計算機アーキテクチャ　113－5
（１９９５．　８．　２３）

Unlimited Speculative Executionの制御オーバヘッド削減手法

山名早人　佐藤三久　児玉祐悦　坂根広史　坂井修一†　山口喜教
電子技術総合研究所　　†新情報処理開発機構

本報告では，タスクレベルの投機的実行を並列計算機上で実現する際に生じる制御オーバヘッド削減について検討する．並列計算機EM-4上に分散制御方式をソフトウェアにより実現し，制御オーバヘッドの発生原因毎に，投機的実行の効果に与える影響を調べた．その結果，放送のレイテンシやタスクの起動処理にかかるオーバヘッドの影響は小さく，制御情報を処理する放送受信処理が最も性能に影響を与えることがわかった．放送受信処理が1/4に削減された場合，現在のインプリメントの約3倍の性能向上が期待できる．この時，理論速度向上が13.6倍のプログラムを32台のPEで実行すると，最大10倍の速度向上が得られる．

Decreasing the Control Overhead of the Unlimited Speculative Execution

Hayato YAMANA　Mitsuhisa SATO　Yuetsu KODAMA　Hirohumi SAKANE
Shuichi SAKAI†　Yoshinori YAMAGUCHI
yamana@etl.go.jp　<URL: http://www.etl.go.jp/People/yamana/>
Electrotechnical Laboratory　　† Real World Computing Partnership
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

This paper discusses how to decrease the control overhead of tasks with speculation on multiprocessors. Firstly, we have implemented the unlimited speculative execution on the EM-4 multiprocessor. Secondly, the overhead is classified into its several sources. After measuring each classified overhead, it has been confirmed that both the broadcast latency and the overhead initiating tasks are not major factors. Insted, the overhead of receiving and manipulating broadcasted control data is major factor. When the factor is decreased by 1/4, the speedup ratio increases up to 3 and we will have 10 times speedup on 32 PEs with the program whose theoretical speedup ratio is 13.6.

1. まえがき

条件分岐の評価結果を待たずに，演算を開始することを投機的実行(**Speculative Execution**)と呼ぶ．本報告では，我々が提案している**マクロタスクレベルUnlimited Speculative Execution**[1]で用いるマクロタスク分散制御方式[1][2]の制御オーバヘッド削減のための検討を行い，制御部分をハードウェア化した場合の性能を予測する．

従来の投機的実行は，スーパスカラプロセッサやVLIW計算機を対象としてプロセッサ内部で行われてきた[3]．これに対し，マクロタスクレベルUnlimited Speculative Executionは，プログラムをマクロタスク（以下，**MT**）に分割し，MT間の投機的実行を並列処理計算機上で実現する方式である．MT間の投機的実行により，プログラム全体に渡る投機的実行が可能となり，投機的実行の理想モデルであるOracle Model[4]適用時の理想的な速度向上率に近づくことができる[3]．Oracle Modelとは，条件分岐の結果が実行前に全て既知であり，計算機資源が無限であるという実行モデルである．Oracle Modelを仮定すると，投機的実行を行わない場合に比較して12～630倍の速度向上が得られる[3][5][6]．

このようなタスクレベルで投機的実行を行うことにより，従来扱うことの出来なかったループ間・イテレーション間・タスク間の投機的実行が可能となる．例えば，ループ内の条件分岐で，一方の分岐が選択されるとloop carry dependenceが存在するが，もう片方の分岐が選択された場合には，loop carry dependenceが存在しないといったループを考える．このようなループは頻繁に現れる[7]．タスクレベルの投機的実行では，loop carry dependenceのない側の各イテレーションを複数同時に実行開始でき，全てのイテレーションでloop carry dependenceがない側が最終的に選択されたとすると，最大，イテレーション回数倍の速度向上が得られる．このような複数のイテレーション間に渡った投機的実行は，従来のSuperscalarやVLIWでは得られず，タスクレベルの投機的実行が得意とする分野である．

我々の提案しているマクロタスクレベルUnlimited Speculative Execution[1]は，投機的実行に適したMT生成手法[1]及び，MTの制御オーバヘッドを隠蔽する分散制御方式[1][2]の2つから構成される．これらの手法を用いることにより，並列計算機が持つ通信性能とマクロタスクサイズから，実行時に動的に投機的実行の段数(Speculation Depth)が決定される．これまでの評価結果から，ソフトウェアにより並列計算機EM-4[8]上に分散制御方式をインプリメントした場合，平均マクロタスク実行時間が14.4μs以内であれば，投機的実行の効果を得られることが分かっている[1]．しかし，14.4μsはEM-4のアセンブラ命令で180命令に相当するため，さらなる制御オーバヘッドの削減が必要とされる．そこで，本稿では，オーバヘッドの発生原因を各プリミティブ別に調査し，各プリミティブにおけるオーバヘッド削減の効果を検討する．以下では，**2.**でマクロタスク制御手法の概要を述べ，**3.**で制御オーバヘッドを各プリミティブに分類しパラメータ化する．そして，**4.**で並列計算機EM-4を用いて，パラメータを検証すると共に，制御部分のハードウェア化について検討する．

