(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第3601341号

(P3601341)

(45)発行日　平成16年12月15日(2004.12.15)　(24)登録日　平成16年10月1日(2004.10.1)

(51)Int.Cl.[7]	F I	
G06F 17/16	G06F 17/16	G
G06F 9/45	G06F 9/44	322H

請求項の数 5　(全 18 頁)

(21)出願番号　特願平11-31568
(22)出願日　平成11年2月9日(1999.2.9)
(65)公開番号　特開2000-231545(P2000-231545A)
(43)公開日　平成12年8月22日(2000.8.22)
審査請求日　平成14年12月25日(2002.12.25)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)復代理人　100102587
弁理士　渡邊　昌幸
(74)代理人　100077274
弁理士　磯村　雅俊
(72)発明者　廣岡　孝志
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内
(72)発明者　太田　寛
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内

最終頁に続く

(54)【発明の名称】並列プログラム生成方法

(57)【特許請求の範囲】

【請求項１】

逐次実行用プログラムを入力して構文解析を行い中間語を出力するステップと、該中間語を並列化変換するステップと、該中間語を入力して並列実行用プログラムを出力するステップとを含む分散共有メモリ型並列計算機用の並列化コンパイラにおいて、

上記中間語を並列化変換するステップは、

（ａ）該中間語を入力して、ループ繰り返し範囲を分散する並列化実施ループの中から最も逐次実行時間が大きいカーネルループを検出するステップと、

上記分散共有メモリ型並列計算機のデータ分散方法のうち、データが初めて参照されたときに該データを含むメモリ上の１連続領域を該データを参照したプロセッサの物理メモリに割り付けるファーストタッチ方式データ分散を、上記カーネルループと同じデータアクセスパターンを再現することにより制御するコードを生成するステップと、

該コードを実行文の先頭に挿入するステップと

を含むことを特徴とする並列プログラム生成方法。

【請求項２】

逐次実行用プログラムを入力して構文解析を行い中間語を出力するステップと、該中間語を並列化変換するステップと、該中間語を入力して並列実行用プログラムを出力するステップとを含む分散共有メモリ型並列計算機用の並列化コンパイラにおいて、

上記中間語を並列化変換するステップは、

（ｂ）該中間語を入力して、ループ繰り返し範囲を分散する並列化実施ループの中から最

も逐次実行時間が大きいカーネルループを検出するステップと、分散共有メモリ型並列計算機のデータ分散方法のうち、データが初めて参照されたときに該データを含むメモリ上の１連続領域を該データを参照したプロセッサの物理メモリに割り付けるファーストタッチ方式データ分散を、上記カーネルループのループ本体で参照されるカーネル配列のデータをカーネル配列と同じ配列形状を有するクローン配列に代入することにより制御するコードを生成し、該コードを該カーネルループの直前に挿入するステップと、該カーネルループ内のカーネル配列の参照をクローン配列の参照に置換するステップとを含むか、あるいは、

（ｃ）該中間語を入力して、プロファイル情報として入力プログラムを並列実行して得たメモリ上の連続領域毎の各プロセッサの参照回数を取得し、分散共有メモリ型並列計算機のデータ分散方法のうち、データが初めて参照されたときに該データを含むメモリ上の１連続領域を該データを参照したプロセッサの物理メモリに割り付けるファーストタッチ方式データ分散を、各メモリ上の連続領域を最も参照回数の多いプロセッサにアクセスさせることにより制御するコードを生成するステップと、該コードを実行文の先頭に挿入するステップとを含むことを特徴とする並列プログラム生成方法。

【請求項３】

逐次実行用プログラムを入力して構文解析を行い中間語を出力するステップと、該中間語を並列化変換するステップと、該中間語を入力して並列実行用プログラムを出力するステップとを含む分散共有メモリ型並列計算機用の並列化コンパイラにおいて、

上記中間語を並列化変換するステップは、

（ｄ）該中間語を入力して、上記コンパイラの静的解析情報から得たカーネルループの各プロセッサの参照範囲を取得し、分散共有メモリ型並列計算機のデータ分散方法のうち、データが初めて参照されたときに該データを含むメモリ上の１連続領域を該データを参照したプロセッサの物理メモリに割り付けるファーストタッチ方式データ分散を、各プロセッサに割り付けたいメモリ上の連続領域の配列要素を参照させることにより制御するコードを生成するステップと、該コードを実行文の先頭に挿入するステップとを含むか、あるいは、

（ｅ）該中間語を入力して、ユーザ指示情報から得たカーネルループの各プロセッサの参照範囲を取得し、分散共有メモリ型並列計算機のデータ分散方法のうち、データが初めて参照されたときに該データを含むメモリ上の１連続領域を該データを参照したプロセッサの物理メモリに割り付けるファーストタッチ方式データ分散を、各プロセッサに割り付けたいメモリ上の連続領域の配列要素を参照させることにより制御するコードを生成するステップと、該コードを実行文の先頭に挿入するステップとを含むことを特徴とする並列プログラム生成方法。

【請求項４】

逐次実行用プログラムを入力して構文解析を行い中間語を出力するステップと、該中間語を並列化変換するステップと、該中間語を入力して並列実行用プログラムを出力するステップとを含む分散共有メモリ型並列計算機用の並列化コンパイラにおいて、

上記中間語を並列化変換するステップは、

（ｆ）該中間語を入力して、プロファイル情報として入力プログラムを並列実行して得たメモリ上の連続領域毎の各プロセッサの参照回数を取得し、各メモリ上の連続領域の最も参照回数の多いプロセッサの情報テーブルを生成してオブジェクトコードに挿入することにより、オペレーティングシステムに連続領域番号とプロセッサ番号の組み合わせによるメモリ上の連続領域割り付け情報を与えるステップを含むか、あるいは、

（ｇ）該中間語を入力して、上記コンパイラの静的解析情報から得たカーネルループの各プロセッサの参照範囲を取得し、メモリ上の各連続領域を割り付けたいプロセッサの情報テーブルを生成してオブジェクトコードに挿入することにより、オペレーティングシステムに連続領域番号とプロセッサ番号の組み合わせによるページ割り付け情報を与えるステップを含むか、あるいは、

（ｈ）該中間語を入力して、ユーザ指示情報から得たカーネルループの各プロセッサの参

照範囲を取得し、メモリ上の各連続領域を割り付けたいプロセッサの情報テーブルを生成してオブジェクトコードに挿入することにより、オペレーティングシステムに連続領域番号とプロセッサ番号の組み合わせによるメモリ上の連続領域割り付け情報を与えるステップを含むことを特徴とする並列プログラム生成方法。

【請求項５】

請求項１，請求項２，請求項３又は請求項４に記載の各ステップをプログラムに変換した並列プログラム生成用コンパイラを格納したことを特徴とする記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ソースプログラムから並列化コンパイラを用いて並列プログラムを生成する方法に係わり、特にデータ分散によるデータローカリティ最適化を行うことが可能な並列プログラム生成方法およびそのプログラムを格納した記録媒体に関する。

