計算機アーキテクチャ　148－7
（2002.　5.　13）

１つのシーケンサに協調動作するクラスタ構成VLIW並列計算機のシミュレーションによる性能評価

藤岡 豊太† , 村上 武† , 永田 仁史† , 安倍 正人†

岩手大学工学部 情報システム工学科 †
〒020-8551 盛岡市上田 4-3-5 岩手大学工学部 †
Tel: (019)621-6972, e-mail: toy@cis.iwate-u.ac.jp

あらまし　従来の MIMD 並列計算機で用いられているメッセージパッシング方式では、メッセージ生成に要する時間が演算のオーバーヘッドとなり、特にプロセッサ間通信を頻繁に要求するような演算の場合マルチプロセッサによる恩恵を十分に得ることが困難となる。本報告では、並列計算機において全てのプロセッサを一つのシーケンサに協調して動作させることにより、メッセージパッシング方式で発生するデータ通信時のオーバヘッド時間を軽減するクラスタ構成 VLIW 並列計算機を提案する。また、本方式の有効性を確認するため、シミュレータを用いて提案方式および同様の構成での MIMD 方式との間での比較検討を行う。

Performance evaluation by simulation of the cluster composition type parallel VLIW computer

Toyota Fujioka† , Takeshi Murakami† , Yoshifumi Nagata† , Masato Abe†

Faculty of Engineering, Iwate Univ.†
(Faculty of Engineering, Iwate Univ. 020-8551 Japan)†

Abstract　In conventional MIMD parallel computer using message passing method, it becomes difficult to fully obtain benefit of multiprocessor, because message generation cause overhead time of data communication. In this report, we propose the clusterized VLIW parallel computer which mitigates overhead time at the data communications by operating all processors in cooperation to one sequencer in a parallel computer. Moreover, to evaluate the validity of this system, we performed comparison examination between a proposal system and MIMD parallel computer in the same composition using a simulator.

1. はじめに

情報処理全般において、処理内容の高度化・多様化に伴い、１つの処理を集中して行うのではなく、作業工程を分割して同時に実行する分散処理が行われるようになってきている。計算機における処理についても同様で、プロセッサ自身の高速化の一方で、多数のプロセッサを用いた並列処理により演算処理の高速化を実現する並列計算機が、コスト・柔軟性・信頼性の面から広く利用されてきている。

本研究では、多数のプロセッサを１つのシーケンサに協調して動作させることにより、メッセージ交換方式のようなメッセージ生成時のオーバーヘッドのない多数のプロセッサによる分散メモリ型クラスタ構成 VLIW 並列計算機について動作シミュレーションを行い、一般的なメッセージ通信型の MIMD 並列計算機との比較による性能評価を行った。

2. クラスタ構成 VLIW 並列計算機

多数のプロセッサによる並列計算機の場合、プロセッサ間の同期をとる方式として (1) 共有部分に置かれた共有変数を参照する方式、(2) メッセージ交換による方式、の２つ方式が一般的である。現在の並列計算機の多くはメッセージ交換によるデータ通信を行っているが、データ通信が頻繁に行われる場合、データ通信に際してのメッセージ生成などに要する時間が処理全体のオーバーヘッドとなり、並列性の効果を十分に得るための障害となる。

そこで我々は、メッセージ生成のオーバーヘッドを回避する手法として、メモリアクセスも含めて全てのプロセッサを１つのシーケンサ（プログラムカウンタ、以下カウンタ) に協調して動作させる分散メモリ型ＶＬＩW並列計算機を提案している [1]。これは、一見共有変数を参照する方式に近いが、参照する部分がカウンタのみであり全プロセッサ上のプログラムはカウンタを基本にして動作するので、既存の共有変数方式のようなプロセッサ間での排他的制御の必要はない。