2. 分散制御方式

本節では，マクロタスクレベルUnlimited Speculative Executionとマクロタスクレベル Unlimited Speculative Executionで用いられる分散制御方式の概要を述べる．

2.1 マクロタスクレベルUnlimited Speculative Execution[1]

マクロタスクレベルUnlimited Speculative Executionは，データ依存が解決されたMTから実行を開始させ，制御フローが最終的に到達しなかったMTの実行を後で停止させる実行方式である．

具体的には，制御依存とデータ依存の競争を行わせ(**competitive execution**と呼ぶ)，データ依存の解決と制御依存の解決において，データ依存の解決の方が早い場合に投機的実行を行う(図1)．

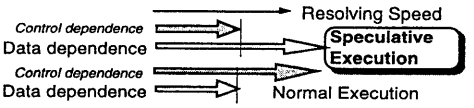


図1 Competitive Execution

これにより，対象とする並列処理計算機に応じた投機的実行が行われる．すなわち，通信コストやタスク起動コストが小さいほど，制御依存が解決する前に，より多くのMTを起動できることになり，広範囲に渡る投機的実行が可能となる．しかし，プロセッサ数が少ない場合には，資源の枯渇を引き起こす可能性がある．そこで，最大先行評価段数Nを設定し，Nよりも先の投機的実行を制限する．

competitive executionに基づいた投機的実行方法を「MTの実行開始を明示することなく，条件ジャンプ段数を制限することなく投機的実行が行われる」という意味から**unlimited speculative execution**と呼ぶ．図2に概要を示す．

2.2 分散制御方式[2]

分散制御方式の概要を図3に示す．各MTは，（1）自MTの後続MT，すなわち，自MTからデータ依存を持つMTを動的に生成すると共に，（2）システム全体に放送される制御情報を随時監視し，自MTの次の状態（制御確定あるいは実行停止）を自分自身で判断する．これにより，各MTは自MTの制御のみを行えばよく，制御オーバヘッドがMT数に依存しない．また，投機的実行中のMT内で得られた分岐方向決定情報も制御情報として用いることにより，MT制御の処理が制御フロー順に直列化されることを回避する．

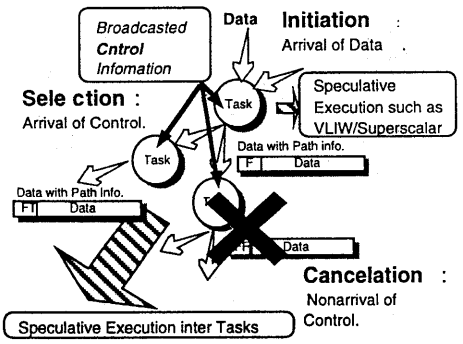
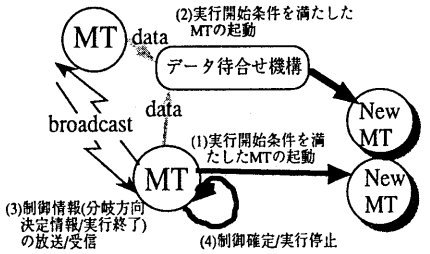


図2 Unlimited Speculative Execution



(1) 起動MT(ディスティネーションMT)が単一MT(ソースMT)からのみデータ依存を受けている場合，ソースMTにおいてディスティネーションMTを生成.
(2) 起動MTが複数MTからデータ依存を受けている場合，データ待合せ機構においてディスティネーションMTが必要とするデータの待合せを行い，データが揃った時点でMTを起動.
(3) 各MTは，自MT内での条件判断結果を全PEに放送.
(4) 各MTは，放送された制御情報に基づいて自MTを制御.

図3 分散制御方式の概要

MTの起動

MTの起動条件は，MTが必要とする全てのデータの定義の終了保証であり，次の3通りの方法がある(図4).

