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

山名早人[†] 佐藤三久[†] 児玉祐悦[†]
坂根広史[†] 坂井修一[††] 山口喜教[†]

本論文では，マクロタスクと呼ぶタスクレベルでの投機的実行を並列計算機上で行うマクロタスク間投機的実行の効果的な制御手法として分散制御方式を提案する．一般に，投機的実行における理想的なモデル（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．しかし，実際には，投機的実行時に発生する制御オーバヘッドのために，上記の理論性能に近づくのは難しく，制御オーバヘッドの小さい制御手法が必要とされる．本論文で提案する分散制御方式は，各マクロタスクが(1)自分の後続のマクロタスク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各マクロタスク自身が次の状態を決定することにより実現される．これにより，マクロタスクの制御を並列化できると共に，マクロタスク制御オーバヘッドがマクロタスク数に依存しなくなり，高速な投機的実行が可能となる．本方式を並列計算機 EM-4 上にインプリメントし，Boolean Recurrence ループを用いて評価した結果，従来提案されている一般的なタスク制御方式である集中制御方式を用いた場合に比較し，マクロタスク制御オーバヘッドを小さくできることを確認した．

A Distributed Control Scheme of Macrotask-level Speculative Execution on the EM-4 Multiprocessor

HAYATO YAMANA,[†] MITSUHISA SATO,[†] YUETSU KODAMA,[†] HIROHUMI SAKANE,[†]
SHUICHI SAKAI[††] and YOSHINORI YAMAGUCHI[†]

The purpose of this paper is to propose a new fast control scheme of macrotasks with speculation. A macrotask is a coarse grain task which is a unit of speculation. Previous works have reported that the speedup ratio is 12 to 630 times in comparison with conventional execution schemes without speculation when both the speculation depth and the computational resource are infinite, that is called *oracle model*. The control overhead, however, prevents the speedup from attaining the theoretical speedup ratio. Thus, the control scheme with small overhead is desired. The distributed control scheme archives small control overhead by adopting two mechanisms—1) Each macrotask creates its successive macrotasks and 2) Each macrotask snoops the broadcasted control signals and determines its next state by itself. Thus, the control of macrotasks can be parallelized and the overhead to control macrotasks does not depend on the number of macrotasks. The scheme has been implemented on the EM-4 multiprocessor. Preliminary evaluations using Boolean Recurrence loops show that the control overhead of the proposed scheme is smaller than that of conventional control schemes-centralized control schemes.

1. はじめに

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，タスクレベルでの投機的実行を並列計算機上で行うマクロタスク間投機的実行[1]の効果的な制御手法として分散制御方式を提案する．

プログラムを並列化する際の問題点に，制御依存に基づく実行順序関係（先行制約）のために十分な並列性が得られないという点がある．例えば，以下の例では，S2の実行がS1の条件分岐で決定されるので，S1において制御が確定するまでS2を実行できない．

```
if (A) then        S1
    B=C×D          S2
endif
```

この時，S2はS1に制御依存していると言う．これに対し，S2とS1の間の先行制約を無視してS2の実

† 電子技術総合研究所情報アーキテクチャ部
Electrotechnical Laboratory, Computer Science Division
†† 新情報処理開発機構つくば研究センタ
Real World Computing Partnership, Tsukuba Research Center

行を開始し，後にS1の制御が確定した時点でS2の結果の有効・無効を判断する方法を投機的実行と呼ぶ．また，投機的実行を行う際，先行制約を受ける条件分岐数を先行評価段数と呼ぶ．

従来，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[2]．これに対し，マクロタスク間投機的実行[1]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[3]適用時の理想的な速度向上率に近づくことができる[2]．Oracle Modelとは，条件分岐の結果が実行前にすべて既知であり，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[2),4),5]．また，doall型ループ等の高並列部分を越える投機的実行を行わないという現実的な仮定の元でも，2～9倍の速度向上が得られる[1]．しかし，実際には，投機的実行時に発生するMTの制御オーバヘッドのために，上記の理論性能に近づくのは難しく，制御オーバヘッドの小さいMT制御手法が必要とされる．

従来，一般的なタスクの実行制御手法として，個々のタスクの制御条件を記述したタスク管理テーブルを用いた集中制御手法が提案されている[6]．しかし，集中制御手法をマクロタスク間投機的実行に用いた場合，(1) 制御確定後のMTで得られた分岐方向決定情報を用いた制御を行わなければならず，マクロタスク制御が制御フロー順に直列化される，(2)MT数の増大に伴いタスク管理テーブル更新のオーバヘッドが大きい，という問題がある．本報告では，これらの問題を解決する制御方式として，集中制御を廃したMT制御手法を提案し，並列計算機EM-4[7]上にインプリメントする．インプリメントにおいては，EM-4が持つ，(1)直接データ待ち合わせ機構[8]，(2) サーキュラオメガネットワーク[8]，(3)負荷最小のPEを検出することのできるMLPEパケット[9]，を利用し，MTを制御する際のオーバヘッドを軽減させる．

以下，2章では，集中制御方式の問題点を明らかにし，3章で，これらの問題点を解決する分散制御方式を提案する．4章では，提案方式の有効性を並列計算機EM-4上で検証する．

2. 集中制御方式の問題

本章では，集中制御方式をマクロタスク間投機的実行に用いた場合の問題点を明らかにする．

2.1 集中制御方式

従来提案されているタスクの一般的な実行制御手法[6]では，制御プロセッサ上でタスクの実行開始条件を集中管理する．制御プロセッサでは，タスク管理テーブルに各タスクの実行開始条件を持ち，実行中のタスクから送られてくる(1)タスクの終了，および(2)分岐方向決定情報を元に，次に実行可能なタスクを選び実行を開始させる．