提案方式では、全プロセッサに対しカウンタが１つであるため、計算機上で動作するアプリケーションは、カウンタの各状態において全プロセッサの挙動 (動作する命令) を考慮してコンパイルする必要がある。しかし、例えばプロセッサ間のデータ通信などの場合、ロード／ストア命令 (もしくはそれに類する命令) 実行時に、各プロセッサで既に通信データに対して必要な情報 (送受信の宛先など) は確定しているため、メッセージ交換方式のようなメッセージ生成の必要はなく、データ通信におけるオーバーヘッドの問題を回避できる。

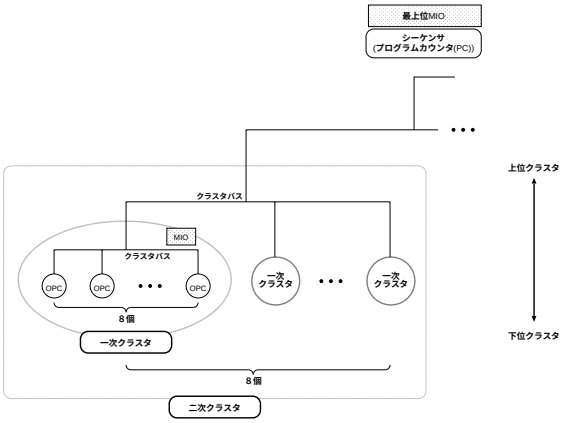

図 1．クラスタ構成

3. VLIW 並列計算機の内部構成

3.1 プロセッサ接続形態

提案するクラスタ構成ＶＬＩW並列計算機は、32[bit] を 1 ワードとしたワード計算機であり、各プロセッサ (Operation Unit with Cache、以下 OPC) を、プロセッサ間のデータ通信に効率性と実装の容易性の双方を考慮して図１のようなクラスタ状に構成する。

カウンタの制御回路は OPC 外部、具体的には最上位クラスタに１つ必要となる。同様に、全 OPC のキャッシュメモリのコヒーレンス維持のためキャッシュミス時の他 OPC の該当ラインの無効化処理 (invalidate 処理) などのために、外部にメモリＩ／O管理ユニット (Memory I/O Control Unit、以下以下 MIO) を設けている。MIO は、各一次クラスタに１つずつと最上位クラスタに１つ配置される。以後、各クラスタ内 MIO を「クラスタ MIO」、最上位クラスタの MIO を「最上位 MIO」と呼び、MIO 全体

を差す場合には単に MIO と呼ぶこととする。何らかの理由で MIO に処理が以降される際には、全 OPC の動作は interrupt され、MIO での処理の終了後、再び OPC での処理が再開される。

MIMD 並列計算機の場合、各プロセッサが独自にカウンタを持っているため、データ通信などで個々のプロセッサ間で同期させる場合は、例えばバリア同期のようにプログラム内で必要な位置に同期のための命令を入れるなどしておけば、同期命令によって他プロセッサとの間で協調のための制御信号の通信などによるオーバヘッドが発生するものの、各プロセッサ上のプログラムのは別個に最適化でき、またプロセッサごとの条件分岐に際しても他プロセッサの処理について考える必要はない。提案アーキテクチャでは、同期のための特別な命令が必要ない代わりに、VLIW 命令の並べ替え同様にコンパイラ側で全ての OPC のプロセス毎の動作を考慮しながらプログラムを構築する必要がある。また条件分岐の場合も、全 OPC のプログラムが一斉に同じ PC 値のアドレスへ分岐していくことになるので、それも考慮してプログラムを構築する必要がある。

3.2 OPC 内部構成

OPC は、複数の命令をパイプライン処理する VLIW 型プロセッサである (図 2)。この他にロード／ストアユニット (DMU)、制御ユニット、即値生成器などが含まれる。