（1）データ依存を持たないMT 他のどのMTからもデータ依存を持たない．このため，いつでも起動できる．先行評価段数を制限するため，最大先行評価段数Nを定義し，次起動の先行評価段数がN以内であれば起動させる．また，起動は，ダミーのMTを1つ作成し，ダミーMTから起動させる．

（2）1つのMTからのみデータ依存を持つMT 先行する1つのMTからのみデータ依存を持つ．このため，ソースとなるMTから起動できる．この場合，ソースとなるMT内にMT起動コードを埋めこむ．

（3）複数のMTからデータ依存を持つMT 先行する複数のMTからデータ依存を持つ．このため，複数のMT内のBTの終了を待つ必要がある．データ定義終了を判定するために，データ駆動を用いる．マッチングのためのタグとして，MT及びBT番号を用いる．

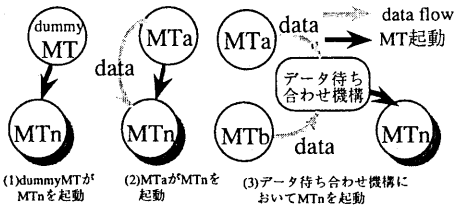


図4 MTの起動方法

MTの制御

MT制御のために，**レベル番号L，経路情報P，最大先行評価段数N**の3パラメータを用いる．実行中の各MTは，レベル番号Lと経路情報Pを持ち，放送される制御情報（分岐方向決定情報を含む）と，自MTの持つ経路情報Pを比較することにより，自MTに対する次の制御（制御確定・実行停止）を決定する．

レベル番号Lは，放送される制御情報と経路情報の比較時に用いる値であり，プログラム実行開始後，現在までに制御確定した条件分岐数を示す．プログラム実行開始時の初期値は0とし，条件分岐が実行される毎にインクリメントする．

経路情報Pは，制御が確定済みのMT（ルートMT）から投機的実行中である自分のMT（自MT）までの間に存在する分岐方向をT（true），F（false），＊（don't care）の記号を用いて羅列したものであり，例えばP＝T＊Tと表す．経路情報のT及びFは，付加情報としてフラグsを持ち，既に確定済みの分岐の場合にはフラグsを付加する（例：P＝T＊Ts）．そして，MTが持つ経路情報に全てフラグsが付くとMTの制御が確定したことを示す．ルートMTの経路情報は空である．また，ルートMTから自MTまでの経路が複数あり，かつ，don't careで表せない場合には，制御情報Pを複数持つものとする．

最大先行評価段数Nは，最大の先行評価段数を表し，N段を越える投機的実行を行わない．

図5に例を示す．図5では，MTiがルートMTである状態（制御確定状態）を示す．また，MTi内部の条件分岐は，実行を開始してからL番目の条件分岐であるとする．この時，MTkは，投機的実行中であり，経路情報としてFTを，レベル番号としてLを持つ．すなわち，経路情報は，ルートMTからの経路を示し，レベル番号は，経路情報の先頭の条件分岐のレベルを示す．ここで，MTnがMTkからのみデータ依存を持つとすると，MTkにおいてMTnの起動を行うことができ，MTnの経路情報は，（MTkの経路情報）＋（MTk～MTn間の経路情報）となる．またレベル番号はルートMTが変わらない限り同一となる．

次に，図6に各MT内での処理を示す．各MT内で条件分岐の結果が得られると，図6に示すように，制御情報として，（MTのレベル番号, MTの経路情報＋分岐方向）を放送する．例えば，ルートMTでF

（False）が得られた場合には，（L，F）を放送する．これは，Fがレベルしであることを示す．一方，投機的実行中のMTkにおいて，F（False）が得られた場合には，（L，FTF）を放送する．この時，FTFの先頭（左端）のFがレベルLであり，FTという仮定の元に分岐方向Fが得られたことを示す．制御情報を受け取ったMTnは，制御情報中の経路情報とMTnの経路情報が等しい場合，制御情報中の分岐方向とMTnの経路情報を比較する．この例では，一致しているのでFをFsに変更し，条件が成立済み（セレクト済み）であるフラグsを付ける．一方，分岐方向が一致しなかった場合は，MTnは自MTの実行を停止させる．

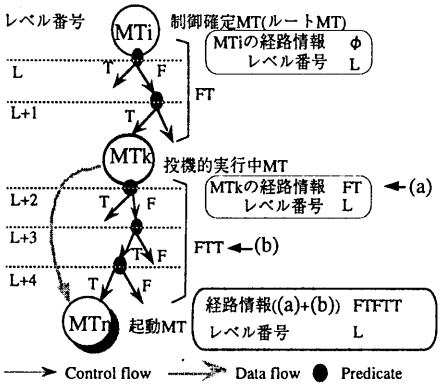
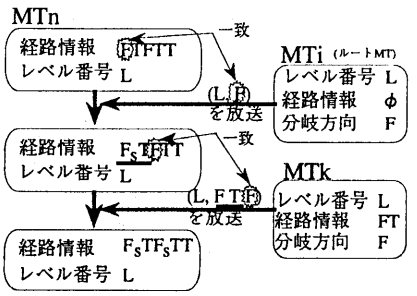