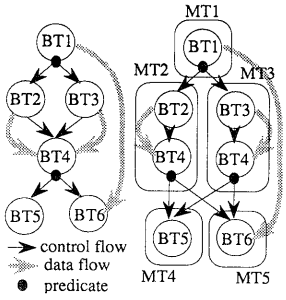
本制御手法は，マクロタスク間投機的実行の制御手法として容易に拡張できる．マクロタスク間投機的実行では，各MT（MTは1つ以上の基本タスク（BT）から構成される[1]）を3つの制御条件（実行開始・制御確定・実行停止）により制御する[1]．実行開始条件は，MTへ入力するデータの定義終了を保証し，制御確定条件・実行停止条件はそれぞれMTへの制御の到達・非到達を保証する．これら3条件をタスク管理テーブルに持ち，実行中のMTから送られてくる(1)BTの終了，および(2)分岐方向決定情報，を元にMTの制御を行う．すなわち，(a)投機的実行の開始，(b)制御が到達した場合の制御確定通知，(c)制御が非到達の場合の実行停止の3制御を行えばよい．

2.2 MT制御オーバヘッドと集中制御の問題点

MT制御オーバヘッドとは，MTから送られてくるBT終了および分岐方向決定情報から上記の(a)～(c)の処理が完了するまでの時間である．

マクロタスク間投機的実行に集中制御を用いた場合，(1)制御確定後のMTにおいて得られた分岐方向決定情報を用いるため[1]，MT制御が制御フロー順に直列化される，(2)MT数の増大に伴いタスク管理テーブル更新のオーバヘッドが大きい，という理由からMT制御のオーバヘッドが大きくなる．

(1)を，**図1**の例により説明する．図1(a)のプログラムは，文献1)の手順により，図1(b)のようにマクロタスク化され，タスク管理テーブル（文献1)ではMT制御条件と記述）は図1(c)のようになる．MT1は制御確定済のMTとして起動され，MT2～MT4は，投機的実行として起動される．今，MT3がMT1での分岐方向決定に先立って終了し，分岐方向決定情報4-5（BT4でBT5側に分岐）が得られたとする．分岐方向決定情報4-5は，MT3が投機的実行中であるため，制御プロセッサに報告できない．これは，制御プロセッサに報告した場合，タスク管理テーブルに従ってMT4が制御確定してしまい，誤った制御となるからである．すなわち，MT1内のBT1で決定される分岐方向決定情報を先に処理し，その後，MT3が制



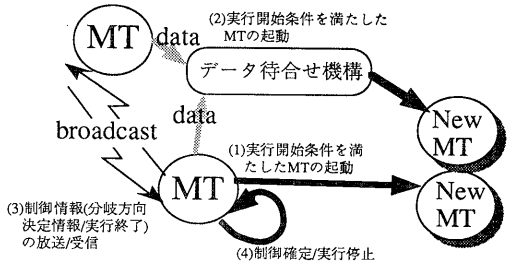
(a) 基本タスク (b) マクロタスク

	実行開始	制御確定	実行停止
MT1	True	True	False
MT2	True	1-2	1-3
MT3	True	1-3	1-2
MT4	True	4-5	4-6
MT5	1	4-6	4-5

1はBT1の終了を，1-2はBT1でBT2側の分岐が選択されることを示す．

(c)タスク管理テーブル

図1 集中制御方式
Fig. 1 A centralized control scheme.



(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2 分散制御方式
Fig. 2 A distributed control scheme.

御確定となった時点（BT 1 で BT 3 側に分岐した時点）で分岐方向決定情報 4-5 を制御プロセッサに報告しなければならない．このように，集中制御方式では，制御確定後の MT で得られる分岐方向決定情報を用いるため，MT 制御の処理が制御フロー順に直列化され，MT の実行と MT 制御の並列化ができない．

（2）は，図 1（c）で示したタスク管理テーブルの更新処理時間がタスク管理テーブル内の MT 数に比例して大きくなることを示す．これは，タスク管理テーブルの条件更新，および，条件が成立した MT に対する制御が MT ごとに逐次処理されることに起因する．

以上のような MT 制御オーバヘッドがタスク管理テーブルを用いた集中制御方式に存在する．

3. 分散制御方式

2 章で述べた問題点を解決するマクロタスク間投機的実行の制御方式として分散制御方式を提案する．

3.1 概　　要

分散制御方式の概要を**図 2** 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．

分散制御を実現するためには，次の 4 つの機能が必要となる．

（1） **データ待ち合わせ機構**：新しく起動する MT が他の複数の MT からデータ依存を持つ場合，必要とする全データが定義されたことを保証する．

（2） **制御情報の放送機構**：各 MT 内で決定した分岐方向決定情報を他のプロセッサへ放送する．

（3） **MT の動的割り当て機構**：生成された MT を負荷の小さいプロセッサへ割り当てる．

（4） **PE 内制御判断機構**：放送された制御情報を元に，投機的実行中の MT の次の状態（実行継続/実行停止）を判断する．

これらの機能を一般の並列計算機上で実現する場合，（1）および（3）の機能をハードウェアレベルでサポートしていないため，分散制御に伴うオーバヘッドが大きく，分散制御の効果が期待できない．これに対して，EM-4 では，データ待合せ機構として直接データ待ち合わせ機構[8]，MT の動的割り当て機構として負荷最小の PE を検出することのできる MLPE と呼ぶ特殊パケット[9]をサポートしており，分散制御に伴うオーバヘッドを小さくできる．また，（2）の機構については，EM-4 が持つサーキュラオメガネットワーク[8]上で 2 進木を構成し制御情報を全プロセッサへ放送し，（4）の機構については，ソフトウェアで実現する．