3.3 メモリ・キャッシュ構成

データメモリ・命令メモリは、分散もしくは共有の形で計算機全体に各一つ実装する。

各 OPC には、命令メモリ用一次キャッシュ、データメモリ用一次、二次キャッシュを配置し、双方ともライトバック方式･ダイレクトマップ型のキャッシュである。命令キャッシュには、機械語に翻訳された自 OPC に該当する命令が格納される。データ用キャッシュは、一次キャッシュ、二次キャッシュ双方ともキャッシュミス発生時のコヒーレンス維持手法に write-invalidate 方式を採用する。キャッシュのコヒーレンス維持

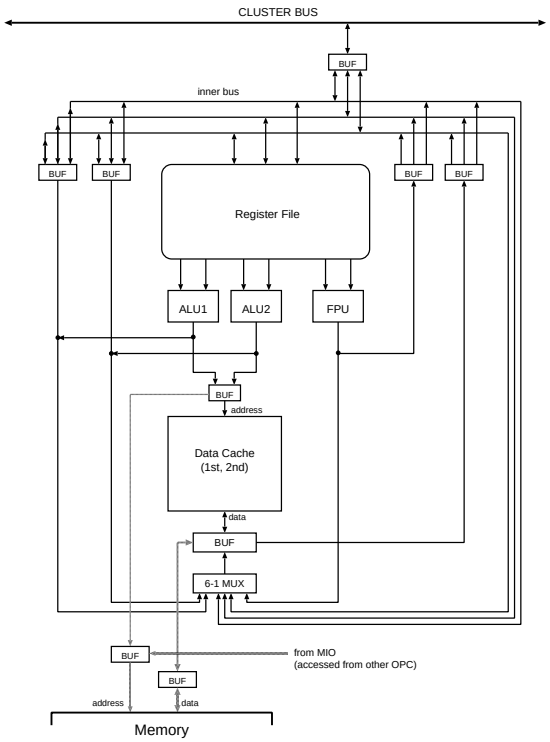

図 2．OPC の内部構成

に関する制御は MIO よって行われるが、クラスタ MIO では各クラスタ内でのキャッシュ制御に関連する具体的な処理を、最上位 MIO ではコヒーレンス維持などの際必要な他クラスタおよび他 OPC との間の必要情報の制御を主に行う。同時に複数の OPC でキャッシュミスが発生した際には、OPC に優先順位を付けて排他的に順次処理していくことになるが、これらの順位付けなどは最上位 MIO が行う。

4. クラスタ構成 VLIW 並列計算機のシミュレーション評価

クラスタ構成 VLIW 並列計算機の性能を評価するため、提案する VLIW 並列計算機および同構成･条件による MIMD 並列計算機双方を模擬したシミュレータを作成した。そのシミュレータ上で、いくつかの並列処理向きと考えられるプログラムを動作させ、性能評価を行った。

本報では、図 3、図 4 の構成で各提案する VLIW 並列計算機、メッセージパッシング方式の MIMD 並列計算機の計 4 種のワード計算機についてのシミュレーション評価を行った。

