類似事例の修正による並列プログラミング

2E－4

山崎 勝弘　松田 浩一　安藤 彰一
立命館大学理工学部

1 はじめに

類似した並列プログラムの構造を極力再利用して並列プログラミングの負担を軽減させる方法について述べる。並列アルゴリズムを四つのクラスに分類し、各クラス毎に並列プログラムを作成して、並列プログラミング用事例ベースを作成する。新たな問題に対して、類似した事例を事例ベースから検索し、それを修正して並列プログラムを生成する。

2 事例ベース並列プログラミング

並列プログラミングでは、複数プロセッサ間の負荷均衡を図り、かつプロセッサ間の情報授受のオーバーヘッドを極力減少させることが最も重要である。また、タスク分割や共有変数の相互排除と共に、そのマシン独自の並列化手法を熟知している必要がある。

本手法では並列プログラムの骨格をスケレトンとして用意する。スケレトンには、タスク分割、同期、相互排除、並列化手法、スレッド使用法など並列プログラムの最も重要な部分が含まれる。また、類似した事例を検索するために、その問題の特徴をインデックス付けする。さらに、プロセッサ数を変化させたときの速度向上を実測し、並列効果とする。インデックス、スケレトン、プログラム、並列効果、及び履歴を一つの事例として、事例ベースに格納する [1]。

図 1にシステム構成を示す。事例ベースには代表的な並列プログラムから作成した事例、及び各並列アルゴリズムクラス毎の実行の流れを示す並列構造が含まれる。問題解析で、新たな問題の解析と特徴づけを各種観点から行い、インデックスを作成する。これを用いて、事例ベースから類似事例を検索し、それを修正して目的とする並列プログラムを作成する。

本研究では仮想共有メモリ並列マシン KSR 1上での Fortran/C プログラミングを仮定する。仮想共有メモリ方式では、単一の仮想アドレス空間が与えられるので、分散メモリプログラミングの困難さを軽減させることができる。

本システムでは、類似事例の自動検索を可能とするために、並列アルゴリズムを分割統治法、プロセッサファーム、プロセスネットワーク、繰り返し変換の四つに分類する。それらは KSR 1上で、パラレルリージョン、スレッド、バリア同期などにより実現される。パラレルリージョンとは複数のスレッドが同一コードを実行する部分であり、マスタースレッドからそれを起動する。各スレッド毎に一つのプロセッサが割り当てられ、並列実行される。また、アルゴリズムの特徴付けを行うために、バーゼル大学で開発された BACS (Basel Algorithm Classification Scheme) を用いる。

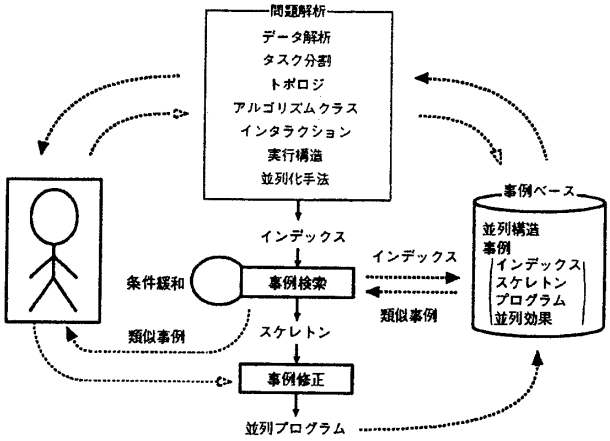

図 1: システム構成

3 類似事例の検索・修正法

3.1 類似事例の検索法

問題解析では、まず応用分野、仕様を規定し、データ依存性の有無、アルゴリズムの終了条件を明らかにする。これらを基に、並列アルゴリズムのクラスを決める [2]。例えば、データ依存、終了条件が共に無ければ、プロセッサファームを適用できる。データ依存はあるが、終了条件が無ければ、プロセスネットワークを使用できる。各クラス毎に代表的な並列構造が事例ベースに格納されているので、決定された並列アルゴリズムクラスに対する並列構造をユーザに提示する。ユーザは同期、並列化手法を基に、使用する並列構造を選択する。その構造に対する BACS 表現がシステムから与えられる。システムは BACS 表現、並列化手法、インタラクション、データ構造、タスク分割などをインデックスとして、最も類似した事例を検索する。完全にマッチする類似事例が検索できないときは、検索条件を緩めて、とにかく類似した事例を探す。

3.2 類似事例の修正法

スケレトン内には、タスク分割、スレッドの使用法、同期などと共に、変数初期化、計算、結果の回収などの一連のプログラミング過程が記述されている。スレッ

Case-Based Parallel Programming,
Katsuhiro Yamazaki, Koichi Matsuda and Shoichi Ando, Ritsumeikan University.