3.2 マクロタスクの起動

MT の起動条件は，MT が必要とするすべてのデータの定義の終了保証である．今，ある MTn が，MTn に先行する MTa からのみデータ依存を受けている場合，MTa でのデータ定義終了のみをトリガとして MTn を起動できる．この場合，「データの定義終了→新たな MTn の起動」という手順を MTa 内で処理でき，集中したタスク管理機構が不用である．以下では，データ依存により MT を分類し，起動方法を示す（**図 3**）．

（1） **データ依存を持たない MT**：他のどの MT か

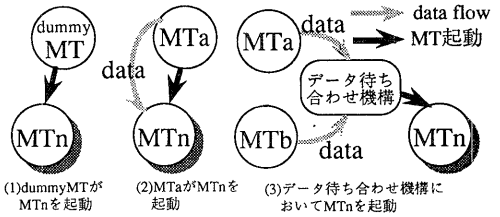


図3 マクロタスク（MT）の起動方法
Fig. 3 Initiation of macrotasks.

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に，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**：先行するある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．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 および BT 番号を用いる．

3.3 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ごと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 および 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 が持つ経路情報にすべてフラグ 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．

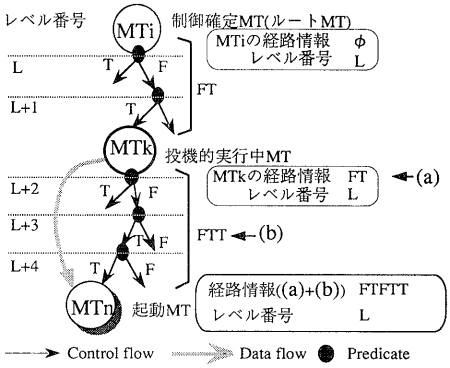


図4 経路情報とレベル番号
Fig. 4 Route informations and level numbers.

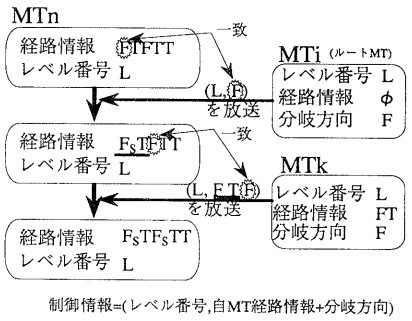


図5 放送される制御情報と処理例
Fig. 5 Broadcasted control informations and an example.

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4に例を示す．図4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5**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5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 L 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．

以下では，制御方法の詳細を順に示す．なお，MT が複数の制御情報 P を持つ場合には，個々の制御情報について，以下の処理を行う．

(a) 制御情報放送処理

ルート MT では，自 MT 内において条件分岐が処理されるごとにレベル番号をインクリメントすると同時に，(レベル番号，分岐方向)を制御情報として放送する．その他の MT では，(レベル番号，自 MT 経路情報＋分岐方向) を放送する．分岐方向は，自 MT 内での条件分岐評価後に得られた分岐先を T/F で表したものである．経路情報は，分岐方向が有効であるための前提条件として(b)以下の処理で用いる．

(b) 制御情報受信処理

受信制御情報のレベル番号を L_r，(経路情報＋分岐方向) 内の T/F/＊の数を n_r，(経路情報＋分岐方向) の最左から k 番目 ($1 \leqq k \leqq n_r$) の値を $P_r(k)$ で表す．同様に，自 MT での値を各々，L_m, n_m, $P_m(k)$ で表す．

受信制御情報中の分岐方向に対応する経路が MT の経路情報中に存在する場合，すなわち，受信制御情報が $L_m < L_r + n_r \leqq L_m + n_m$ を満たす場合に以下の(1)以降の処理を行い，満たさない場合は受信制御情報を破棄する．$L_r + n_r - 1$ は受信制御情報中の分岐方向のレベルを，L_m と $L_m + n_m - 1$ はそれぞれ MT が持つ経路情報中のレベルの最小と最大を表す．$L_r + n_r - 1 < L_m$ の場合，受信制御情報中の分岐方向は MT において既に確定済みの経路に対応し，MT 制御に不要となる．また，$L_m + n_m - 1 < L_r + n_r - 1$ の場合，受信制御情報の分岐方向は，MT が前提とする経路の先，すなわち未来の経路に対応し，MT 制御に不要となる．

なお，各 MT は，現在の MT のレベル L_m から $N-1$ 段前までの分岐方向の履歴を保持し，$L_r < L_m$ の際，受信制御情報中の経路情報が MT の経路情報と一致するかどうかの判断に用いる．ここで，$L_m < L_r + n_r$，かつ，n_r の最大値は N であるため，$L_m - L_r < N$ となり，$N-1$ 段前までの履歴を保持すれば十分である．

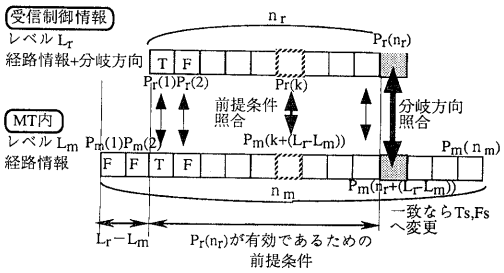


図6 経路情報の比較
Fig. 6 Comparison of route informations.