【０００２】

【従来の技術】

分散共有メモリ型並列計算機の論理共有物理分散メモリを実現する方法の一つとして、論理共有の仮想メモリ空間をページと呼ばれる単位毎に切り分けて、物理的に分散されたメモリに割り付ける方法がある。このとき、どのページをどのプロセッサに割り付けるかを決定する方法として、以下の２つが知られている。その１つは、データが初めて参照された時にそのデータ含むページをそのデータを参照したプロセッサのメモリに割り付けるファーストタッチ方式と呼ばれるデータ分散方法である。もう１つは、明示的にデータ分散指示文によりデータ分散形状を指定するデータ分散方法である。

例えば、図９に示すような逐次実行用ソースプログラム１１が入力された場合、プロセッサ数を４台とし、ページサイズを配列要素５つ分としてファーストタッチ方式のデータ分散に従うと、配列Ａの各要素は初めて参照される手続きｉｎｉｔの初期化ループ（図９の２３行目から２５行目）により、図１０（ａ）に示すようにｐｅ０にＡ（１：２５）、ｐｅ１にＡ（２６：５０）、ｐｅ２にＡ（５１：７５）、ｐｅ３にＡ（７６：１００）が割り付けられる。なお、ｐｅ０はプロセッサ０を示し、ｐｅ１はプロセッサ１を示し、ｐｅ２はプロセッサ２を示し、ｐｅ３はプロセッサ３を示す。このように、従来のデータの分散状況は、１：１００ページサイズを各プロセッサｐｅ０～３に平等に分散している。

【０００３】

また、データ分散指示文”ｃ＄ｄｉｓｔｒｉｂｕｔｅ Ａ（ｂｌｏｃｋ）”をプログラムの宣言部（例えば、後述の図１１の４行目から７行目を、１：２５、２６：５０、５１：７５、７６：１００に指定した場合）に挿入すれば、図１０（ａ）と同様に各プロセッサｐｅ０～ｐｅ３に平等のデータ分散がされる。

上記のファーストタッチ方式によるデータ分散方法、及びデータ分散指示文によるデータ分散方法は、例えば、文献、Ｒｏｈｉｔ Ｃｈａｎｄｒａ，Ｄｉｎｇ－Ｋａｉ Ｃｈｅｎ，Ｒｏｂｅｒｔ Ｃｏｘ，Ｄｒｏｒ Ｅ．Ｍａｙｄａｎ，Ｎｅｄｅｌｊｋｏｖｉｃ，Ｊｅｎｎｉｆｅｒ Ｍ．Ａｎｄｅｒｓｏｎ著の”Ｄａｔａ Ｄｉｓｔｒｉｂｕｔｉｏｎ Ｓｕｐｐｏｒｔ ｏｎ Ｄｉｓｔｒｉｂｕｔｅｄ Ｓｈａｒｅｄ Ｍｅｍｏｒｙ Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒｓ”，ＳＩＧＰＬＡＮ’９７ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｐｒｏｇｒａｍｍｉｎｇ Ｌａｎｇｕａｇｅ Ｄｅｓｉｇｎ ａｎｄ Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ，（ＰＬＤＩ）Ｌａｓ Ｖｅｇａｓ，ＮＶ，ｊｕｎｅ １５－１８，１９９７，ｐｐ．３３４－３４５に記載されている。

【０００４】

【発明が解決しようとする課題】

上述のように、従来のファーストタッチ方式データ分散方法では、初期化ループでのデータアクセスパターンがカーネルループ（全プログラムの中で最も実行時間の長いループ）でのデータアクセスパターンと一致しない場合は、カーネルループでのデータローカリティを悪化させ、並列プログラムの処理速度向上を妨げる原因の一つとなっていた。例えば

、図１１（ａ）に示すようにプログラムが４つのプセッサｐｅ０～３に平等に分散されている場合、特に図９の３３行目から３５行目の１：６０のカーネルループのサブルーチンが１００００回繰り返し実行されるときには、これらが全て実行するプロセッサｐｅの場所に配置されていないと、遠い場所にアクセスして取り込まなければならないため、処理速度は遅くなる。

【０００５】

また、上記従来のデータ分散指示文によるデータ分散方法では、指示文で表現することが困難なデータ分散が存在し、最適なデータ分散を実現することができないため、データローカリティを悪化させ、並列プログラムの処理速度向上を妨げる原因の一つとなっていた。

例えば、図９に示すような逐次実行用ソースプログラム１１が入力された場合、プロセッサ数４台でファーストタッチ方式データ分散に従えば、配列Ａの各要素は初めて参照される手続きｉｎｉｔの初期化ループ（図９の２３行目から２５行目）により、図１０（ａ）に示すようにｐｅ０にＡ（１：２５）、ｐｅ１にＡ（２６：５０）、ｐｅ２にＡ（５１：７５）、ｐｅ３にＡ（７６：１００）が割り付けられる。ところが、手続きｋｅｒｎｅｌのカーネルループ（図９の３３行目から３５行目）の配列Ａの参照範囲は、図１０（ｂ）に示すようにｐｅ０がＡ（４１：５５）、ｐｅ１がＡ（５６：７０）、ｐｅ２がＡ（７１：８５）、ｐｅ３がＡ（８６：１００）となり、図１０（ｃ）に示すように配列Ａの（４１：７０）、及び（７６：８５）が他のプロセッサに割り付けられたデータの参照、すなわちリモート参照（Ｒ）になり、カーネルループ中の全参照の６６．７％がリモート参照（Ｒ）になってしまう。図１０（ｂ）の場合、自分のプロセッサに割り付けられたデータを参照するリモート参照（Ｌ）は極めて少なく、ＰＥ３の全部とＰＥ２の一部分のみである。

また、データ分散指示文では、図１０（ｂ）に示すようなデータ分散を指示することは困難である。

【０００６】

そこで、本発明の目的は、このような従来の課題を解決し、カーネルループでの最適なデータ分散が、従来のファーストタッチ方式データ分散や、データ分散指示文によるデータ分散で実現できない場合にも、カーネルループでの最適なデータ分散を実現し、それにより、データローカリティを向上させて並列プログラムの処理速度を高速化できる並列プログラム生成方法を提供することである。

【０００７】

【課題を解決するための手段】

上記目的を達成するため、本発明の並列プログラム生成方法では、並列化実施ループを検出し、カーネルループを検出し、次にファーストタッチ制御コードを生成し、そのファーストタッチ制御コードを実行文の先頭、又はカーネルループの直前に挿入した並列プログラムを生成する。これにより、図９に示すような逐次実行用ソースプログラム１１が入力された場合、図１０（ｄ）に示すようにプロセッサｐｅ０にＡ（１：２５）、及びＡ（４１：５５）、ｐｅ１にＡ（２６：４０）、及びＡ（５６：７０）、ｐｅ２にＡ（７１：８５）、ｐｅ３にＡ（８６：１００）が割り付けられる。これにより、カーネルループでのデータローカリティを向上させることができ、その結果、並列プログラムの処理速度を高速化できる。