本シミュレーションでは、プロセッサ間のデー

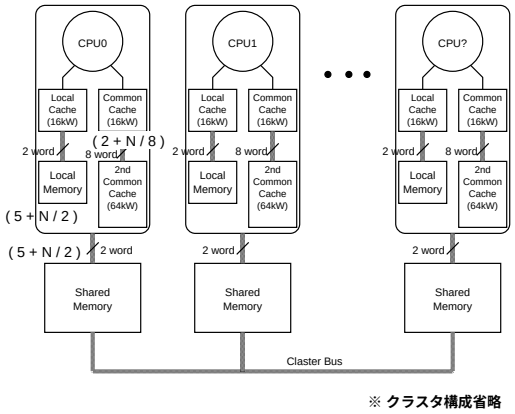

図 3．分散メモリ型クラスタ構成

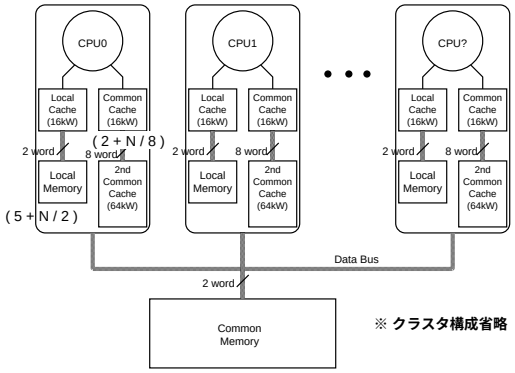

図 4．共有メモリ型

タ通信面の改善が与える効果について検証を目的としている。そのため命令メモリに関しては、命令は全て各 OPC の命令キャッシュに格納されている状態を想定した、以後外部の命令メモリとのアクセスはないものと仮定している。

想定したプロセッサコア部およびキャッシュ、クラスタバスの動作クロックを表 1 に示す。この値を元に各動作に要するクロック数を算出している。VLIW 並列計算機で想定したクロック値は、現在のプロセッサ動作クロックからすると低速なクロック数であるが、外部シーケンサと協調して動作させる点や、バスや外部メモリ

表 1．シミュレーション時の想定クロック値

(単位：MHz)	MIMD	VLIW
プロセッサコア	500	100
キャッシュメモリ	250	
クラスタバス	100	

(DRAM) の動作クロック数の観点から、提案する VLIW 並列計算機では妥当な動作クロック数と考えられる。

4.1 シミュレータ構成

4.1.1 OPC

OPC の基本構成は図 2 に示すように、3 本の内部バス A、B、C を持ち、32[bit] x 64 個の汎用レジスタと、プレディケート付き状態バッファリングによる投機的実行 [2] が可能なように 1[bit]× 4 個の条件レジスタを持つレジスタファイル、およびバッファや MUX を経由してキャッシュや ALU、FPU と接続される。演算ユニットとして 2 個の ALU、1 個の FPU がレジスタファイルと結線され、2 個の ALU は各々DMU とも直結している。そのため、各々の ALU がアドレス生成器の役割も果す。その他に一次、二次キャッシュと即値生成器、制御ユニットがある。一次キャッシュ、二次キャッシュは各々 16k ワード、64k ワード、一次キャッシュ、二次キャッシュ間のバス幅を 8 ワード、二次キャッシュ、メモリ間のバス幅を 2 ワード、ラインサイズは 128 ワードとした。

1OPC あたりの長命令語長は 128[bit] と設定した。そのため 3 つの命令 (スロット) で構成され、3 命令までを並列実行可能としている。また、各スロットの命令は (i) 命令フェッチおよびデコード、(ii) レジスタリード、(iii) 演算またはバス出力、(iv) レジスタライトの 4 段のステージによるパイプライン動作を行う。

4.1.2 OPC およびメモリ配置

分散メモリ型は、図 1 のように 8 個の OPC から成る一次クラスタを基本クラスタとしたクラスタ構成である。また、各 OPC に同容量で他 OPC からも参照可能なメモリが図 3 のように配置されている形態である。共有メモリ型は、図 4 のように、各 OPC 間の結合はクラスタ構造ではなく単一バスで結線され、単一のメモリを確保している形態である。

VLIW 並列計算機と MIMD 並列計算機双方とも、キャッシュミス時には外部メモリおよび他 OPC のキャッシュとのコヒーレンスを保つ

ため write-invalidate 処理を必要とする。本シミュレーションでは、invalidate 処理に要する時間として 100[ns] を設定した。

本シミュレーションでは、クラスタ構成分散メモリ型ではバス結合のデータ遅延を想定して、自 OPC のメモリへのデータ通信のみに要する時間を 1 として、自クラスタの他 OPC メモリとの間のデータ転送はその 4 倍、他クラスタのメモリに対しては 8 倍というマージンを設けることにした。また共有メモリについては、メモリが外部にあるということから、どの OPC に対しても分散メモリ型での他クラスタ他 OPC の場合と同一である。メッセージ通信時の初期時間は、富士通の高並列計算機 AP1000+を参考にして 5.0[μs] とした。[3]