(1) **前提条件照合**：受信制御情報中の経路情報と MT が持つ経路情報との対応をとる．**図6** に示すように，$P_r(k)$ と $P_m(k+(L_r-L_m))$ が同一の分岐方向を表す．次に，$1 \leqq k \leqq n_r - 1$ の範囲 (受信制御情報が持つ分岐方向が有効であるための前提条件) で $P_r(k)$ と $P_m(k+(L_r-L_m))$ とを比較する．なお，$k+(L_r-L_m)$ の値がマイナスになる場合は，MT が保持している過去の分岐方向と比較する．上記の範囲を満たす $\forall k$ について両者が一致した場合についてのみ(2)以下の処理を行う．異なる場合は，自 MT の経路情報と受信制御情報中の経路情報 (分岐方向が有効であるための前提条件となる経路) が一致しないことを示すので，受信制御情報を破棄する．一致となる組み合わせは，$(P_r(k), P_m(k+(L_r-L_m)))$ が，(T, T) (F, F) (＊, F) (＊, T) (＊, ＊) (F, ＊) (T, ＊) である．ただし，(F, ＊) (T, ＊)の場合には，MT が持つ経路情報の複製を作り，その複製に対して，$P_m(k+(L_r-L_m))$ を F あるいは T に設定し，以下の処理を行う．これは，MT が持つ経路情報が経路を特定していないのに対し，受信制御情報が経路を F あるいは T に特定しているためである．

(2) **分岐方向照合**：$P_r(n_r)$ と $P_m(n_r+(L_r-L_m))$，すなわち，受信制御情報内の分岐方向と自 MT の経路情報の対応部分とを比較する．両者が一致した場合，自MT中の比較対象の経路情報部分に条件が成立済みであるフラグsを付ける．不一致の場合は，自MTが選択されなかったことを示すので，自MTの投機的実行を停止する．一致となる組み合わせは，$(P_r(n_r), P_m(n_r+(L_r-L_m)))$ が，(T, T) (T, ＊) (F, F) (F, ＊) であり，前者の2つの場合には，Ts に，後者の2つの場合には，Fs に変更する．

(3) **レベル番号更新**：$\exists j$, $1 \leqq j \leqq n_r$ について

$P_m(k)$, $1 \leqq k \leqq j$ がすべて Ts あるいは Fs である時，自 MT の経路情報の $P_m(1) \sim P_m(j)$ までを削除し，レベル番号を j 増加させる．これは，j 番目までの条件が成立していることを示す．この時，自 MT の経路情報長が 0 となれば，ルート MT になったことを示す．

(4) **MT 起動**：3.2 節で述べた MT 起動コードが存在する場合には，図 4 で示す経路情報とレベル番号を持つ MT を起動させる．この時，経路情報長（図 4 の FTFTT では 5）が起動 MT の先行評価段数に一致するので，この値が最大先行評価段数 N より大きい時には，起動をさせない．これにより，最大先行評価段数を制御する．

(5) **レベル番号回収処理**：レベル番号は有限資源であるため，使い果たさないように，実行中に回収する必要がある．回収・再利用法を以下に示す．レベル番号の最大値を L_{max} とする．次にレベル番号を 2 グループ，0～L_{mid} と $L_{mid}+1 \sim L_{max}$ に分割する．今，レベル番号を 0 から使い始め，ルート MT において，レベル $L_{mid}+1$ の制御情報を放送したとする．この時，レベル L_{mid} までのレベル回収フラグを制御情報に付加する．レベル回収フラグ付きの制御情報を受け取った全 PE は，自 PE 内で実行中のすべての MT のレベルが $L_{max}+1$ 以上になった時点で，回収済報告を回収処理を担当する PE（あらかじめ指定）に対して行う．回収処理担当 PE では，すべての PE から回収報告を受けた後，全 PE に対して 0～L_{mid} までのレベル番号の再利用が可能となったことを放送する．これにより，各 PE は，0～L_{mid} までのレベル番号を再利用できる．このように，レベル番号をグループごとに回収し，再利用する．また，L_{max} をあらかじめ大きくとることにより，MT の実行と並行して，レベル番号の回収ができる．

3.4 集中制御との比較

集中制御および分散制御時に発生する MT 制御オーバヘッドの比較を**表 1** に示す．MT 制御オーバヘッドとは，実行中の MT から送られてくる BT 終了および分岐方向決定情報から MT に対して，実行開始・制御確定・実行停止の制御を行うまでの時間である．

集中制御では，制御情報は「実行中 MT →制御プロセッサ→制御対象 MT」のように，一対一の通信となるのに対し，分散制御は「実行中 MT →全 MT」のように，放送となる．ただし，分散制御では，新たに MT を起動する場合，「実行開始要求 MT →起動先 MT」となり放送は不要である．一対一通信に必要な時間を T_1（EM-4 上へのインプリメントでは 3.3 μs），放送に必要な時間を T_b（同 26.4μs）とし，集中制御・分散制御時の制御にかかる時間をそれぞれ，C_c，C_d とすると，MT 制御オーバヘッドは，表 1 のようになる．

表 1 集中制御と分散制御方式の比較
Table 1 The comparison between the centralized control scheme and the distributed control scheme.