また、本発明の並列プログラム生成方法では、プロファイル情報、又はコンパイラ静的解析情報、又はユーザ指示情報を取得し、ファーストタッチ制御コードを生成し、ファーストタッチ制御コードを実行文の先頭に挿入した並列プログラムを生成する。

さらに、本発明の並列プログラム生成方法では、プロファイル情報、又はコンパイラ静的解析情報、又はユーザ指示情報を取得し、ページ割り付け情報を生成し、ページ割り付け情報を挿入したオブジェクトプログラムを生成する。

【０００８】

【発明の実施の形態】

（用語と図面の対応関係）
最初に、以下の実施例で用いる用語、および図面との対応関係について説明する。
▲１▼並列化コンパイラ２とは、高級言語で記述された逐次実行用ソースプログラム１を入力し、並列実行用の並列プログラム３を出力するコンパイラを示す（図１参照）。
▲２▼プログラム先頭版ファーストタッチ制御方法とは、カーネルループと同じデータアクセスパターンを再現するループを、実行文の先頭に挿入してファーストタッチ方式データ分散を制御する方法を指す（図２参照）（第１の実施例）。
▲３▼ループ直前版ファーストタッチ制御方法とは、カーネルループと同じデータアクセスパターンを再現しながら、データ分散対象配列のデータをデータ分散対象配列と同じ配列形状を有するクローン配列にコピーするループを、カーネルループの直前に挿入することによりファーストタッチ方式データ分散を制御する方法を指す（図３参照）（第２の実施例）。
【０００９】
▲４▼プロファイル情報版ファーストタッチ制御方法とは、プロファイル情報を基に各ページの最も参照回数の多いプロセッサに参照させるループを、実行文の先頭に挿入してファーストタッチ方式データ分散を制御する方法を指す（図４参照）（第３の実施例）。なお、プロファイル情報とは、１回実行させて取得した種々の情報であり、どの点を参照したかを示す情報である。
▲５▼静的解析情報版ファーストタッチ制御方法とは、コンパイラの静的解析情報を基に各プロセッサに割り付けたいページの配列要素を参照させるループを、実行文の先頭に挿入してファーストタッチ方式データ分散を制御する方法を指す（図５参照）（第４の実施例）。なお、静的解析情報とは、コンパイラを実行させなくても判別できる解析情報である。
▲６▼ユーザ指示情報版ファーストタッチ制御方法とは、ユーザ指示情報を基に各プロセッサに割り付けたいページの配列要素を参照させるループを、実行文の先頭に挿入してファーストタッチ方式データ分散を制御する方法を指す（図５参照）（第５の実施例）。なお、ユーザ指示情報とは、コンパイラが自分で解析しなくても、ユーザから指示してもらう情報である。静的解析情報版ファーストタッチ制御方法とは、同じフローでよく、静的解析情報の代りにユーザ指示の配列参照範囲テーブル等を参照することにより行う。
【００１０】
▲７▼プロファイル情報版データ分散制御方法とは、プロファイル情報から得た各ページの最も参照回数の多いプロセッサの情報をオブジェクトコードに挿入してオペレーティングシステムに最適なデータ分散を実現させる方法を指す（図６参照）（第６の実施例）。この方法は、オブジェクトコードをプログラムコードの下欄に挿入することにより、ＯＳ（オペレーティングシステム）はそれに従って割り付ける。
▲８▼静的解析情報版データ分散制御方法とは、コンパイラの静的解析情報から得た各プロセッサに割り付けたいページの情報をオブジェクトコードに挿入してオペレーティングシステムに最適なデータ分散を実現させる方法を指す（図７参照）（第７の実施例）。この方法は、上記プロファイル情報の代りに静的解析情報を用いてオブジェクトコードに挿入する点のみが異なる。
▲９▼ユーザ指示情報版データ分散制御方法とは、ユーザ指示情報から得た各プロセッサに割り付けたいページの情報をオブジェクトコードに挿入してオペレーティングシステムに最適なデータ分散を実現させる方法を指す（図７参照）（第８の実施例）。この方法は、上記静的解析情報の代りにユーザ指示情報を用いてオブジェクトコードに挿入する点のみが異なる。
【００１１】
（実施例）
以下、本発明の実施例を、図面により詳細に説明する。
図１は、本発明の並列化コンパイラの一実施例を示す構成図である。
本発明は、計算機上で実施され、ソースプログラム１を入力として、並列プログラム３を

生成する並列化コンパイラの機能、すなわち並列プログラム生成方法である。
図１において、並列化コンパイラ２は、高級言語で記述された逐次実行用ソースプログラム１を入力し、並列実行用の並列プログラム３を生成・出力し、その処理の過程で中間語２５を生成する。なお、並列化コンパイラ２の出力は、以下の例ではソースプログラム形式で示すが、一般にはソースプログラム形式とは限らない。並列化コンパイラ２は、逐次ソースプログラム１を読み込み構文解析を行って中間語２５を生成する構文解析部２１と、中間語２５から逐次ソースプログラム１のカーネルループを検出し、ファーストタッチ制御コードを生成し、ファーストタッチ制御コードを挿入して中間語２５を複数個のプロセッサで並列に処理する構造を持った中間語２５に変換する並列化部２２と、変換された中間語２５から並列プログラム３を生成して出力するコード生成部２３から構成されている。

【００１２】
また、並列化部２２内には、入力した逐次ソースプログラム１のループ繰り返し範囲を分散する並列化実施ループの中から最も逐次実行時間が大きいカーネルループを検出するカーネルループ検出部４と、解析情報２４として入力プログラムを並列実行して得たページ毎の参照回数や配列参照範囲を取得する解析情報取得部５と、解析情報２４から各プロセッサへのデータ割り付け情報を生成するデータ割り付け情報生成部６とファーストタッチを制御するコードを生成するファーストタッチ制御コード生成部７と、ファーストタッチを制御するコードを挿入するファーストタッチ制御コード挿入部８とデータ割り付け情報を挿入するデータ割り付け情報挿入部９とを有している。