図5 MT制御概要



制御情報=(レベル番号,自MT経路情報+分岐方向)

図6 MT制御情報処理

3．制御オーバヘッドの分類

本節では，MTの分散制御に伴うオーバヘッドを発生原因別に分類し，各々のオーバヘッドの基本値を示す．基本値とは，他の負荷が存在しない場合の理想的な値である．

3.1 制御オーバヘッドの分類

MTの分散制御に伴うオーバヘッドは，**図7**に示すように，**MT起動**と**MT停止**に大別できる．

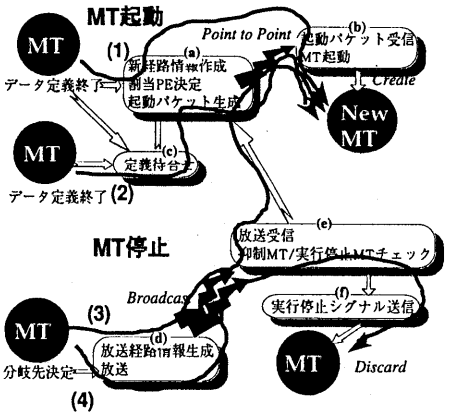


図7 分散制御のオーバヘッド

MT起動

(1) DummyMT→MT起動・MT→MT起動

図7(1)に示すように，あるMTにおいてデータ定義が終了してから，そのデータを必要とする次のMTの実行が開始されるまでの時間，あるいはデータ依存を持たないMTをDummyMTが生成するまでの時間である．すなわち，以下の時間である．

(a)新MT用経路情報生成→割付PE決定→起動用パケット送信→ (point to point通信)

(b)起動パケット受信→MT起動

(2) 複数MT→MT起動

図7(2)に示すように，複数MTからデータ依存を持つMTを起動する場合に，最後に定義されたデータが定義が終了してから，そのデータを必要とする次のMTの実行が開始されるまでの時間である．すなわち，以下の時間である．

(c)データマッチング処理→

(a)新MT用経路情報生成→割付PE決定→起動用パケット送信→ (point to point通信)

(b)起動パケット受信→MT起動

(3) 最大先行評価段数Nで起動抑制されたMT

起動対象となったMTが最大先行評価段数Nを超えた投機的実行である為に，起動が抑制された状態にあるとする．この時，図7(3)に示すように，他のMTからの経路情報放送により起動可能となった場合にMTが起動されるまでの時間である．すなわち，以下の時間である．

(d)放送経路情報生成→放送→ (broadcast通信)

(e)放送受信→抑制中MTチェック→

(a)新MT用経路情報生成→割付PE決定→起動用パケット送信→ (point to point通信)

(b)起動パケット受信→MT起動

MT停止

(4) 投機的実行失敗時のMT停止処理

図7(4)に示すように，投機的実行中のMTが投機

に失敗した為に，実行を停止するまでの時間である．すなわち，以下の時間である．

(d)放送経路情報生成→放送→ (broadcast通信)

(e)放送受信→実行停止MTチェック→

(f)実行停止シグナル送信→MT実行停止

3.2 並列計算機EM-4へのインプリメント

EM-4は，80台のPEからなるデータ駆動機構を持つ並列計算機であり，データ駆動機構によるスレッド間の高速な通信同期が行える．各PEは，12.5MHzのクロックで動作し，メモリ参照命令が2クロックであるのを除き大部分の命令は1クロックで実行される．ネットワーク性能はPEのポート当たり60.9Mbytes/secである．

EM-4上に，**2**節で述べた分散制御方式を図8に示すように2種類の方法でインプリメントした．制御部分とMT部分を1PEに共存させる**(a)制御共存型**と，これらを別PEで処理する**(b)制御分離型**である．これらを比較することにより，制御部分の処理が全体の実行時間に与える影響を調べることができる．なお，Broadcast Controllerは放送を司るため，全PEで動作している．MT実行に使用したPE数は32台であり，(b)制御分離型では，制御用にさらに32台を用いた．MTの制御は，EM-C[9]によりソフトウェアで記述し，Broadcast ControllerとSchedulerについてアセンブラレベルで最適化を行った．

