超並列計算機 JUMP-1 のクラスタの実装及び予備的性能評価

山本考伸[†]　津田健[†††]　秤谷雅史[††]
五島正裕[††]　森眞一郎[††]　富田眞治[††]

超並列計算機 JUMP-1 は、プロセッサ間の通信手段として共有メモリを提供している。通常の共有メモリ型並列計算機では、メモリ管理のコストが高く、細粒度の並列プログラムにおいて台数効果を上げることはできない事が多い。JUMP-1 においては、キャッシュコントローラと MBP-light を主体とする高機能メモリシステムが、細粒度並列計算時における通信レイテンシの隠蔽、削減を可能としている。

本稿においては、JUMP-1 全体のシステムの予備的な性能評価として、クラスタ内に閉じた並列計算の性能を評価した。結果、同一プロセッサを用いた市販の SMP に比べて、細粒度通信が必要な並列プログラム実行時に格段に高速実行可能であることがわかった。

The implementation and preliminary evaluation of the cluster of a massively parallel processor JUMP-1

Takanobu Yamamoto,[†] Takeshi Tsuda,[†††] Masashi Hakariya,[††]
Masahiro Goshima,[††] Shin-ichiro Mori[††] and Shinji Tomita[††]

A massively parallel processor JUMP-1 has shared memory. Ordinary shared memory systems don't fit for applications which frequently exchanges fine-grained messages. The inteligent memory system of JUMP-1 can hide or reduce fine-grained inter-processor communication latency.

We evaluated applications on a cluster. The result indicates that JUMP-1 can execute applications with fine-grained message effectively.

1. はじめに

JUMP-1 は、最大 1024 の要素プロセッサが接続される、分散共有メモリ型並列計算機である。JUMP-1 はクラスタ構造をとっており、4 つの要素プロセッサ毎に 1 つのクラスタを形成している。

JUMP-1 の特徴としては高機能メモリシステムがあげられ、共有メモリを用いたプロセッサ間通信のレイテンシを隠蔽、削減している。

現在 JUMP-1 のクラスタの実装が完了したため、クラスタ内で並列プログラムを実行した際の JUMP-1 の性能評価を行った。本稿では以下、2 章で JUMP-1 の設計方針、物理的な構成、及び今回の性能評価において用いた、高機能メモリシステムの特徴、機能等について述べる。その後 3 章で今回性能評価に用いた問題の詳細、及び各問題を実行した場合の評価結果を示し、クラスタ内の並列計算に関する考察を行う。

2. JUMP-1

本章では、超並列計算機 JUMP-1 のシステム全体の特徴について述べる。以下、設計方針、物理的構成、及び JUMP-1 が提供する高機能メモリシステムの概要の順に述べる。

2.1 JUMP-1 の設計方針

並列計算機においては、局所的な演算とプロセッサ間の通信を行う必要がある。並列計算機の要素プロセッサとして、市販の RISC プロセッサを使用した場合、各プロセッサの局所演算における最大性能は良くなるが、長時間の通信レイテンシを想定して設計されていないことが多い。そのため、通信完了を待つ必

[†] 京都大学大学院工学研究科情報工学専攻
Division of Information Science, Graduate School of Engineering, Kyoto Univ.
[††] 京都大学大学院情報学研究科通信情報システム専攻
Division of Communications and Computer Engineering, Graduate School of Informatics, Kyoto Univ.
[†††] 京都大学工学部情報学科
Department of Information Science, Faculty of Engineering, Kyoto Univ.