【００１３】
（第１の実施例）
図２は、本発明の第１の実施例を示すプログラム先頭版ファーストタッチ制御方法の動作フローチャートである。
並列化部２２が実施するプログラム先頭版ファーストタッチ制御方法を、図２に従って説明する。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を例に述べる。並列化部２２では、まず、処理ステップ４１で入力した逐次ソースプログラム１１のループ繰り返し範囲を分散する並列化実施ループを検出し、並列化実施ループの中から最も逐次実行時間が大きいカーネルループを検出する。なお、処理ステップ４１は、図１に示すカーネルループ検出部４で実行される。
図８は、図１におけるカーネルループ検出部の動作フローチャートであり、図１８は、カーネルループ検出の際に作成されるループテーブルの図である。
図８を用いて、図２に示す処理ステップ４１のカーネルループ検出処理を説明する。まず、処理ステップ４１１で図９の入力プログラム１１中の行番号２２に示すループ並列化指示文”ｃ＄ｐａｒａ”を検出し、行番号２３から行番号２５の並列化実施ループを検出する。次に、処理ステップ４１２では、図１８に示す第１番目のループテーブル８１２を生成する。１番目のループテーブル８１２には、テーブル番号９１１、ＮＥＸＴテーブル９１２、ループポインタ９１３、カーネルループフラグ９１４、ファーストタッチ制御方法番号９１５、逐次実行時間９１６の各情報が含まれる。本実施例では、第１番目のループテーブル８１２の各項目の値として、テーブル番号９１１に１、ＮＥＸＴテーブル９１２は未登録、ループポインタ９１３に２３（図９の２３行目参照）、カーネルループフラグ９１４にｆａｌｓｅ、ファーストタッチ制御方法番号９１５は未登録、逐次実行時間９１６は未登録（最初の動作）が設定される。
次に、処理ステップ４１３で当ループの逐次実行時間を検出する（図９の２３行目参照）。実行時間は、代入文を単位１として表し、ループ繰り返し回数が１００であることから１×１００＝１００とする。よって、第１番目のループテーブル８１２の逐次実行時間９１６に１００を設定する。次に、処理ステップ４１４で他のループが存在するか否かを判定し、他にループが存在するため、処理ステップ４１１に戻り、上記処理を繰り返す。処理ステップ４１１で入力プログラム１１中の行番号３２に示すループ並列化指示文”ｃ＄ｐａｒａ”を検出し、行番号３３から行番号３５の並列化実施ループを検出する。次に、

処理ステップ４１２で、図１８に示すような第２番目のループテーブル８１３を生成する。

【００１４】

本例では、第２番目のループテーブル８１３の各項目の値として、テーブル番号９１１に２、ＮＥＸＴテーブル９１２は未登録、ループポインタ９１３に３３、カーネルループフラグ９１４にｆａｌｓｅ、ファーストタッチ制御方法番号９１５は未登録、逐次実行時間９１６は未登録を設定し、第１番目ループテーブル８１２のＮＥＸＴテーブル９１２に２を設定する。次に、処理ステップ４１３で当ループの逐次実行時間を検出する。実行時間は、代入文を単位１として表し、ループ繰り返し回数が当手続きｋｅｒｎｅｌで６０、当手続きｋｅｒｎｅｌの呼び出し元手続きｍａｉｎにおける手続きｋｅｒｎｅｌの手続き呼び出し文を含むループの繰り返し回数が１００００であることから１×６０×１００００＝６０００００とする。よって、第２番目のループテーブル８１３の逐次実行時間９１６に６０００００を設定する。

次に、処理ステップ４１４で他のループが存在するか否かを判定し、他にループが存在しないため、処理ステップ４１５に進む。処理ステップ４１５では、逐次実行時間が最大のループテーブル８１３を検出し、第２番目のループテーブル８１３のカーネルループフラグ９１４にｔｒｕｅを設定し、入力プログラム１１中にプログラム先頭版ファーストタッチ制御方法を指示する指示文”ｃ＄ｆｔ１”が含まれるため、図１８の第２番目ループテーブル８１３のファーストタッチ制御方法番号９１５に１を設定し、本カーネルループ検出処理を終了する。

【００１５】

図１２は、第１の実施例で作成される出力プログラムのフォーマット図である。

再び図２に戻り、処理ステップ４２でカーネルループと同じネスト（同じループ構造）、及び同じループ長となるループ構造を有するループを生成する。処理ステップ４２では、図１２に示す出力プログラム３１の４３行目に示すようなループを生成する。次に、処理ステップ４３で出力プログラム３１の４４行目に示すようなデータ分散対象配列Ａが右辺に含まれる代入文を生成し、ループ本体に挿入する。この代入文は、ファーストタッチさせるためのダミーの参照点になる。なお、処理ステップ４２、及び処理ステップ４３は、図１のファーストタッチ制御コード生成部７で実施される。次に、処理ステップ４４では、生成した手続きｆｔ１の手続き呼び出し文を出力プログラム３１の１０行目に示すような実行文の先頭に挿入し、実引数はカーネル手続き呼び出しに合わせてＡ（４１）とする。ここで、４１は出力プログラム３１の４１行目（要素番号）を指している。　なお、処理ステップ４４は、図１のファーストタッチ制御コード挿入部８で実施される。以上で、第１の実施例であるプログラム先頭版ファーストタッチ制御方法の説明を終了する。

この実施例によれば、ファーストタッチ制御コードを挿入することにより、カーネルループと同構造のループによってデータがファーストタッチで割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【００１６】

（第２の実施例）

図３は、本発明の第２の実施例を示す並列プログラム生成方法の処理フローチャートであり、図１３は並列化コンパイラが生成した出力プログラムのフォーマット図である。

以下、図１の並列化部２２が実施するループ直前版ファーストタッチ制御方法について説明する。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を想定する。

並列化部２２では、まず、処理ステップ５１で入力した逐次ソースプログラム１１のループ繰り返し範囲を分散する並列化実施ループを検出し、並列化実施ループの中から最も逐次実行時間が大きいカーネルループを検出する。本例では、処理ステップ５１は前述の処理ステップ４１と同様の処理を行い、図１８に示すループテーブル８１２、及び８１３を生成する。なお、処理ステップ５１は、図１のカーネルループ検出部４で実施される。次に、処理ステップ５２では、カーネルループと同じネスト、及び同じループ長となるループ構造を有するループを２つ生成する。処理ステップ５２では、図１３に示す出力プログ

ラム３２の６３行目に示すようなループ、及び７３行目に示すようなループを生成する。次に、処理ステップ５３では、出力プログラム３２の６１行目、及び７１行目に示すようなデータ分散対象配列Ａと同じ配列形状を有するクローン配列ｃｌｏｎｅＡを生成する。次に、処理ステップ５４では、出力プログラム３２の６４行目に示すようなクローン配列ｃｌｏｎｅＡが左辺、データ分散対象配列Ａが右辺に含まれる代入文を生成し、１つ目のループ本体に挿入する（図１３の６４行目）。

【００１７】

次に、処理ステップ５５では、出力プログラム３２の７４行目に示すようなクローン配列ｃｌｏｎｅＡが右辺、データ分散対象配列Ａが左辺に含まれる代入文を生成し、２つ目のループ本体に挿入する（図１３の７４行目）。なお、処理ステップ５２から処理ステップ５５までは、図１のファーストタッチ制御コード生成部７で実施される。次に、処理ステップ５６では、生成した手続きｆｔ２１の手続き呼び出し文を、出力プログラム３２中の１１行目に示すようなカーネルループの直前に挿入する。次に、処理ステップ５７では、出力プログラム３２の１３行目に示すように、手続き呼び出しの実引数をクローン配列ｃｌｏｎｅＡに置換する。次に、処理ステップ５８では、生成した手続きｆｔ２２の手続き呼び出し文を出力プログラム３２中の１５行目に示すようなカーネルループの直後に挿入し、本ループ直前版ファーストタッチ制御方法を終了する。このように、第２の実施例では、クローン配列として別の１次的な配列を作り、そこに移してカーネループで行う作業を実施した後、元の配列に戻すことになる。