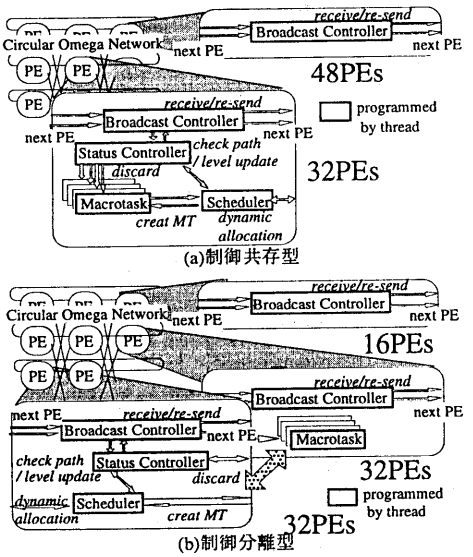


図8 分散制御のEM-4へのインプリメント

3.3 制御オーバヘッド基本値

EM-4上で，**3.1**で分類したオーバヘッドの基本値をアセンブラ命令をカウントすることにより求めた(**表1**)．つまり，他の負荷が存在しない場合の理想的な値である．但し，Broadcastは実測値である．

表1 制御オーバヘッドの基本値

	時間(μs)
Broadcast	13.2
(a)	17.2
(b)	6.5
(c)	3.3
(d)	7.8
(e)	16.7+9.6×MT数[1]
(f)	4.0

1 該当PEで実行中のMT数

4．並列計算機EM-4による評価

本節では，前節で求めた(1)～(4)の制御オーバヘッドの内，(1)と(4)の制御オーバヘッドが実行時間に与える影響について調べる．なお，(2)は(1)とほぼ同等と考えることができる．また，(3)についての評価は今後の課題である．

4.1 評価に用いたプログラム

評価に用いたプログラムは，**図9**(a)に示すようにループを構成する．このループは，BT1（BTは基本タスク[1]を示す）において一方（例えばBT2側）が選択され続けるとデータ依存が継続して存在する（図9(b)）が，BT2とBT3が交互に選択された場合，データ依存関係が存在しなくなるループであり，Boolean Recurrenceループと呼ばれる．本プログラムから文献[1]の手順に従ってMTを生成すると，図9(c)に示す4つのMTが得られる．これら4つのMTの内，MT2とMT4は他のMTからデータ依存を持たないので，投機的実行を行う場合，いつでも起動可能である．例えばループの繰返回数が20の場合，各イテレーションについてMT2とMT4の2個のMT，合計40個のMTが他のMTに関わらず起動可能となる．これに対し，MT1とMT4は，前イテレーションのMTからデータ依存を持つため，前イテレーションのMTが終了しなければ起動できない．

本評価では，繰返回数を20とし，条件分岐は，投機的実行の効果が最も表れるBT2とBT3が交互に選択される**パタンA**(BT1-BT3-BT1-BT2-BT-1-BT3..)と，

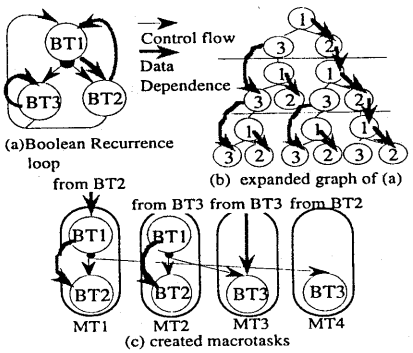


図9 評価に用いたプログラム

全く投機的実行の効果が得られないBT2が連続して選択される**パタンB**(BT1-BT2-BT1-BT2-....)を用いた．

4.2 投機的実行の効果

3.2で述べた**(a)制御共存**と**(b)制御分離**による性能を比較した．結果を**図10**と**図11**に示す．

図10は，投機的実行しない場合を基準とした速度向上率を表している．パタンAでは，BTの実行時間が14.4μs以上で投機的実行の効果が得られているにの対し，パタンBでは，投機的実行の効果が得られていない．これは，元々，パタンBは，データ依存が継続するために投機的実行の効果が表れないからである．このようなパタンに対して投機的実行を行うと，制御オーバヘッドが表れ，投機的実行をしない場合に比較して速度が低下する．すなわち，現在のインプリメントでは，投機的実行の効果のないパタンが選択された場合，実行時間が逆に増大してしまうことを示す．