要がある場合には、システムの性能を低下させる大きな要因となる。要素プロセッサとして並列計算機専用プロセッサを開発した場合、プロセッサ間の通信遅延による計算性能低下を減少させることができる。しかし、要素プロセッサの局所演算性能を市販の RISC プロセッサ程度まで高めるための開発コストは、非常に大きくなってしまう。JUMP-1 においては、局所演算の最大性能を確保するために、要素プロセッサとして汎用 RISC プロセッサを用いている。

JUMP-1 では、要素プロセッサ間の通信手段として共有メモリを提供している。通常の共有メモリ型計算機においては、プロセッサ間の通信が頻繁に起こり、またその通信パターンが非定型的な場合、通信レイテンシが計算性能を著しく低下させる要因となってしまう。そこで、JUMP-1 においては、共有メモリ管理用の高機能メモリシステムを構築することによって、細粒度の通信を行った場合にも通信レイテンシを削減、隠蔽し、通信のオーバヘッドを下げようと考えている。

2.2 JUMP-1 の構成

要素プロセッサ間の接続形態を決定するにあたって、物理的な実装条件を自然なかたちで論理的な接続形態に反映できるように、JUMP-1 はクラスタ構成をとっている。要素プロセッサ 4 つ毎に 1 つのクラスタを形成している。各クラスタは 1 枚のプリント基盤上に実装されている。JUMP-1 において高速実行可能な並列プログラムを開発するにあたっては、クラスタ内とクラスタ間を意識する必要があるが、大規模問題の並列化においては、アルゴリズムを工夫することによって階層化された並列性を実現できる事が多いと思われる。

JUMP-1 においては、最大 256 のクラスタが、RDT ネットワークによって接続される。各クラスタ内の構成を図 1に示す。

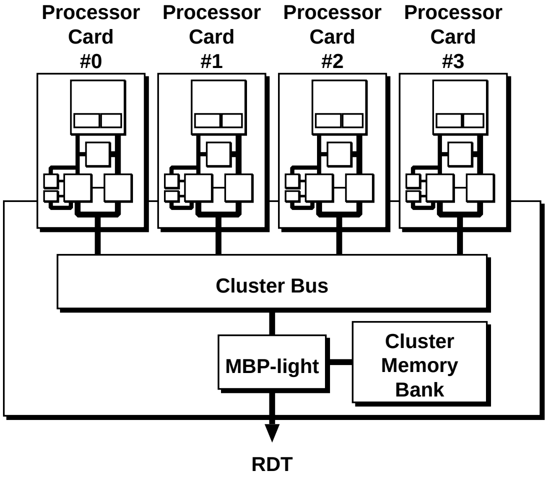

図 1　JUMP-1 のクラスタの構成

以下に、クラスタ内の各要素の詳細を述べる。

MBP-light1)　MBP-light は RDT ネットワークを介したクラスタ間のパケット送受信を管理するプロセッサで、クラスタ間にまたがる共有メモリ管理を担当している。JUMP-1 においては、クラスタ内とクラスタ間の 2 階層に階層化された構成となっているので、共有メモリ管理もクラスタ間とクラスタ内の 2 階層に分かれて実現されている。クラスタ内の共有メモリ管理は、後述のキャッシュコントローラ (CC) によって行われる。

Cluster Bus　Cluster Bus(CBus) は 4 つの要素プロセッサが共有しているメモリバスである。スプリットトランザクション方式を用いている。

Processor Card　クラスタ内の 4 つの要素プロセッサは、それぞれ 1 枚のプリント基盤上に実装されている。これを Processor Card と呼ぶ。図 2 に Processor Card 内の構成を示す。

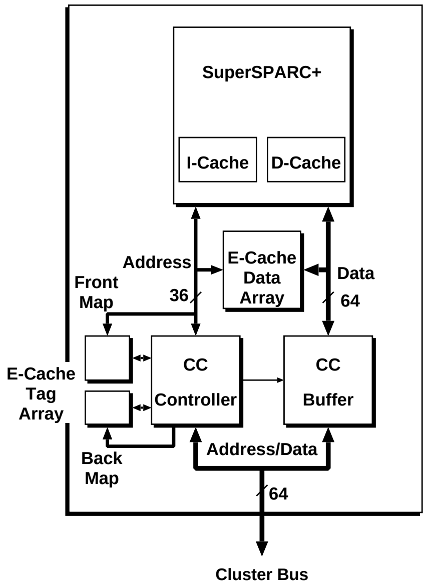

図 2　Processor Card の構成

以下に Processor Card 上の要素について詳細を述べる。

SuperSPARC+　JUMP-1 の要素プロセッサとなる汎用の RISC プロセッサであり、1 次キャッシュとして 16Kbyte のデータキャッシュ(D-cache)、20Kbyte の命令キャッシュ(I-cache) を内蔵している。通常の動作周波数は 50MHz であるが、今回の性能評価においては 25MHz で動作している。