なお、処理ステップ５６から処理ステップ５８は、図１のファーストタッチ制御コード挿入部８で実施される。

本実施例によれば、ファーストタッチ制御コードを挿入することにより、カーネルループと同構造のループによってデータがファーストタッチで割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【００１８】

（第３の実施例）

図４は、本発明の第３の実施例を示す並列プログラム生成方法の処理フローチャートであり、図１４は、並列コンパイラが作成した出力プログラムのフォーマット図である。

以下、図１の並列化部２２が実施するプロファイル情報版ファーストタッチ制御方法の概要を説明する。プロファイル情報とは、１回実行させて取得した種々の情報のことである。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を想定する。並列化部２２では、まず、処理ステップ６１で、プロファイル情報として図１９に示す形式のページ毎の各プロセッサの参照回数を示す参照回数テーブル２４１を取得する。なお、参照回数テーブル２４１は、図１の解析情報２４に含まれており、処理ステップ６１は、図１の解析情報取得部５で実施される。例えば、入力プログラム１１を並列に実行する際のプロセッサ数を４とし、ページサイズを配列要素５つ分の４０Ｂｙｔｅとし、さらに、配列Ａ（１：１００）は、論理共有の仮想メモリアドレスの０番地から７９２番地に割り付けられているとする。ページサイズは４０Ｂｙｔｅであるため、各ページの先頭アドレスは０番地、４０番地、８０番地、・・・と４０おきになる。入力プログラム１１を並列実行した場合、カーネルループにおける先頭アドレス３２０のページの参照回数は、ｐｅ０で５００００回、ｐｅ１で０回、ｐｅ２で０回、ｐｅ３で０回となる。同様にして得られた各ページの参照回数が、参照回数テーブル２４１に登録されている。

【００１９】

なお、参照回数テーブル２４１は、ページアドレス８３１、ｐｅ０の参照回数８３２、ｐｅ１の参照回数８３３、ｐｅ２の参照回数８３４、ｐｅ３の参照回数８３５の各情報を含み、先頭アドレス３２０番地のページの各プロセッサの参照回数９３２、先頭アドレス３６０番地のページの各プロセッサの参照回数９３３のように各ページの参照回数が含まれる。次に、処理ステップ６２では、図１９の参照回数テーブルから図２０に示すような割り付けページアドレステーブル２４２を生成する。図２０の割り付けページアドレステーブル２４２は、ｐｅ０が最も参照回数の多かったページの先頭アドレス８２１、ｐｅ１が

最も参照回数の多かったページの先頭アドレス８２２、ｐｅ２が最も参照回数の多かったページの先頭アドレス８２３、ｐｅ３が最も参照回数の多かったページの先頭アドレス８２４のような情報を含む。本実施例では、図２０の８２１にページ先頭アドレス３２０、３６０、４００が登録され、図２０の８２２にページ先頭アドレス４４０、４８０、５２０が登録され、図２０の８２３にページ先頭アドレス５６０、６００、６４０が登録され、図２０の８２４にページ先頭アドレス６８０、７２０、７６０が登録される。

【００２０】

なお、処理ステップ６２は、データ割り付け情報生成部６で実施される。次に、処理ステップ６３では、図１４に示す出力プログラム３３の５２行目から５７行目に示すような２重ループを生成し、５２行目の外側ループのループ範囲をプロセッサ数４に合わせて０から３とし、５３行目の内側ループのループ長にページ数３を設定する。次に、処理ステップ６４では、出力プログラム３３の５４行目、５５行目に示す割り付けページアドレステーブル２４２（図２０）に登録されたページアドレスを参照する命令コードを生成し、ループ本体に挿入する。なお、処理ステップ６３から処理ステップ６４は、図１のファーストタッチ制御コード生成部７で実施される。次に、処理ステップ６５では、生成した手続きｆｔ３の手続き呼び出し文を出力プログラム３３中の３行目に示すような実行文の先頭に挿入し、プロファイル情報版ファーストタッチ制御方法を終了する。なお、処理ステップ６５は、ファーストタッチ制御コード挿入部８で実施される。

本実施例によれば、ファーストタッチ制御コードを挿入することにより、各ページが最も参照回数の多いプロセッサにファーストタッチで割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【００２１】

（第４の実施例）

図５は、本発明の第４の実施例を示す並列プログラム生成方法の処理フローチャートであり、図１５は、並列化コンパイラが作成した出力プログラムのフォーマット図である。

以下、図１の並列化部２２が実施する静的解析情報版ファーストタッチ制御方法の概要を説明する。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を想定する。並列化部２２では、まず、処理ステップ７１でコンパイラの静的解析情報として図２１に示す形式の各プロセッサの配列参照範囲を示す配列参照範囲テーブル２４３を取得する。なお、配列参照範囲テーブル２４３は、図１の解析情報２４に含まれており、処理ステップ７１は、図１の解析情報取得部５で実施される。例えば、入力プログラム１１を並列に実行する際のプロセッサ数が４の場合、配列参照範囲テーブル２４３には、各プロセッサの配列参照範囲として、ｐｅ０にＡ（４１：５５）、ｐｅ１にＡ（５６：７０）、ｐｅ２にＡ（７１：８５）、ｐｅ３にＡ（８６：１００）が登録されている。次に、処理ステップ７２では、図２１の配列参照範囲テーブルから図２２に示すような割り付け添字テーブル２４４を生成する。図２２の割り付け添字テーブル２４４は、ｐｅ０が参照するページの先頭添字８４１、ｐｅ１が参照するページの先頭添字８４２、ｐｅ２が参照するページの先頭添字８４３、ｐｅ３が参照するページの先頭添字８４４のような情報を含む。本実施例では、ページサイズを配列要素５つ分としているので、ｐｅ０が参照するページは、先頭添字が４１、４６、５１であるような３ページである。したがって、図２２の８４１にページ先頭添字４１、４６、５１が登録される。

【００２２】

同様にして、図２２の８４２にページ先頭添字５６、６１、６６が登録され、図２２の８４３にページ先頭添字７１、７６、８１が登録され、図２２の８４４にページ先頭添字８６、９１、９６が登録される。なお、処理ステップ７２は、図１のデータ割り付け情報生成部６で実施する。次に、処理ステップ７３では、図１５に示す出力プログラム３４の３２行目から３６行目に示すような２重ループを生成し、３２行目の外側ループのループ範囲をプロセッサ数４に合わせて０から３とし、３３行目の内側ループのループ長にページ数３を設定する。次に、処理ステップ７４では、出力プログラム３４の３４行目に示す割り付け添字テーブル２４４に登録された添字を参照する命令コードを生成し、ループ本体