表1 ハフ変換と KMP 法のインデックス

	ハフ変換	KMP 法
応用分野	イメージ処理	文字列照合
仕様	メッシュ分割された各格子点で直線数を計数	TEXT 内で複数 PAT と照合し、PAT の位置を検出
アルゴリズム	プロセッサファーム	プロセッサファーム
終了条件	なし	なし
ソースデータ	$x-y$平面上の点の座標値のファイル *points*	文字列1次元配列 $TEXT$、文字列2次元配列 PAT
結果データ	整数型3次元配列 *data*[スレッド数][Y][X]	構造体2次元配列 *searchbox*[スレッド数][]
タスク分割	*points* , ブロック *data*[スレッド数][Y][X] , コピー	*TEXT* , ブロック *PAT* , コピー *searchbox*[スレッド数][] , コピー
トポロジ	マスター・ワーカー	マスター・ワーカー
インタラクション	なし	なし
並列化手法	パラレルリージョン	パラレルリージョン
BACS	C^t	C^t

ドの起動と終了、スレッド間の同期などスレッド関連はスケレトン内の情報を再利用できる。従って、変数初期化、タスク分割、計算、結果回収などを主に修正する必要があり、これは通常ユーザにより手動で行われる。逐次プログラムが作成されている場合には、計算部分に逐次プログラムの計算部分を埋め込むことができる。検索事例が与えられた問題と酷似している場合には、自動修正を試みる。

4 事例の修正による並列ハフ変換

4.1 問題の定義

画像から直線を抽出する。$x-y$ 平面上の直線 $y=ax+b$ 上の複数の点は、$a-b$ 平面上では一点で交わる。すなわち、$a-b$ 平面をメッシュに分割し、各格子を通過する直線の数を求め、その最大値をもつ格子が a、b の値となる。

4.2 問題の解析

ソースデータは $x-y$ 平面上の点の座標値のファイル (*points*)、結果データは各格子点の直線の数 (*data*) である。*points* を複数のブロックに分割し、各スレッドが一つのブロックを処理する。*data* には複数スレッドから同時に書き込まれる可能性があるので、スレッド数分用意し、最後に全部を加える。各プロセッサ上での計算は全く独立に行え、かつ結果データはアルゴリズムの終了条件に作用しないので、プロセッサファームが使用できる。各スレッド間の同期は不要であるので、パラレルリージョンが使用でき、実行構造は C^t (計算がトータル、すなわち、全スレッドで同一コードを実行) である。以上のことからハフ変換のインデックスが作成され、それに最も類似した事例として、KMP 法が検索される。ハフ変換と KMP 法のインデックスを表1に示す。KMP 法のスケレトンを図2に示す。main ではスレッド数を入力し、チーム ID を得る。それをもとにパラレルリージョンを起動する。各スレッドは work を実行する。work ではスレッド番号を取得し、各 PAT 毎に、自分が処理すべき TEXT のブロック内で、照合を行う。

```
main
{ input the number of threads;
   get a team ID;
   activate a parallel region( team ID, work );
   print the results;
}
work
{ get a thread no.;
   len = length of TEXT for each thread                    ↑
     for ( the no. of PATs )                               a
     { for ( k = 0; k <= len + length of PAT -1; k++ )     ↓
          { CALCULATION;
            store the results into serachbox;
} } }
```

図2: KMP 法のスケレトン

4.3 事例の修正

スケレトン内の太字はスレッドを同時実行させるパラレルリージョンに関連する部分で再利用できる。CALCULATION は各問題の単位計算であり、仕様で記述された処理を行う。逐次プログラムがあれば、それを埋め込む。タスク分割 (a の部分)、変数初期化、結果出力をハフ変換用に修正する。

5 おわりに

文字列照合 (KMP 法) のプログラムをハフ変換プログラムに修正する例を示した。スレッド生成など最も重要な部分は再利用でき、タスク分割はスケレトンを参考に修正することを示した。現在、細線化、スプライン変換などに対して、同様の実験を進めている。

参考文献

[1] 山崎、松田、安藤 "並列プログラミング用事例ベースの構築"、信学技報、CPSY96-39、pp.1-6、1996。

[2] 松田、安藤、山崎 "並列プログラミング用事例ベースの作成と類似事例の検索・修正法"、情処学 52 回全大、3L-4、1996。