4.2 シミュレーション用プログラム

各シミュレータでは、以上の設定値を元に、専用アセンブラ言語によりプログラムされた後に専用機械語にエンコードされたアプリケーションにより、実行させた際の所要クロック値と各キャッシュのヒット、ミス数を出力する。

4.2.1 高速フーリエ変換 (FFT)

データ長 N、OPC 数を P とした場合、0 ～ N/P-1 番目のデータを 0 番目の OPC で、N/P ～ 2*N/P-1 番目のデータを 1 番目の OPC で、… という配置でで並列処理させている。

4.2.2 基数ソート

基数ソートは、ソートされるデータを基数のビット幅ごとに分割し (これをビットグループと呼ぶ)、下位のビットグループから順に基数の各値ごとの数を数え上げ、その値を元にデータの移動先を決定しソートしていく。この数え上げ演算は複数プロセッサにより並列に行うことができ、並列処理向きのソートアルゴリズムとされている [4]。

4.3 シミュレーション結果

4.3.1 FFT

図 5、図 6 に、データ長 128k ワードの場合の FFT のうう時間と OPC 数に対する性能向上率のシミュレーション結果を示す。

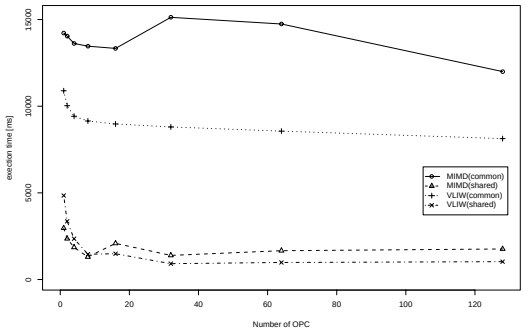

図 5．実行時間

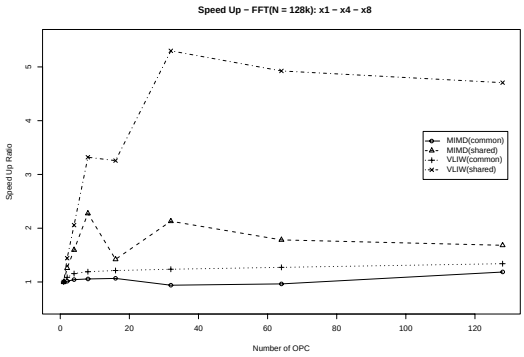

図 6．OPC 数に対する性能向上率

図 5 から、VLIW 並列計算機、MIMD 並列計算機とも分散メモリ型に比べ共有メモリ型のほうが大幅に実行時間を要している。

また分散メモリ型について見ると、OPC 数が 8 個、つまり一次クラスタに収まる個数までについては MIMD 並列計算機のほうが VLIW 並列計算機に比べ良い性能を示すが、16 個以上になると逆に VLIW 並列計算機のほうが良い性能となり、MIMD の場合の最適性能に比べても良い性能を示している。

4.3.2 基数ソート

図 7、図 8 に、データ長 1M ワード、基数 16 の場合の基数ソートの実行時間と性能向上率のシミュレーション結果を示す。

図 7 から、FFT の場合と同様に分散メモリ型に比べ共有メモリ型のほうが大幅に実行時間を要している。分散メモリ型についても、OPC 数が少ない (一次クラスタ内で完結する) 場合の挙動に違いはあるが、VLIW 並列計算機の方が MIMD 並列計算機に比べ良い性能を出して