に挿入する。なお、処理ステップ７３から処理ステップ７４は、図１のファーストタッチ制御コード生成部７で実施される。次に、処理ステップ７５では、生成した手続きｆｔ４の手続き呼び出し文を出力プログラム３４中の１０行目に示すような実行文の先頭に挿入し、静的解析情報版ファーストタッチ制御方法を終了する。なお、処理ステップ７５は、図１のファーストタッチ制御コード挿入部８で実施される。

本実施例によれば、ファーストタッチ制御コードを挿入することにより、各プロセッサにカーネルループで各プロセッサが参照するページをファーストタッチで割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【００２３】

（第５の実施例）

次に、本発明の第５の実施例の並列プログラム生成方法を説明する。

第４の実施例と同じ図５を用いて、並列化部２２が実施するユーザ指示情報版ファーストタッチ制御方法の概要を説明する。ここでは、図１１に示す逐次ソースプログラム１２が入力された場合を想定する。第５の実施例が第４の実施例と異なるのは、第４の実施例がコンパイラが自分で解析していたのに対して、第５の実施例では、ユーザから教えてもらうことによりコンパイラの解析を不要にしている点である。

並列化部２２では、まず、処理ステップ７１の代わりにユーザ指示情報として図２１に示す形式の各プロセッサの配列参照範囲を示す配列参照範囲テーブル２４３を取得する。なお、配列参照範囲テーブル２４３は、図１の解析情報２４に含まれており、本処理ステップは、図１の解析情報取得部５で実施される。本実施例の場合、入力プログラム１２（図１１）の４行目から７行目までのユーザ指示文により、配列参照範囲テーブル２４３には、各プロセッサの配列参照範囲として、ｐｅ０にＡ（４１：５５）、ｐｅ１にＡ（５６：７０）、ｐｅ２にＡ（７１：８５）、ｐｅ３にＡ（８６：１００）が登録されている。次に、処理ステップ７２から処理ステップ７５を実施し、ユーザ指示情報版ファーストタッチ制御方法を終了する。なお、処理ステップ７２は、図１のデータ割り付け情報生成部６で実施され、処理ステップ７３から処理ステップ７４は、図１のファーストタッチ制御コード生成部７で実施され、処理ステップ７５は、図１のファーストタッチ制御コード挿入部８で実施される。

本実施例によれば、ファーストタッチ制御コードを挿入することにより、各プロセッサにカーネルループで各プロセッサが参照するページをファーストタッチで割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。

【００２４】

（第６の実施例）

図６は、本発明の第６の実施例を示す並列プログラム生成方法の処理フローチャートであり、図１６は、第６の実施例で作成されるオブジェクトコードに付加されたテーブル例の図である。

以下、図１の並列化部２２が実施するプロファイル情報版データ分散制御方法の概要を説明する。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を想定する。

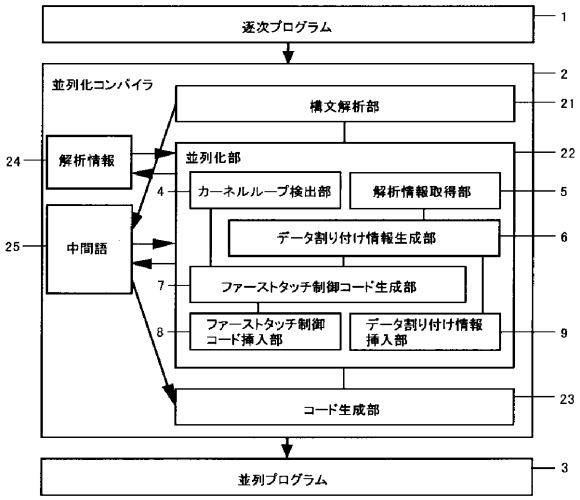
並列化部２２では、まず、処理ステップ８１で処理ステップ６１（図４）と同様にプロファイル情報として図１９に示す形式のページ毎の各プロセッサの参照回数を示す参照回数テーブル２４１を取得する。なお、参照回数テーブル２４１は、図１の解析情報２４に含まれており、処理ステップ８１は、図１の解析情報取得部５で実施される。次に、処理ステップ８２では、処理ステップ６２（図４）と同様に図２０に示すような割り付けページアドレステーブル２４２を生成する。なお、処理ステップ８２は、データ割り付け情報生成部６で実施される。次に、処理ステップ８３では、図１６に示すような形式で出力オブジェクトコード３５に割り付けページアドレステーブル２４２の情報を挿入し、プロファイル情報版データ分散制御方法を終了する。ここで、オブジェクトコード内のａｌｌｏｃ＿ｈｉｎｔの部分は、オペレーティングシステムがプログラムを実行開始するときに、ページ割り付けのためのヒントとして用いられる。例えば、１行目の０：３２０、３６０、４００；は、ｐｅ０にアドレス３２０、３６０、４００で指定されるページを割り付ける

べきであることを意味する。なお、処理ステップ８３は、図１のデータ割り付け情報挿入部９で実施される。
本実施例によれば、オペレーティングシステムにより、各ページが最も参照回数の多いプロセッサに割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。
【００２５】
（第７の実施例）
図７は、本発明の第７の実施例を示す並列プログラム生成方法の処理フローチャートであり、図１７は、第７の実施例で作成されたオブジェクトコードの後に挿入される付加コードの図である。
以下、図１の並列化部２２が実施する静的解析情報版データ分散制御方法の概要を説明する。ここでは、図９に示す逐次ソースプログラム１１が入力された場合を想定する。並列化部２２では、まず、処理ステップ９１で処理ステップ７１（図５）と同様にコンパイラの静的解析情報として図２１に示す形式の各プロセッサの配列参照範囲を示す配列参照範囲テーブル２４３を取得する。なお、配列参照範囲テーブル２４３は、図１の解析情報２４に含まれており、処理ステップ９１は、図１の解析情報取得部５で実施される。次に、処理ステップ９２では、処理ステップ７２（図５）と同様に図２２に示すような割り付け添字テーブル２４４を生成する。なお、処理ステップ９２は、データ割り付け情報生成部６で実施される。次に、処理ステップ９３では、図１７に示すような形式で出力オブジェクトコード３６に割り付け添字テーブル２４４の情報を挿入し、静的解析情報版データ分散制御方法を終了する。ここで、オブジェクトコード内の＄Ａ＋４１＊８は、配列Ａの先頭アドレス＄Ａから４１＊８＝３２８バイト上位のアドレスを表す。４１は配列添字、８は１要素の長さである。なお、処理ステップ９３は、図１のデータ割り付け情報挿入部９で実施される。
本実施例によれば、オペレーティングシステムにより、各プロセッサにカーネルループで各プロセッサが参照するページを割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。
【００２６】
（第８の実施例）
次に、本発明の第８の実施例の並列プログラム生成方法を説明する。
第７の実施例と同じ図７を用いて、並列化部２２が実施するユーザ指示情報版データ分散制御方法の概要を説明する。なお、第７の実施例と異なる点は、静的に解析して情報を取得することなく、ユーザからの指示により情報を設定すればよいことである。ここでは、図１１に示す逐次ソースプログラム１２が入力された場合を想定する。並列化部２２では、まず、処理ステップ９１に代わってユーザ指示情報として図２１に示す形式の各プロセッサの配列参照範囲を示す配列参照範囲テーブル２４３を取得する。なお、配列参照範囲テーブル２４３は、図１の解析情報２４に含まれており、本処理ステップは、図１の解析情報取得部５で実施される。次に、処理ステップ９２から処理ステップ９３を実施し、ユーザ指示情報版データ分散制御方法を終了する。なお、処理ステップ９２は、図１のデータ割り付け情報生成部６で実施され、処理ステップ９３は、図１のデータ割り付け情報挿入部９で実施される。図１７に示すように、オブジェクトコードの後にテーブルを与えることにより、オペレーティングシステムがそれに従って割り付けてくれる。
本実施例によれば、オペレーティングシステムにより、各プロセッサにカーネルループで各プロセッサが参照するページを割り付けられるため、データローカリティが向上し、並列プログラムの処理速度が高速化される。
【００２７】
以上、第１～第８実施例を説明したが、これらの各ステップをプログラムに変換したものが、それぞれ並列プログラム生成用コンパイラである。このコンパイラをそれぞれＣＤ－ＲＯＭや磁気ディスク等の記録媒体に格納することにより、その記録媒体を任意の場所に設置されたコンピュータにインストールして実行させれば、本発明を実現することができ