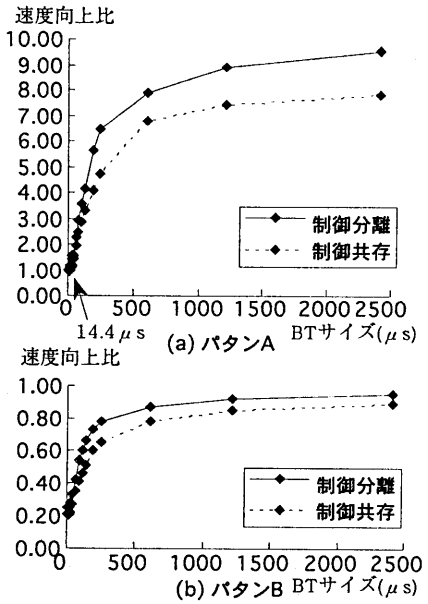


図10　投機的実行の効果

図11は，制御共存型を基準とした時の制御分離型の速度向上率を示す．図11に示すように制御の分離により，20～30%の速度向上が得られる．BTサイズが増加した場合，全体の実行時間に占める制御オーバヘッドの割合が減少し，これらの差は小さくなる．しかし，実プログラムで対象とするタスクサイズは，我々が対象とするHTG[10]の最下層で7～9μs，最下層から2番目の層で150～300μsであり[1]，この付近では，制御分離の効果がよく表れている．なお，EM-4での実行トレースを解析したところ，制御を分離しても，制御用のプロセッサの負荷が重く，十分な効果が得られていないことがわかった．そこで，

4.3以降では，制御分離型において，各種制御オーバヘッドを変化させ，実行性能に与える影響を調べた．

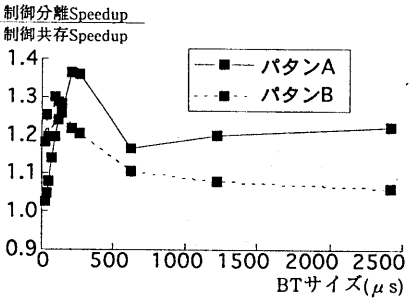


図11　制御分離の効果

4.3 Broadcast時間が実行時間に与える影響

表1のBroadcast時間を2倍，4倍に設定することにより，Broadcast時間が実行時間に与える影響を調べた．結果を図12に示す．図12は，図10での速度を基準とした時の速度比を示している．図12より，Broadcast時間が実行時間に与える影響は小さいことがわかる．これは，Broadcastのレイテンシが増大しても，各PEからのBroadcast要求がパイプライン的に並列処理されているからだと考えられる．

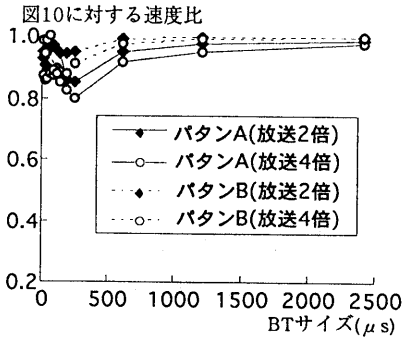


図12　Broadcast時間が性能に与える影響 (制御分離型)

4.4 MT→MT起動オーバヘッドが実行時間に与える影響

3.1の(1)で述べたMT起動オーバヘッドが実行時間に与える影響を，MT起動オーバヘッドを2倍，4倍に設定することにより調べた．結果を**図13**に示す．性能低下は，**4.3**とほぼ等しく，小さいことがわかる．しかし，パタンAにおいてMT起動時間を4倍にした際の性能低下が著しく，BTサイズが200μsまで，ほぼ40%の性能低下となっている．実行トレースを解析したところ，最初のDummyMTでのMT2とMT4の起動処理時間がネックになっているためだとわかった．すなわち，パタンAのように，全くデータ依存を持たないMTが最終的に選択されるような場合に，

MT起動オーバヘッドが大きいと十分な性能が得られないことがわかる．

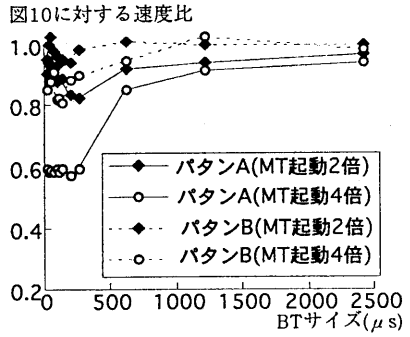


図13 MT起動オーバヘッドが性能に与える影響 (制御分離型)

4.5 投機的実行失敗時のMT停止オーバヘッドが実行時間に与えるの影響

3.1の(4)で述べたMT停止オーバヘッドが実行時間に与える影響を，MT停止オーバヘッドを2倍，4倍に設定することにより調べた．結果を**図14**に示す．図14に示すように，MT停止オーバヘッドは投機的実行の効果に大きな影響を与えることがわかる．**3.1**の(4)で述べたように，MT停止オーバヘッドは，制御オーバヘッド基本値の(d)(e)(f)で構成される．基本値の(d)(e)(f)の中でも，表1に示したように(e)の影響が最も大きい．そこで，**4.6**では，制御オーバヘッド基本値(e)が与える影響を調べる．