	集中制御方式	分散制御方式
MT 制御オーバヘッド	$2T_1+C_c$	C_d+T_1 (MT 起動処理) T_b+C_d (MT 制御確定・実行停止処理)
MT 制御順序	制御確定順に MT 制御 (MT 制御は直列化)	分岐方向の決定順に MT 制御 (MT 制御は並列化)

T_1：1 PE 対 1 PE の通信時間
T_b：1 PE 対全 PE への放送通信時間
C_c：タスク管理テーブルの操作時間（MT 数に比例）
C_d：各 MT における制御情報処理時間（MT 数に無関係）

ここで，C_c は，2.2 節で述べたように MT 数に比例する．一方，分散制御では自 MT の処理のみを行うので C_d は，MT 数に関わらず一定である．また，集中制御では，制御確定後の MT で得られる分岐方向決定情報を用いるため，MT 制御が制御確定順でしか行えず，MT 制御が直列化される．一方，分散制御では，投機的実行中の MT で得られた分岐方向決定情報を用いた制御ができ，MT 制御を並列化できる．

以上より，MT 数が増加した際，分散制御の方が制御オーバヘッドを小さくできると考えられる．

4. 分散制御の有効性評価

集中制御方式および分散制御方式を EM-4[7] 上へインプリメントし，分散制御の有効性の評価を行った．

4.1 並列計算機 EM-4

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 12.5 MHz 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 60.9 Mbytes/sec である．

4.2 EM-4 上へのインプリメント

各方式の EM-4 上へのインプリメントの概要を**図 7** に示す．MT の制御は EM-C[10] によりすべてソフトウェアで記述した．集中制御では，タスク管理テーブルの 3 つの制御条件を並列処理するため，3 つのスレッドに処理を分割し，3 台の PE を用いて処理を行い，残りの 77 台の PE に MT を割り付ける．分散制御では，80 台の各 PE 内に図 7(b) で示される制御関係の 3 スレッドおよび MT のスレッド（1～4 スレッド）を同時に起動させ処理を行った．なお，投機的実行する際，

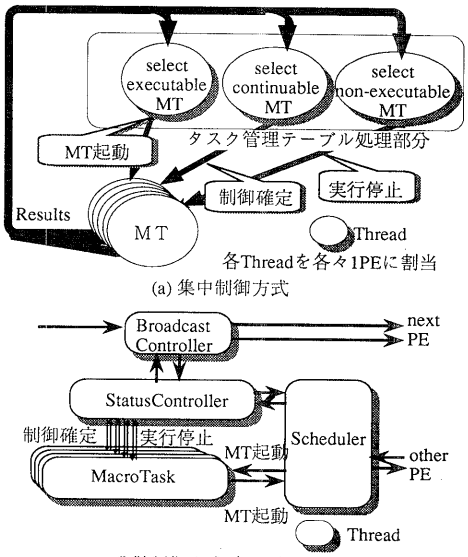


図7 集中/分散制御方式の EM-4 上へのインプリメント
Fig. 7 An implementation of the centralized/distributed control scheme on the EM-4 multiprocessor.

先行評価段数の制限をしないようにするため，最大先行評価段数 N は十分に大きく設定し，32 とした．また，レベル番号の最大値 L_{max} は $2^{23}-1$ とした．

4.3 評価対象プログラム

評価に用いたプログラムは，図8(1)に示すようにループを構成する．このループは，BT 1 (BT は基本タスク[1])を示す) において一方 (例えば BT 2 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8(2))が，BT 2 と BT 3 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[11]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8(3)に示す4つの MT が得られる．これら4つの MT のうち，MT 2 と MT 4 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 10 の場合，各イテレーションについて MT 2 と MT 4 の2個の MT，合計 20 個の MT が他の MT に関わらず起動可能となる．これに対し，MT 1 と MT 4 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．従って，本プログラムを投機的実行する際に必要となる PE 数は，1 MT を 1 PE に割り当てるとし，先行評価段数を制限しないとすると，(繰返回数×2) となる．

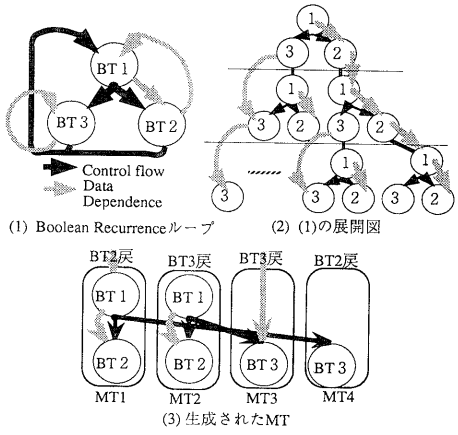


図8 評価プログラム
Fig. 8 A workload.

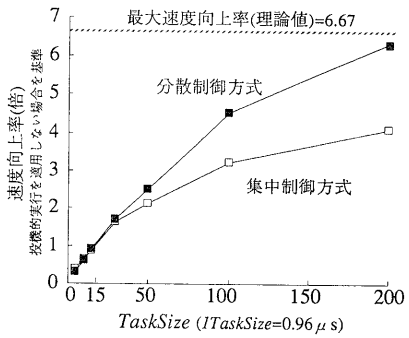


図9 投機的実行による速度向上 No. 1 (使用 PE 数=20)
Fig. 9 Speedup with speculation No. 1 (# of used PE=20).