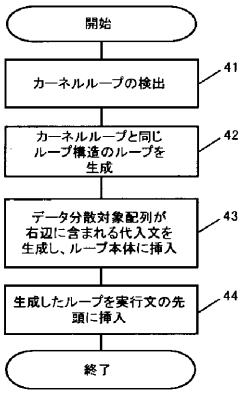
る。
【００２８】
【発明の効果】
以上説明したように、本発明によれば、カーネルループでの最適データ分散を実現するファーストタッチ制御コードを挿入した最適並列プログラムを生成することができるので、データのローカリティが向上し、並列プログラムの処理速度が高速化できる。
【図面の簡単な説明】
【図１】本発明の一実施例を示す並列化コンパイラのブロック図である。
【図２】本発明の第１の実施例を示すプログラム先頭版ファーストタッチ制御方法の処理フロチャートである。
【図３】本発明の第２の実施例を示すループ直前版ファーストタッチ制御方法の処理フロチャートである。
【図４】本発明の第３の実施例を示すプロファイル情報版ファーストタッチ制御方法の処理フロチャートである。
【図５】本発明の第４および第５の実施例を示す静的解析情報版およびユーザ指示情報版ファーストタッチ制御方法の処理フロチャートである。
【図６】本発明の第６の実施例を示すプロファイル情報版データ分散制御方法の処理フロチャートである。
【図７】本発明の第７および第８の実施例を示す静的解析情報版およびユーザ指示情報版データ分散制御方法の処理フロチャートである。
【図８】本発明におけるカーネルループを検出する処理のフロチャートである。
【図９】本発明の入力となる逐次実行用ソースプログラムの説明図（第６の実施例以外の実施例）である。
【図１０】本発明および従来における配列Ａのデータ分散状況、及び参照範囲の説明図である。
【図１１】本発明の入力となる逐次実行用ソースプログラムの説明図（第６の実施例）である。
【図１２】本発明の出力となる並列実行用ソースプログラムの説明図（第１の実施例）である。
【図１３】本発明の出力となる並列実行用ソースプログラムの説明図（第２の実施例）である。
【図１４】本発明の出力となる並列実行用ソースプログラムの説明図（第３の実施例）である。
【図１５】本発明の出力となる並列実行用ソースプログラムの説明図（第４の実施例）である。
【図１６】本発明の出力となるオブジェクトプログラムの説明図（第７の実施例）である。
【図１７】本発明の出力となるオブジェクトプログラムの説明図（第８の実施例）である。
【図１８】本発明で用いられるループテーブルの説明図である。
【図１９】本発明で用いられるページ毎参照回数テーブルの説明図である。
【図２０】本発明で用いられる割り付けページアドレステーブルの説明図である。
【図２１】本発明で用いられる配列参照範囲テーブルの説明図である。
【図２２】本発明で用いられる割り付け添字テーブルの説明図である。
【符号の説明】
１…逐次プログラム、２…並列化コンパイラ、３…並列プログラム、
４…カーネルループ検出部、５…解析情報取得部、
６…データ割り付け情報生成部、７…ファーストタッチ制御コード生成部、
８…ファーストタッチ制御コード挿入部、９…データ割り付け情報挿入部、
２１…構文解析部、２２…並列化部、２３…コード生成部、２４…解析情報、

２５…中間語、１１，１２…入力プログラム、３１～３４…出力プログラム、
３５，３６…オブジェクトコード、８１２，８１３…ループテーブル、
９１１…テーブル番号、９１２…ＮＥＸＴテーブル、
９１３…ループポインタ、９１４…カーネルループフラグ、
９１５…ＦＩ制御方法番号、９１６…逐次実行時間、
２４１…ページ毎参照回数テーブル、
２４２…割り付けページアドレステーブル、２４３…配列参照範囲テーブル、
２４４…割り付け添字テーブル。