CC　クラスタ内の共有メモリ管理を行う。MBP-light がプロセッサであるのに対し、クラスタ内の共有メモリ管理にかかる速度がシステムの性能を大きく左右すると思われるので、

CC は完全にハードウェアで構成されている。主に 1Mbyte の 2 次キャッシュ(E-cache) の管理を行っている。

2.3 高機能メモリシステム

JUMP-1 の高機能メモリシステムは、要素プロセッサ間の通信レイテンシを隠蔽、もしくは最小限に抑えるために様々な機能を有している。これらの機能は MBP-light 及び CC によって実現されている。本節では、JUMP-1 におけるアドレス体系と、高機能メモリシステムが提供する機能をユーザが使用するためのメモリコマンド、及びコヒーレンス制御方式の概要について述べる。

2.3.1 アドレス体系

各クラスタに分散配置されたクラスタメモリは、システム全体における主記憶としてだけではなく、他のクラスタメモリのコピーとしても利用される。すなわち、プロセッサから見ると 3 次キャッシュとして利用できる。

クラスタメモリを 3 次キャッシュとして利用する際のブロックサイズは、4K である。これは、SuperSPARC+が提供する最小のページサイズである。他のクラスタメモリのコピーであるページはコピーページと呼ばれ、元々自クラスタに主記憶として割り当てられたページはオリジナルページと呼ばれる。各クラスタのクラスタメモリ内には、コピーページとオリジナルページが混在することとなる。

要素プロセッサからのコピーページに対するアクセスは、オリジナルページと同様にクラスタ内における物理アドレスによって行われる。このアドレスをクラスタアドレスと呼ぶ。ユーザプロセス上の論理アドレスとクラスタアドレス間のマッピングは、通常の MMU が使用するページテーブルによって管理される。

一方、ページがクラスタ間で共有されている (オリジナルページが他のクラスタにコピーされている) 場合、各クラスタにおけるクラスタアドレスは異なるので、それらのページが同一ページであることを示すアドレスが必要となる。このアドレスをネットワークアドレスと呼び、各メモリオブジェクトを一意に決定する。簡単な方法として、ネットワークアドレスはユーザプロセス内の論理アドレスを、システム全体におけるプロセス ID で拡張したものを用いることができるが、ネットワークアドレスは完全にクラスタアドレスから仮想化されているため、MBP-light によって自由にアドレス付けすることが可能である。

2.3.2 メモリコマンド

高機能メモリシステムが提供する機能は、通常のシステムと同様のロード命令とストア命令を用いて利用することができる。高機能メモリシステムは、メモリアクセス命令の対象アドレスに応じた機能を提供する。例えは、キャッシュコヒーレンスプロトコルや、要素プロセッサ間の通信操作等を、メモリコマンドと呼ばれる、アドレスの一部によって動的に指定できる。

メモリコマンドはクラスタアドレスの一部として特定される。SuperSPARC+の物理アドレス (クラスタアドレス PA[35:0]) は 36 ビットであり、クラスタメモリの最大容量は 128Mbyte であるため、クラスタメモリ上の物理アドレスとしては PA[27:0] しか使用されない。PA[35:32] は、プロセッサが CC に対して、PA[27:0] で指定されるデータに対するアクセス方法を指定するための付加情報として使用され、メモリコマンドと呼ばれる。この 4 ビットによって、16 種類のアクセスパターンが提供される。PA[31:28] は各メモリオブジェクトの静的な属性を示す。

図 3にアドレス空間とメモリコマンドの関係を示す。ユーザプロセスが論理アドレス Vm で指定されるメモリオブジェクトに対して、異なるメモリコマンド n によってアクセスしたい場合、OS に対して