4.4 タスクサイズによる投機的実行の効果

MT 制御オーバヘッドによる実行時間の増大を無視でき，投機的実行の効果を得ることができるタスクサイズを調べる評価を行った．評価では，各 BT のタスクサイズは同一とし，各々 (0.96 μs× *TaskSize*) 時間分のダミーの実行を行い，BT 間のデータ依存においては，1 データ (32 bit) を受け渡すとした．ループの繰返回数は 10 とし，BT 1, BT 2, BT 1, BT 3,.... のように各イテレーションの条件分岐において BT 2 と BT 3 を交互にとるとした．この時，MT 制御オーバヘッドを 0 とした理論的な最大速度向上率は，6.67 倍となる．MT 実行用に用いた PE 数は 20 台である．

投機的実行を行わず，図8のプログラムを1台のプロセッサで実行した時の実行時間を基準とした時の速度向上率を**図9**に示す．図9より，*TaskSize* が 15 以下では，集中・分散制御共に，投機的実行を行わない場合に比較して実行時間が増大している．これは，*TaskSize* が 15 以下では，投機的実行の効果に比較して

MT 制御オーバヘッドが大きいことを示す．また，*TaskSize* が増すにつれ，分散制御の方が速度向上率の立ち上がりが良い．これは，全体の実行時間に占める MT 制御オーバヘッドが分散制御の方が小さいことを示す．すなわち，表 1 に示したように，集中制御では，制御確定順に MT 制御を行うため，MT 制御にかかる時間が全体の実行時間に直列に加算される．これに対し，分散制御では，投機的実行中の MT においても分岐方向が決定次第，分岐方向決定情報を放送することにより，制御確定順ではなく，分岐方向決定順に処理できる．このため，全体の実行時間に及ぼす影響が小さくなるためであると考えられる．

4.5 MT 数による投機的実行の効果

実行 MT 数が投機的実行に与える影響を調べた．図 8 のプログラムの繰返回数を増減させ，実行 MT 数を変化させる．評価では，*TaskSize* を 50 に固定し，繰返回数を 5，10，20，30 と変化させた時の速度向上率を求めた．結果を MT 実行用に用いた PE 数と共に**図 10** に示す．

繰返回数が 10 までは，集中・分散制御共に速度向上率が増大している．これは，全体の実行時間に対する MT 制御オーバヘッドの割合が相対的に小さくなったためである．一方，繰返回数が 20，30 と増大すると，集中制御では，速度向上率が低下するのに対し，分散制御では，ほぼ一定で推移している．集中制御では，表 1 に示したように，MT 制御オーバヘッドが実行 MT 数に比例する．すなわち，繰返回数 10 の時には，タスク管理テーブルに最大 40 個の MT (4 MT×10 イテレーション) が，繰返回数 20 の時には最大 80 個の MT が登録される．このため，MT 制御オーバヘッドが増大し速度向上率が下がる．一方，分散制御では，制御オーバヘッドが MT 数に依存しないため，速度向上率は低下しない．しかし，MT 制御オーバヘッドにより，投機的実行の効果が飽和し，繰返回数を増やしても速度向上率が増加しないのだと考えられる．

4.6 通信ディレイが投機的実行に与える影響

分散制御では，制御に放送を用いるため，通信ディレイが投機的実行の効果に大きく影響を及ぼすことが予想される．そこで，通信ディレイが投機的実行に与える影響を調べた．評価では，4.5 節と同一条件でプログラムを実行し，通信ディレイのみを 4.5 節に比較して 2 倍，3 倍に変化させた．すなわち，一対一通信に必要な時間 T_1 (3.3 μs)，および，放送に必要な時間 T_b (26.4 μs) をプログラム中への nop 命令の挿入により，2 倍，3 倍に設定した．4.5 節で得られた実行時間と比較して増加した実行時間 (ΔT) を**図 11** に示す．

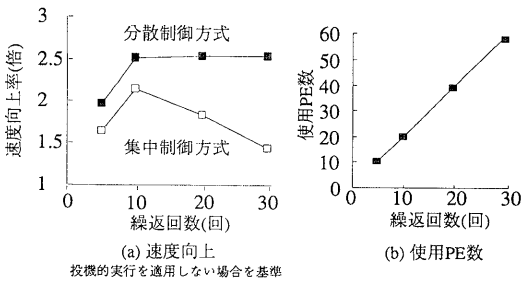


図 10 投機的実行による速度向上 No. 2 (*TaskSize*＝50)
Fig. 10 Speedup with speculation No. 2 (*TaskSize*＝50).

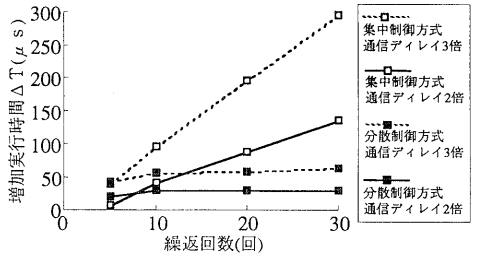


図 11 通信ディレイによる実行時間増加 (*TaskSize*＝50)
Fig. 11 Increased execution time with various communication delays (*TaskSize*＝50).