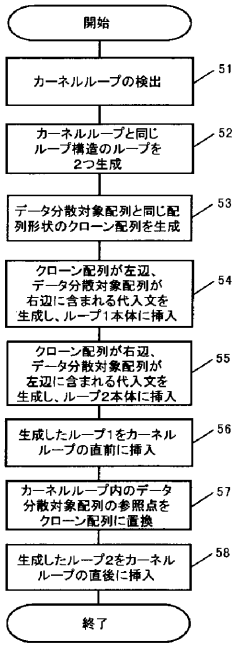
【図１】



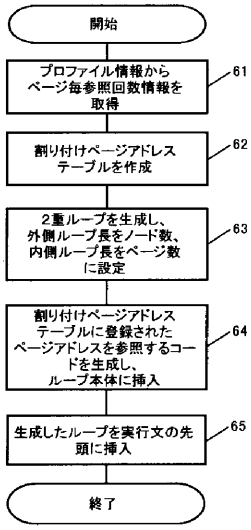
【図２】



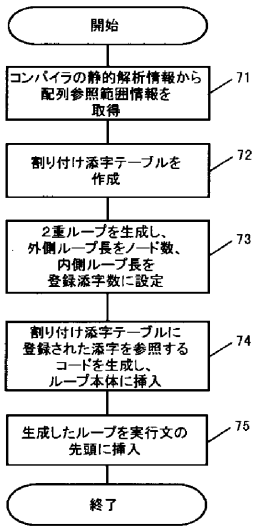
【図３】



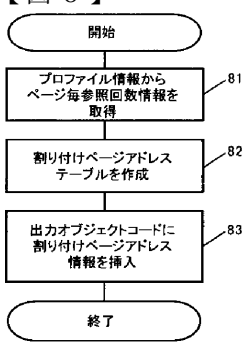
【図４】



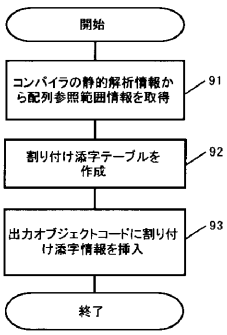
【図５】



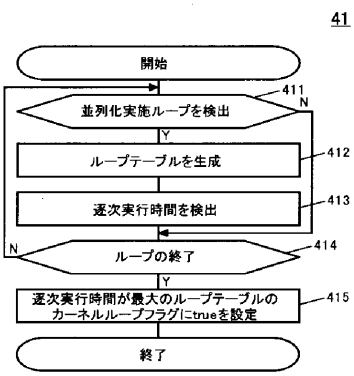
【図６】



【図７】



【図８】



【図９】

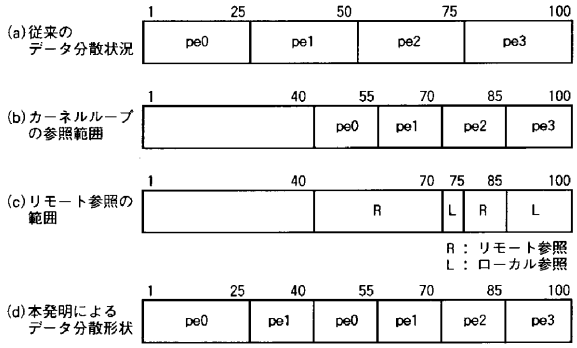
11

```
 1: Program main
 2: real A(100)
 3: c$ft1
    .....
10: call init(A)
11: do itr=1, 10000
12:   call kernel(A(41))
13: enddo
14: end

20: subroutine init(A)
21: real A(100)
22: c$para
23: do i=1, 100
24:   A(i)=0
25: enddo
26: return

30: subroutine kernel(A)
31: real A(60)
32: c$para
33: do i=1, 60
34:   A(i)= ...
35: enddo
36: return
```

【図１０】



【図１１】

12

```
 1: Program main
 2: real A(100)
 3: c$ft5
 4: c$dist5(A(41:55),0)
 5: c$dist5(A(56:70),1)
 6: c$dist5(A(71:75),2)
 7: c$dist5(A(76:100),3)
    .....
10: call init(A)
11: do itr=1, 10000
12:   call kernel(A(41))
13: enddo
14: end

20: subroutine init(A)
21: real A(100)
22: c$para
23: do i=1, 100
24:   A(i)=0
25: enddo
26: return

30: subroutine kernel(A)
31: real A(60)
32: c$para
33: do i=1, 60
34:   A(i)= ...
35: enddo
36: return
```

【図１２】

31

```
 1: Program main
 2: real A(100)
    .....
10: call ft1(A(41))
11: call init(A)
12: do itr=1, 10000
13:   call kernel(A(41))
14: enddo
15: end

40: subroutine ft1(A)
41: real A(60)
42: c$para
43: do i=1, 60
44:   =A(i)
45: enddo
46: return
```

【図１３】

32

```
 1: Program main
 2: real A(100), cloneA(60)
    .....
10: call init(A)
11: call ft21(A(41),cloneA)
12: do itr=1, 10000
13:   call kernel(cloneA)
14: enddo
15: call ft22(A(41),cloneA)
16: end

60: subroutine ft21(A, cloneA)
61: real A(60), cloneA(60)
62: c$para
63: do i=1, 60
64:   cloneA(i)=A(i)
65: enddo
66: return

70: subroutine ft22(A, cloneA)
71: real A(60), cloneA(60)
72: c$para
73: do i=1, 60
74:   A(i)=cloneA(i)
75: enddo
76: return
```

【図１４】

33

```
 1: Program main
 2: real A(100)
 3: call ft3
     .....
10: call init(A)
11: do itr=1, 10000
12:   call kernel(A(41))
13: enddo
14: end

 (実行時ライブラリ)
50: subroutine ft3
51: c$para
52: do node=0, 3
53:   do page=1, 3
54:     load r1, table[node][page]
55:     load r2, MEM(r1)
56:   enddo
57: enddo
58: return
```

【図１５】

34

```
 1: Program main
 2: real A(100)
     .....
10: call ft4
11: call init(A)
12: do itr=1, 10000
13:   call kernel(A(41))
14: enddo
15: end

30: subroutine ft4
31: c$para
32: do node=0, 3
33:   do ix=1, 3
34:     =A(table(node,ix))
35:   enddo
36: enddo
37: return
```

【図１６】

35

オブジェクトコード

alloc-hint:
0:320,360,400;
1:440,480,500;
2:540,580,620;
3:660,700,740;

【図１７】

36

オブジェクトコード

alloc-hint:
0:A$+41*8,A$+46*8,A$+51*8;
1:A$+56*8,A$+61*8,A$+66*8;
2:A$+71*8,A$+76*8,A$+81*8;
3:A$+86*8,A$+91*8,A$+96*8;

【図１８】

811	812	813	
テーブル番号	1	2	911
NEXTテーブル	2	-	912
ループポインタ	23	33	913
カーネルループフラグ	false	true	914
FT制御方法番号	-	1	915
逐次実行時間	100	600000	916

FT:ファーストタッチ

【図１９】

241

831 ページアドレス	832 pe0	833 pe1	834 pe2	835 pe3	931
:	:	:	:	:	
320	50000	0	0	0	932
360	50000	0	0	0	933
400	50000	0	0	0	934
440	0	50000	0	0	935
480	0	50000	0	0	936
520	0	50000	0	0	937
560	0	0	50000	0	938
:	:	:	:	:	

【図２０】

242

821 pe0	822 pe1	823 pe2	824 pe3	
320	440	560	680	921
360	480	600	720	922
400	520	640	760	923
				924

【図２１】

243

851 pe0	852 pe1	853 pe2	854 pe3	
A(41:55)	A(56:70)	A(71:85)	A(86:100)	951
				952

【図２２】

244

841 pe0	842 pe1	843 pe2	844 pe3
41	56	71	86
46	61	76	91
51	66	81	96

941
942
943
944

フロントページの続き

(72)発明者　飯塚　孝好
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内
(72)発明者　菊池　純男
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内

審査官　後藤　和茂

(56)参考文献　特開平１０－１４３３８２（ＪＰ，Ａ）
特開平９－３１９６５３（ＪＰ，Ａ）
鬼頭　宏幸他，プログラミング環境ＥＵＬＡＳＨにおけるコンパイラの実装，情報処理学会研究会報告，日本，社団法人情報処理学会，１９９６年１２月１２日，Ｖｏｌ．９６　Ｎｏ．１２２，Ｐ．１３～１８

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 15/16
G06F　9/45