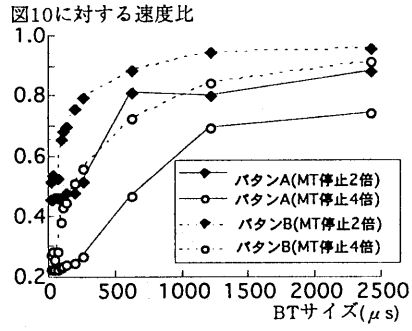


図14 MT停止オーバヘッドが性能に与える影響 (制御分離型)

4.6 放送受信処理(基本値(e))オーバヘッドが実行時間に与える影響

表1の制御オーバヘッド基本値(e)，すなわち，放送受信処理にかかる時間を2倍，4倍に設定し，実行時間に与える影響を調べた．結果を**図15**に示す．図15の結果は，ほぼ図14の結果と一致しており，MT停止オーバヘッドの主原因が，放送受信処理であることがわかる．

放送受信処理では，放送された制御情報を受信し，まず，自PEが担当しているMTに必要な情報かどうかを判断する．そして，その判断結果に基づいて，投機的実行中のMTを停止させたり，投機的実行中のMTが最終的に正しかったことを判断する．すなわち，放送受信処理の処理速度は，図1におけるControl DependenceのResolving Speedに相当する．Control DependenceのResolving Speedが遅くなると，MT自体は，実行を開始しているのに(あるいは既に実行が終了しているのに)，そのMTが有効なのか無効なのかを判断できない．このため，図15に示すように，実行性能に大きく影響を及ぼす．現在のEM-4上へのインプリメントでは，放送受信処理をソフトウェアで実現しており，オーバヘッドが大きくなっている．

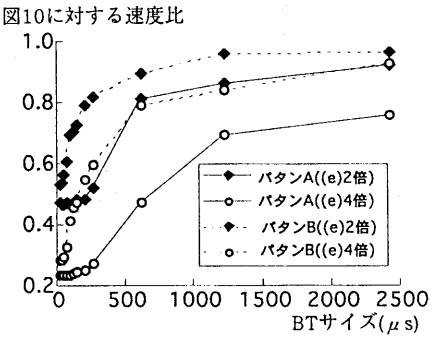


図15 基本値(e)のオーバヘッドが性能に与える影響 (制御分離型)

4.7 分散制御オーバヘッド削減手法と制御部分のハードウェア化の検討

制御オーバヘッド基本値(e)が削減できた場合の性能を予測する．**4.6**で得られた基本値(e)による実行時間増大の絶対値を元に，基本値(e)が1/2及び1/4に削減された場合の実行時間を算出した．

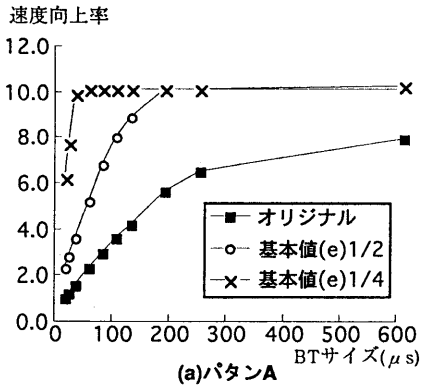
結果を**図16**に示す．図16より，基本値(e)が現在の1/2に削減された場合(基本値(e)=13μs程度)，現在のソフトウェアによるインプリメントの約2倍の性能を出せることがわかる．また，1/4に削減された場合(基本値(e)=7μs程度)，約3倍の性能向上となる．

パタンAでは，速度向上率が10で飽和している．理論上の最大速度向上は13.3倍であり，10倍以上の性能が出ていないのは，表1で示した他の制御オーバヘッドが原因であると考えられる．パタンBでも0.95で飽和しており，理論値の1にならない．これも，表1で示した他の制御オーバヘッドが原因である．このように，表1で示した他の制御オーバヘッドによって，BTサイズを増加させた場合も，理論値と等しくはならない．しかし，図16からわかるように，基本値(e)を改善することによる速度向上は非常に大きい．