集中制御では，ΔT が繰返回数に比例して増加している．一方，分散制御では，繰返回数に関係なく，ΔT はほぼ一定であり，かつ，その値は増加させた通信ディレイにほぼ一致している．これは，表 1 に示すように，集中制御では，MT 制御にかかる時間 (通信ディレイを含む) が全体の実行時間に直列に加算されるのに対し，分散制御では，MT 制御が並列化されているため，MT 制御にかかる時間 (通信ディレイを含む) が全体の実行時間にあまり影響を及ぼさないためであると考えられる．さらに，分散制御で用いる放送は，ソフトウェアにより 2 進木を構成し実現しているため，複数の放送を並列処理できることも，通信ディレイによる影響を小さくできる理由であると考えられる．

以上述べたように，分散制御では，通信ディレイが大きくなった場合も，ΔT を通信ディレイの増加分程度の時間に抑えることができることがわかった．

4.7 結果に対する考察

4.4〜4.6 節での評価により，分散制御について，次の 3 点を確認することができた．

(a) タスクサイズの増大に伴い，理論性能に近い速度向上率が得られる．

(b) 実行 MT 数が増加した場合でも，速度向上率は低下せず，MT 制御オーバヘッドにより，投機的実行

の効果が飽和するまで速度向上率が増加する．

（c） 通信ディレイが大きくなっても，実行時間の増加は，通信ディレイ程度に抑えられる．

これらの特性により，分散制御方式は，MTを効率よく投機的実行できる方式であると考えられる．しかし，今回の評価では，小規模なプログラムを用いたため，いくつかの点で問題が残る．

（1） 実アプリケーションでの投機的実行の効果

実アプリケーションを投機的実行した場合の速度向上は本評価からは，判断できない．しかし，4.4節で求めた *TaskSize* と投機的実行による速度向上の理論性能に対する割合を用いて予測することができると考えられる．すなわち，*TaskSize* が15以下では投機的実行の効果は得られず，*TaskSize* が200程度で理論性能の90%程度を達成できることがわかる．従って，アプリケーションが持つ投機的実行の理論性能と *TaskSize* が与えられれば，本評価の結果得られたMT制御オーバヘッドの挙動から，投機的実行の効果を予測できる．しかし，理論性能はアプリケーションごとに異なり，かつ，実行時の条件分岐の方向，使用できるPE数によっても変化する．例えば，4.4節において使用PE数を10台とした場合，理論性能は，約半分に低下する．また，*TaskSize* は，文献1)のMT生成手法により大きくすることができるが，その大きさはアプリケーションに依存する．このため，これらのパラメータを整理した上で，実アプリケーションによる総合的な評価を行う必要がある．なお，投機的実行に適したアプリケーションとしては，SpecINTのベンチマーク中のeqntott等，整数演算系のアプリケーションが挙げられる[2),5)]．文献5)では，Oracle Model[3)]を仮定することにより630倍の速度向上が得られている．

（2） 最適な最大先行評価段数 N の設定

今回の評価では，$N=32$ とし，先行評価段数の制限をしないようにした．このように，N を十分に大きく設定すると，図10(a)に示したように，MT制御オーバヘッドによって投機的実行の効果が飽和するまで，速度向上が得られる．しかし，PE数が十分にない状況では，無駄な実行により逆に速度が遅くなることが考えられる．従って，図10(a)の場合には，繰返回数10回分（飽和する点）を常に投機的実行するように $N=10$ と設定すればよいと考えられる．MT制御のオーバヘッドが大きい場合や，使用できるPE数が少ない場合には N は小さく設定すべきである．さらに，1段の先行評価によって起動されるMT数が多い場合には，使用できるPE数が少なくなったのと同様に考えることができ，N は小さく設定すべきである．このように，最適な N は，MT制御のオーバヘッドと使用可能なPE数，そしてアプリケーションがもつMT間のデータ依存関係により決定する．

（3） 放送機構の改善

分散制御では，各MTが制御情報を放送すると共に，放送された制御情報を解釈し，各MTが自ら次の実行状態を決定することにより制御が進む．しかし，あるMTiが必要とする制御情報は，MTiが持つ経路情報の上流にあるMTからの制御情報のみである．すなわち，特定のMT群のみにマルチキャストするという手法も考えられる．本論文では，マルチキャスト先のMTの計算が複雑になると考えたため，採用していないが，今後，その有効性について検討を行いたい．

5. おわりに

本論文では，マクロタスク間投機的実行の制御手法として分散制御方式を提案した．分散制御方式は，(1)各MTが自分の後続の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各MT自身が自MTの次の制御を行うことにより実現される．これにより，MTの制御を並列化することができると共に，MT制御オーバヘッドがMT数に依存しなくなり，高速な投機的実行が可能となる．

分散制御方式を並列計算機EM-4[7)]上にインプリメントし，Boolean Recurrenceループ[11)]を用いて評価した結果，従来提案されている一般的なタスク制御方式である集中制御を用いた場合に比較し，MT制御オーバヘッドを小さくできることを確認した．また，タスクサイズを大きくすることにより，プログラムが持つ理論的な最大速度向上率に近づけることを確認した．

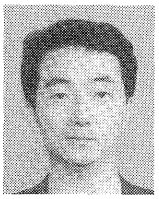
今後は，4.7節で述べたように，実アプリケーションを用いた評価を行うことにより，並列計算機上での投機的実行の効果を具体化していく予定である．特に，投機的実行に適したアプリケーションの選定，最適な最大先行評価段数 N の決定，制御情報伝達のマルチキャストによる実現，ソースプログラムから文献1)の手順に従って自動的にMTを生成するコンパイラの開発を行っていきたい．また，今回のインプリメントでは，MT制御をソフトウェアで実現したが，MT制御のオーバヘッドをさらに小さくするため，ハードウェア化も検討していく予定である．

