24th ISCA, 1997.
2) T. Lovett and R. Clapp, “STiNG: A CC-NUMA Computer System for the Commercial Marketplace,” The 23rd ISCA, pp.308–317, 1996.
3) K. Hiraki et. al., “Overview of the jump-1, an mpp prototype for general-purpose parallel computations,” IEEE International Symposium on Parallel Architectures, Algorithms and Networks, pp.427–434, 1994.
4) Y.Yang et. al., “Recursive Diagonal Torus: An Interconnection Network for Massively Parallel Computers,” In IEEE Transactions on Parallel and Distributed Systems, Vol.12, No.7, pp 701–715, July 2001
5) 安生 健一朗 他, “超並列計算機 JUMP-1 における分散共有メモリ管理プロセッサ MBP-light”, 情報処理学会論文誌, Vol. 39,No. 6,pp.1632-1643, 1998 年 6 月.
6) Hieharu Tanaka et. al., “The Massively Parallel Processing System JUMP-1,” Ohmsha, ISBN4-274-90083-5, 1996.
7) M. Konishi et. al., “Implementation of Distributed Shared Memory Management of the JUMP-1 Multiprocessor,” IPSJ Transactions, Vol. 42, No. 4, pp. 674 － 682, 2001.
8) N. Suzuki et. al., “Performance Evaluation of a Multicast Mechanism of RDT Network for a Massively Parallel Processor JUMP-1” The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol J85-D-I, No.12, pp.1114-1125, Dec. 2002.
9) Yasunori Osana et. al., “Performance Evaluation of a Parallel I/O Mechanism on a Massively Parallel Processing System JUMP-1,” In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) 2001, June 2001.
10) J. Kuskin et. al., “The Stanford FLASH Multiprocessor,” The 21st ISCA, pp.302–313, 1994.
11) D. Chaiken and A. Agarwal, “Software-Extended Coherent Shared Memory: Performance and Cost,” The 21st ISCA, pp.314–324, 1994.
12) T. Kudoh et. al., “Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1,” International Conference on Parallel Processing, August, pp.I-186–I-193, 1995.