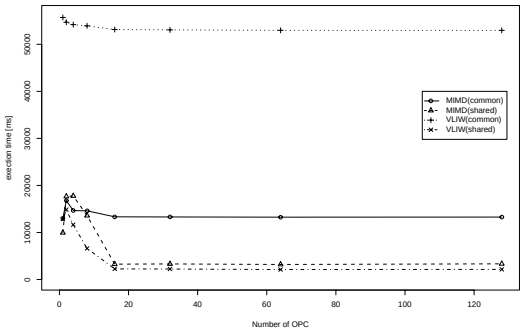

図 7．実行時間 (基数 = 16)

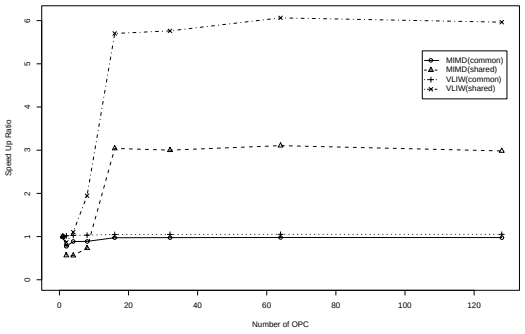

図 8．OPC 数に対する性能向上率 (基数 = 16)

いることがわかる。

4.3.3 各計算機性能の比較・検討

FFT、基数ソートでのシミュレーション結果より、提案する分散メモリ型クラスタ構成 VLIW 並列計算機で演算性能、OPC 数に対する性能向上率ともに最も良い性能が得られた。

分散メモリ型クラスタ構成、共有メモリ型双方とも、演算速度、性能向上率双方とも、並列度が向上すると VLIW 並列計算機のほうが MIMD 並列計算機に比べ良い性能を示した。これは、演算速度自体は MIMD 並列計算機のほうが大幅に高速であるため、OPC 数が少ない場合はデータ通信量に比べプロセッサの演算に要する割合が大きいが、並列度を増加させるにつれ OPC 間でのデータ通信量が増大するため、MIMD でのメッセージ生成の初期時間の影響が増大してくるためであるものと考えれられる。

メモリ配置の違いから見ると、VLIW 並列計算機、MIMD 並列計算機双方について、共有メモリ型では、分散メモリ型に比べ演算速度、性能向上率ともに大幅に下回る結果を示した。一つには、各 OPC の参照する外部メモリが共通していることから一連の処理データを連続したアドレスに配置したためであるものと考えられる。この場合、データのメモリ配置やアドレス計算に関しての労力は少なくなるが、並列度に対して各 OPC のキャッシュのヒット率は特に向上しないため、全体としてのキャッシュミス数がほとんど減少しないためであるものと考えられる。

5. まとめ

本報では、提案するクラスタ構成分散メモリ型 VLIW 並列計算機のアーキテクチャについて、シミュレーションにより性能評価を行った。その結果、FFT や基数ソートのようなメモリアクセスの頻繁なプログラムに対し、提案する VLIW 並列計算機でにより、メッセージパッシング方式での MIMD 並列計算機に比べ良い性能が得られることを確認した。

参考文献

[1] 安倍正人, 松沢伸一, 岡部公起, 根本義章, ”分散演算器・データキャッシュを持つクラスタ構成 VLIW 計算機”, 情報処理学会 計算機アーキテクチャ研究会 技術研究報告 108-7, pp.41-47 (1994)

[2] 安藤秀樹, 中西千嘉子, 原哲也, 中屋雅夫, ”プレディケート付き状態バッファリングによる投機的実行”, 並列処理シンポジウム JSPP’95, pp.307-314

[3] 白川長武, 小柳洋一, 今村信貴, 林憲一, 清水俊幸, 堀江健志, 石畑宏明, ”高並列計算機 AP1000+のメッセージハンドリング機構”, 並列処理シンポジウム JSPP ’95, pp.233-240

[4] 児玉祐悦, 坂根広史, 佐藤三久, 山名早人, 坂井修一, 山口喜教, ”高並列計算機 EM-X による radix ソートの実行”, 並列処理シンポジウム JSPP’96, pp.307-314