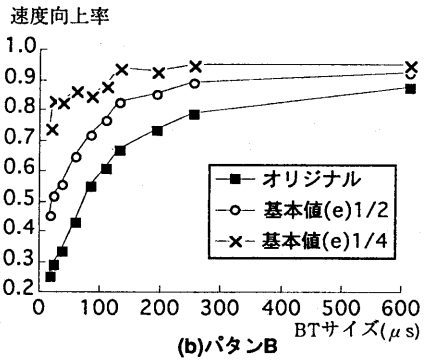
さらに，図16(b)より，投機的実行の効果の全くな

いパスが選択された場合も，BTサイズが20μs以上あれば速度低下を20%に，BTサイズが100μs以上あれば速度低下を10%にまで抑えられることが分かる．

4.2及び**4.7**から得られた知見を元に，我々は現在，分散制御部分をマクロタスク実行部分と分離し，かつ，ハードウェアによって実現することを検討している．基本値(e)の制御は，放送された制御情報を各ビットフィールドに分解し，PE自身が持っている経路情報と比較，再構成する処理であり，ソフトウェア上でのステップ数は大きくなるが，ハードウェアにより実現できれば，処理時間を大幅に短縮できるものと考えられる．



(a)パタンA



(b)パタンB

図16 基本値(e)を削減した場合の速度向上(制御分離型)

5. まとめ

本報告では，分散制御方式に伴うオーバヘッドの発生原因を分類し，各プリミティブが投機的実行の効果に与える影響を調べた．

並列計算機EM-4上に分散制御方式をソフトウェアにより実現して検証した結果，放送受信処理（制御オーバヘッド基本値(e)）が最も性能に影響を与えることがわかった．放送受信処理をハードウェアで実現し，処理時間が1/2に削減された場合(基本値(e)=13μs程度)，現在のソフトウェアによるインプリメントの約2倍の性能を出せることがわかった．また，1/4に削減された場合(基本値(e)=7μs程度)，約3倍の性能向上が期待できることがわかった．

Boolean Recurrence Loopを用いた評価では，ハードウェア化により放送受信処理が7μs程度になったと仮定すると，理論速度向上が13.6倍の時，最大10倍の速度向上を得られ，理論速度向上が1，すなわち，投機的実行の効果が全くないパスが選択された場合，10%～20%の性能低下で済むことがわかった．

現在，これらの制御機構のハードウェア化の検討を行っており，今後，より詳細なデータをとり，さらに検討を進めて行く予定である．なお，分散制御方式と平行して，我々が提案している投機的実行に適したマクロタスク生成手法を実際のプログラムを用いて評価すると共に，マクロタスクを自動的に生成するコンパイラの開発も行う予定である．

謝辞

本研究を遂行するにあたり御指導，御討論いただいた太田情報アーキテクチャ部長ならびに計算機方式研究室の同僚諸氏に感謝いたします．

参考文献

[1] Hayato Yamana, Mitsuhisa Sato, Yuetsu Kodama, Hirofumi Sakane, Shuichi Sakai, and Yoshinori Yamaguchi: "A Macrotask-level Unlimited Speculative Execution on Multiprocessors", Proc. of ICS'95, Balcelona, Spain, pp.328-337 (1995.7)

[2] 山名，佐藤，児玉，坂根，坂井，山口:"並列計算機EM-4におけるマクロタスク間投機的実行の分散制御方式",情報処理学会論文誌, Vol.36, No.7, pp.1-7 (1995.7)

[3] 山名，佐藤，児玉，坂根，坂井，山口:"投機的実行の現状とUnlimited Speculative Execution Schemeの提案"，情報処理学会研究報告，ARC-107-14，pp.105-112(1994.7).

[4] A.Nicolau, J.A.Fisher:" Measuring the Parallelism available for Very Long Instruction Word Architecture", IEEE Trans. Comput., Vol.33, No.11, pp.968-976 (1984).

[5] E.Riseman, C.Foster:"The Inhibitation of Potential Parallelism by Conditional Jumps", IEEE Trans. Comput., Vol.21, No.12, pp.1405-1411 (1972).

[6] M.S.Lam, R.P.Wilson:"Limits of Control Flow on Parallelism", Proc. of 19th Ann. Symp. on Computer Architecture, pp.46-57 (1992).

[7] P.K.Dubey et.al.:"Single-Program Speculative Multithreading(SPSM) Architecture: Compiler-assisted Fine-Grained Multithreading", Proc. of PACT'95, pp.109-121 (1995.6)

[8] S.Sakai,Y.Yamaguchi,K.Hiraki, Y.Kodama,T.Yuba : "An Architecture of a Dataflow Single Chip Processor", Proc. of 16th Ann. Symp. on Computer Architecture, pp.46-53(1989).

[9] 佐藤，児玉，坂井，山口:"並列計算機EM-4の並列プログラミング言語EM-C",情報処理学会論文誌, Vol.35, No.4, PP.551-560(1994).

[10] M.Girkar, C.D.Polychronopolos:"Automatic Extraction of Functional Parallelism from Ordinary Programs", IEEE Trans. Parallel & Distributed Syst., 3, 2, pp.166-178 (1992).