- 論理ページ Vm
- 新しいメモリコマンド n (n = 0,1, …,15)
- 新しい論理ページ Vn

を指示する。OS は Vm の物理ページ Pm の物理アドレス PA[27:0] を、新しいメモリコマンド n で拡張した物理アドレスを持つ物理ページ Pn を用意し、論理ページ Vn を物理ページ Pn にマッピングする。ユーザプロセスは同一メモリオブジェクトに対して異なるメモリコマンドを用いてアクセスすることができる。

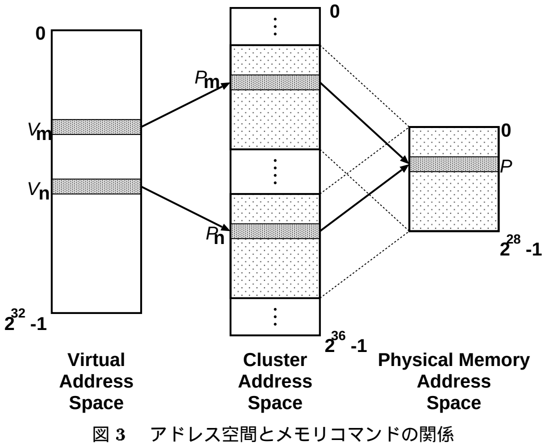

図 3　アドレス空間とメモリコマンドの関係

同一メモリオブジェクトに対して異なる物理アドレスを与えているため、SuperSPARC+ のデータキャッシュには、同一ブロックが異なるオブジェクトとしてキャッシングされることがある。SuperSPARC+は 4-way セットアソシアティブであるため、同一オブジェクトが最大 4 つまでキャッシングされる可能性があるが、CC は最大 2 つまでしかキャッシングされないように管理している。

2.4 キャッシュコヒーレンス制御

JUMP-1 のキャッシュシステムは、プログラミングが容易になるようにコヒーレントな共有メモリを提供するだけでなく、各メモリオブジェクトに対して適切なコヒーレンスプロトコルを指定することによる性能向上を目的としている。さらに、通信量と通信レイテンシを減少するために各種の機能が実装されている。以下では、キャッシュコヒーレンス制御の概要、及び今回の性能評価において用いた CC の機能について述べる。

2.4.1 キャッシュコヒーレンス制御の概要

本節では、キャッシュコヒーレンス制御の方式について述べる。

キャッシュコヒーレンスプロトコル CC は主に 2 種類のプロトコルを提供している。write-update プロトコルと write-invalidate プロトコルである。ユーザはメモリコマンドを通してコヒーレンスプロトコルを指定する。

2 次キャッシュの状態 CC は 2 次キャッシュの状態を管理している。2 次キャッシュの状態は、MOESI プロトコルに類似しているが、CC はクラスタ間の共有状態も管理している。

2.4.2 Update プロトコルの装備

超並列計算機において洗練されたプログラムを実行する場合、write-update プロトコルは write-invalidate プロトコルよりも優れていると言われている。write-update プロトコルのブロックの一部に対して store が起った時、CC は直ちに update パケットを送信せず、同一ブロックに対する store 要求を一定時間待って、update パケットを送信する。update パケットにおいては、1 ワード (32bit) 毎に更新情報が付加されている。ユーザは、update パケット送信までの待機時間、待機ブロック数等を指定できる。これに対し、write-invalidate プロトコルのブロックに対して store が起った時には、直ちに invalidate パケットが送信される。

3. クラスタ内の性能評価

本章では JUMP-1 の性能評価について述べる。現在のテスト環境においては、3CPU 以上の実行が保証されていないため、1CPU 及び 2CPU を用いて実行した結果を示す。コヒーレンスプロトコルは、write-update 及び write-invalidate の 2 種類それぞれをもちいて評価した。