謝辞 本研究を遂行するにあたりご指導，ご討論いただいた太田情報アーキテクチャ部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) 山名, 安江, 石井, 村岡：並列処理システムにおけるマクロタスク間先行評価方式, 電子情報通信学会論文誌 (D-I), Vol. J 77-D-I, No. 5, pp. 343-353 (1994).
2) 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).
3) Nicolau, A. and Fisher, J.A.: Measuring the Parallelism Available for Very Long Instruction Word Architecture, *IEEE Trans. Comput.*, Vol. 33, No. 11, pp. 968-976 (1984).
4) Riseman, E. and Foster, C.: The Inhibitation of Potential Parallelism by Conditional Jumps, *IEEE Trans. Comput.*, Vol. 21, No. 12, pp. 1405-1411 (1972).
5) Lam, M.S. and Wilson, R.P.: Limits of Control Flow on Parallelism, *Proc. of 19th Ann. Symp. on Computer Architecture*, pp. 46-57 (1992).
6) 本多, 合田, 岡本, 笠原：Fortran プログラム粗粒度タスクの OSCAR における並列実行方式, 電子情報通信学会論文誌 (D-I), Vol. J 75-D-I, No. 8, pp. 526-535 (1992).
7) Sakai, S., Yamaguchi, Y., Hiraki, K., Kodama, Y. and Yuba, T.: An Architecture of a Dataflow Single Chip Processor, *Proc. of 16th Ann. Symp. on Computer Architecture*, pp. 46-53 (1989).
8) Yamaguchi, Y., Sakai, S. and Kodama, Y.: Synchronization Mechanisms of a Highly Parallel Dataflow Machine EM-4, *IEICE Trans.*, Vol. E 74, No. 1, pp. 204-213 (1991).
9) 児玉, 坂井, 山口：データ駆動計算機 EM-4 の関数分散方式, 情報処理学会研究報告, ARC-85-19, pp. 63-70 (1990).
10) 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).
11) Banerjee, D. U. and Gajski, D.D.: Fast Execution of Loop with IF Statements, *IEEE Trans. Comput.*, Vol. C-33, No. 11, pp. 1030-1033 (1984).

(平成 6 年 9 月 21 日受付)
(平成 7 年 3 月 13 日採録)



山名 早人（正会員）

昭和 39 年生. 昭和 62 年早稲田大学理工学部電子通信学科卒業. 平成元年同大学大学院修士課程修了. 平成 5 年同大学院博士後期課程修了. 平成元～5 年同大学情報科学研究教育センター助手. 工学博士. 平成 5 年情報処理学会第 46 回全国大会奨励賞受賞. 平成 5 年より電子技術総合研究所に勤務. コンピュータアーキテクチャ, 並列処理システムの研究に従事. 著書「超並列コンピュータ入門」(共著). 電子情報通信学会, IEEE 各会員.

佐藤 三久（正会員）

昭和 34 年生. 昭和 57 年東京大学理学部情報科学科卒業. 昭和 61 年同大学院理学系研究科博士課程中退. 同年新技術事業団後藤磁束量子情報プロジェクトに参加. 平成 3 年より, 通産省電子技術総合研究所勤務. 現在, 同所情報アーキテクチャ部計算機方式研究室主任研究官. 理学博士. 並列処理アーキテクチャ, 言語およびコンパイラ, 計算機性能評価技術等の研究に従事. 日本応用数理学会会員.

児玉 祐悦（正会員）

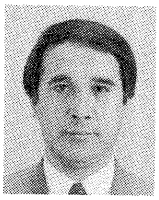
昭和 37 年生. 昭和 61 年東京大学工学部計数工学科卒業. 昭和 63 年同大学院工学系研究科情報工学専門課程修了. 同年通商産業省工業技術院電子技術総合研究所入所. 以来, データ駆動型計算機などの並列計算機システムの研究に従事. 特にプロセッサアーキテクチャ, 並列性制御, 動的負荷分散, 並列探索問題などに興味あり. 現在, 情報アーキテクチャ部計算機方式研究室に所属.

坂根 広史

昭和 41 年生. 平成 2 年山口大学工学部電子工学科卒業. 平成 4 年電気通信大学大学院博士前期課程電子工学専攻修了. 同年通商産業省工業技術院電子技術総合研究所入所. 情報アーキテクチャ部計算機方式研究室に所属. 以来, 並列計算機のアーキテクチャおよびその性能評価の研究に従事. 電子情報通信学会, 神経回路学会各会員.

坂井　修一（正会員）

昭和33年生．昭和56年東京大学理学部情報科学科卒業．昭和61年同大学院情報工学専門課程修了．工学博士．同年，電子技術総合研究所入所．平成3年4月より1年間米国MIT招聘研究員．平成5年3月よりRWC超並列アーキテクチャ研究室室長，現在に至る．計算機システム一般，特にアーキテクチャ，並列処理，スケジューリング問題などの研究に従事．情報処理学会研究賞（平成元年），同論文賞（平成2年度），元岡記念賞（平成3年），日本IBM科学賞（平成3年）各受賞．



山口　喜教（正会員）

昭和24年生．昭和47年東京大学工学部電子工学科卒業．同年通商産業省工業技術院電子技術総合研究所入所．以来，高級言語計算機，データフロー計算機などの研究に従事．現在，情報アーキテクチャ部計算機方式研究室長．工学博士．情報処理学会論文賞（平成2年度）受賞．著書「データ駆動型並列計算機」（共著）．電子情報通信学会会員．