性能評価の対象としては、同一の CPU(SuperSPARC+) を用いた SMP、SPARCserver1000 を用いている。2 次キャッシュの容量は JUMP-1 と同じ 1Mbyte である。SPARCserver1000 は OS 稼働状態で計測しているため、各性能評価において JUMP-1 よりも 1 割弱程度性能が低下している。

性能評価には行列の LU 分解を用いた。JUMP-1 の機能の特徴を見るために、3 種類の異なるアルゴリズムを用いて性能評価を行った。現在の JUMP-1 クラスタには主記憶が搭載されていない (使用できない) ため、問題サイズは 2 次キャッシュ(1M) 以内となっている。プログラムは C 言語で記述しており、SPARC Compiler ver 3.0.1 を用いている。コンパイル時に SPARC V8 対応の最適化オプションを選択している。各プログラムにおいて性能評価開始時には、行列の全要素を 2 次キャッシュに格納している。2CPU で実行する場合には、一方のプロセッサの 2 次キャッシュにのみ格納している。

本章では以下、性能評価に用いたアルゴリズムの詳細、及び各アルゴリズムに用いた時の性能評価結果を示す。

3.1 性能評価に用いたアルゴリズム

今回性能評価に用いた LU 分解アルゴリズムは以下のとおりである。

内積形式 LU 分解 1 ワード単位 (以下 IP1) 内積形式の LU 分解アルゴリズムにおいて、j(列番号)、i(行番号)、k(各要素を求める際のイタレーション) の順に外側からループのイタレーションに用いている。問題サイズを N とした時、ロードの回数が $\frac{2}{3}N^3$、ストアの回数が N^2、フローティング演算の回数が $\frac{2}{3}N^3$ となる。並列化においては、列方向 (イタレーション j) で 1 列ずつサイクリックに行列の要素を求めるプロセッサを分割している。プロセッサ間の同期は行列の 1 要素を計算終了する度にとっている。C 言語においては行列は行方向にマッピングされる。本アルゴリズムにおいては同一キャッシュブロック (8 ワード) に対して計算責任を持つプロセッサが分割されるため、コヒーレンス制御のための通信が頻繁に起こる。

内積形式 LU 分解 1 ブロック単位 (以下 IP8) 上記の内積形式 LU 分解において、要素を 8 ワード (1 ブロック size) ずつタイリングしている。ロード、ストア、フローティング演算の回数は、上記の IP1 と同様であるが、データの参照及び更新が同一ブロックに対して連続して行われるためキャッシュヒット率が向上する。並列化においては、列方向に 8 列毎にサイクリックに分割している。各ブロックに対する計算責任を持つプロセッサが固定されているため、コヒーレンス制御のための通信は各ブロックにつき 1 回しか行われない。プロセッサ間の同期は、8 ワード (1 ブロック) 毎にとっている。

外積形式 LU 分解 (以下 OP1) 外積形式 LU 分解において、k、i、j の順に外側からループのイタレーションに用いている。ロードの回数は $\frac{2}{3}N^3$、ストアの回数は $\frac{1}{3}N^3$、フローティング演算の回数は $\frac{2}{3}N^3$ となる。IP1、IP8 に比べてストアのオーダー

が増している。各ストアは行方向に連続して起こる。並列化においては、行方向 (イタレーション j) で 1 行ずつサイクリックに分割している。同一ブロックに対する計算責任を持つプロセッサは固定している。プロセッサ間の同期は k(最外側ループ) における各プロセッサの計算が終了する度にとっている。

3.2 性能評価結果

上記の 3 種類の LU 分解アルゴリズムを JUMP-1 及び SPARCserver1000 で実行した時の単位サイクルあたりのフローティング演算の回数を、図 4、図 5、図 6 に示す。先に述べたとおり、SPARCserver1000 は OS 稼働状態で性能評価を行っているので、1CPU 実行時における性能差は、OS のオーバヘッド等によるものと思われる。各アルゴリズム、問題サイズにおける性能向上率に注目して頂きたい。

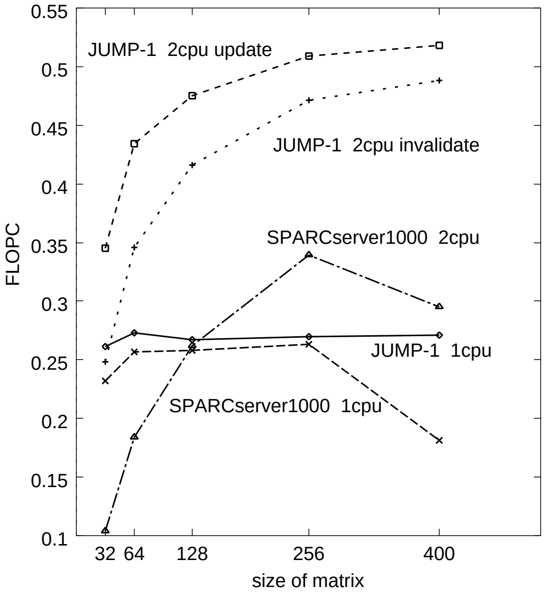

図 4 IP1

3.3 性能評価に関する考察

本章では、3 章で行った性能評価に関する考察を行う。問題サイズが 64×64 を越えたとき 1 次キャッシュのあふれが起こっている。問題サイズが 256 を越えてから、IP1、IP8 を SPARCserver1000 で実行した場合の性能低下の理由は、OS が 2 次キャッシュ領域を確保していることによる 2 次キャッシュのあふれ等が予想されるが、OP1 では低下していないことから、断定はできない。現在調査中である。SuperSPARC+における単位サイクルあたりのフローティング演算の理論最大値は$\frac{2}{3}$である。

各アルゴリズムにおいて、SPARCserver1000 に比べて JUMP-1 の性能向上率が優れていることがわかったが、特に問題サイズが小さい場合と、IP1 において JUMP-1 と SPARCserver の性能向上率に大きな隔たりが見られた。以下に、演算量と通信量の関係による性能向上率の差、及びコヒーレンスプロトコルによる性能向上率の差の 2 つの側面から今回の性能評価に関する考察を行う。

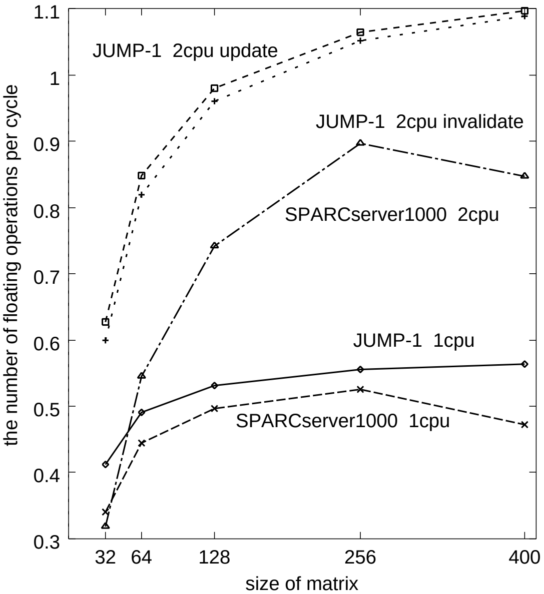

図 5 IP8

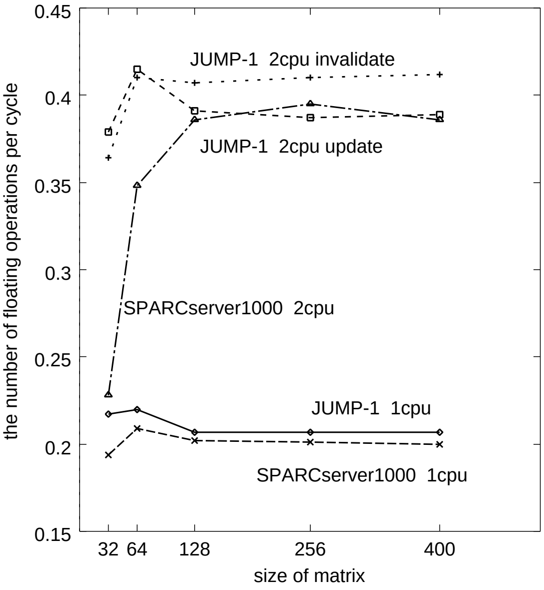

図 6 OP1

3.3.1 問題サイズと性能向上率

今回用いた 3 種類のアルゴリズム全てにおいて、SPARCserver1000 では問題サイズが小さい時には台数効果を得ることができない。これは、演算量に比べてプロセッサ間の通信量が多いために、通信レイテンシを隠蔽できない為だと思われる。JUMP-1 において

も、IP1をwrite-invalidateプロトコルを用いて行った場合、問題サイズが小さい時には通信レイテンシを隠蔽できず、台数効果が得られていない。問題サイズが大きくなるとSPARCserver1000、JUMP-1ともに十分な台数効果を得ることができている。

3.3.2 プロトコルと性能向上率

2で述べたメモリコマンドを用いてwrite-updateとwrite-invalidateを切り替えている。一般的に、適切に最適化されたプログラムにおいてはwrite-updateの方が高速であると思われる。IP1は、同一キャッシュブロック内で演算責任が分かれてしまっているので問題外ではあるが、タイリングという最適化を行ったIP8においてもwrite-updateの方が高速であり、JUMP-1にwrite-updateプロトコルが装備されていることは有効であることを示している。

一方、OP1においては、write-invalidateの方が高速である。このアルゴリズムにおいては、各データに対する演算の途中結果をメモリに保存しているためである。一般に、アルゴリズムを最適化することによってメモリに対する演算途中結果の書き込みを減少させることができると思われ、このような最適化を施し、参照されるべきデータのみ書き込む様にした場合、write-updateプロトコルの方が通信量を削減できる。計算機資源の制約上その様な最適化ができない場合には、複数回更新されるデータに対してはwrite-invalidateプロトコルを用いることが望ましい。

4. まとめ

JUMP-1のクラスタ内並列計算における性能評価を行った。結果、演算量に対して通信量が多いようなアルゴリズム及び問題サイズにおいては特に、同一プロセッサを用いているSPARCserver1000より台数効果を出せることが示された。この事は、JUMP-1においては細粒度の同期をとる共有メモリプログラムも高速実行可能であることを示しているものと思われる。

高度の最適化が施された並列プログラムにおいては共有データに対する無駄なコヒーレンス制御が不必要となり、キャッシュコヒーレンスプロトコルとしてwrite-updateプロトコルを用いた方が性能が向上するものと思われている。しかし、問題の性質及び計算機資源の環境によってはwrite-invalidateプロトコルの方が優れている場合があるため、各データに対するコヒーレンスプロトコルを切り替えることは有効であると思われる。

謝辞

日頃から貴重な御意見を頂いている文部省重点領域研究共同研究者各位に感謝します。

また、メンター・グラフィックス・ジャパン株式会社のHigher Education Programの一環として製品とサービスをご提供頂いたことに感謝します。

なお本研究の一部は、文部省科学研究費補助金（基盤研究(C)課題番号09680334、奨励研究(A)課題番号09780268）による。

参考文献

1) 佐藤充ほか: 超並列マシンJUMP-1のための分散共有メモリ管理プロセッサ, 並列処理シンポジウムJSPP'97予稿集, pp. 265–272 (1997).
2) 安生健一朗ほか: 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予稿集, pp. 321–328 (1997).
3) 松本尚, 平木敬ほか: 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予稿集, pp. 245–252 (1993).
4) 平木敬ほか: JUMP-1共有バス仕様書 (1995).
5) 五島正裕ほか: JUMP-1 CBus仕様書 (1994).
6) 五島正裕ほか: The Intelligent Cache Controller of a Massively Parallel Processor JUMP-1 (1997).
7) 津田孝夫: 数値処理プログラミング, 岩波書店 (1988).