事例修正による並列プログラミングと並列実行の可視化

朝倉 啓之　安藤 彰一　山崎 勝弘
立命館大学理工学部

1　はじめに

本稿では並列プログラムを事例として蓄積し，それを再利用することにより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．また，並列実行を可視化することにより，並列性の高いプログラミングの実現を支援する．文字列照合(KMP法)から3次スプライン，さらにエッジ抽出の並列プログラミングを行う．

2　並列プログラミング支援

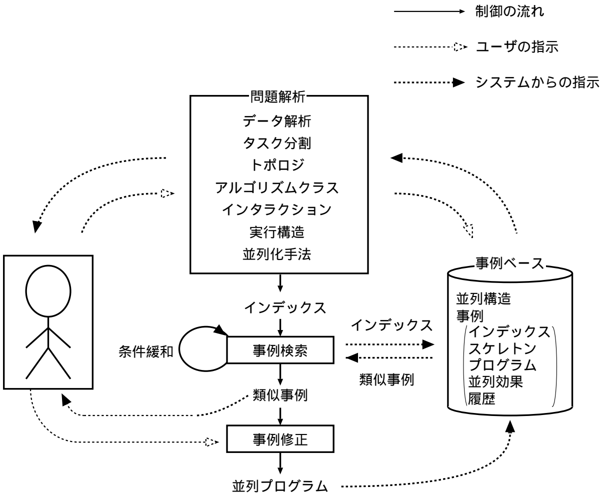


図 1: システム構成

図1にシステム構成を示す．事例にはインデックス，スケルトン，プログラム，並列効果，及び履歴がある[1]．ユーザが問題解析を行って並列プログラムの特徴を記すインデックスを作成する．このインデックスを用いて最も類似した事例を事例ベースから検索する．次に，タスク分割，同期などの並列プログラムの重要な骨格が記述されているスケルトンに，単位計算部分などをユーザが肉付け，修正を行って，プログラムを完成させる[2]．

3　3次スプライン

3.1　問題の定義

図2のように，複数の座標点(● 印)の各区間で滑らかな曲線となるような座標点(× 印)を求める．

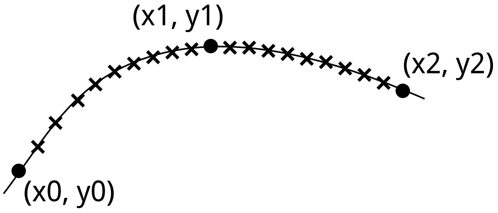


図 2: 3次スプライン

3.2　問題解析

ソースデータ，結果データを実数型構造体の1次元配列とする．それらをスレッド数分のブロックに分割し，ブロック毎にスレッドに割り当てる．各スレッドでは，補間を行う各区間で3次パラメータ表現した座標点の1・2次微係数が一致し，かつ座標点を通ることを用いて，各座標点の3次パラメータ式を求める．スレッド間の同期，終了条件は無いのでアルゴリズムはプロセッサファームを使用できる．並列化手法はパラレルリージョン，全スレッドで同じ計算を行うのでBACS表現はC^t(全スレッドでの同一計算)となる．表1に3次スプラインのインデックスを示す．類似事例として文字列照合(KMP法)が検索される．

表 1: 3次スプラインのインデックス

応用	数値計算
仕様	複数の座標点の補間を行う.
ソースデータ	実数型構造体1次元配列point
結果データ	実数型構造体1次元配列result
タスク分割	point, 要素ブロック, ブロック result, 要素ブロック, ブロック
終了条件	なし
トポロジ	マスター&ワーカー
アルゴリズム	プロセッサファーム
並列化手法	パラレルリージョン
インタラクション	なし
BACS	C^t

3.3　事例修正による並列プログラミング

KMP法の事例を修正して3次スプラインの並列プログラムを作成する．図3にKMP法のスケルトンを示す．mainからパラレルリージョンを起動して，全スレッドでworkを実行する．スレッド，同期はスケルトンをそのまま再利用できる．単位計算は逐次プログラムを殆ど再利用し，一部修正している．タスク分割ではKMP法の重複ブロック

Case-Based Parallel Programming and Parallel Performance Visualization
Hiroyuki Asakura, Shoichi Ando and Katsuhiro Yamazaki
Department of Computer Science, Ritsumeikan Univ.

をブロックに修正し，新規作成も必要とした．表2に事例修正による並列プログラミングの評価を示す．

```
main{
    スレッド数の入力;
    チームID=チーム作成(スレッド数);
    パラレルリージョン起動(チームID,work);
    結果出力;
}
work{
    THREADno=スレッド番号の取得;
    TEXTのタスク分割;
    for(PAT数){
        for(k=0; k≤ (分割TEXT長); k++){
            単位計算;
} } }
```

図3: KMP法のスケルトン

表2: 3次スプラインの事例修正による評価

(行数)

	スレッド	同期	タスク分割	単位計算
再利用	8	1	0	107
修正	0	0	11	3
新規	0	0	12	0

4　エッジ抽出

4.1　問題の定義

2値化画像から輪郭となる部分を抽出する．

4.2　問題解析

ソースデータ，結果データを整数型2次元配列とする．複数行から成るブロックに分割し，ブロック毎にスレッドに割り当てる．値が1の点の上下左右の値がすべて1であるならその点はエッジではなく，値が0の点が1つでもあればその点はエッジとなる．スレッドではそれぞれ割り当てられた範囲でエッジを抽出する．各スレッドは独立に計算でき，スレッド間の同期は必要なく，また終了条件もないので，アルゴリズムはプロセッサファームを使用できる．並列化手法はパラレルリージョン，BACS表現はC^tとなる．表3にエッジ抽出のインデックスを示す．類似事例として3次スプラインが検索される．

4.3　事例修正による並列プログラミング

表1と表3はデータ構造が違うだけなので，タスク分割の一部を修正し，単位計算部分を肉付けすることで作成できた．事例修正による並列プログラミングの評価を表4に示す．

表2と表4から，スレッドと同期はスケルトンをそのまま再利用でき，単位計算は殆どを逐次プログラムから再利用できているのが分かる．またエッジ抽出のタスク分割はデータ構造が違うだけなので一部を再利用している．

表3: エッジ抽出のインデックス

応用	画像処理
仕様	2値化画像の輪郭を抽出する．
ソースデータ	整数型2次元配列indata[ ][ ]
結果データ	整数型2次元配列outdata[ ][ ]
タスク分割	indata　, 行ブロック, ブロック outdata , 行ブロック, ブロック
終了条件	なし
トポロジ	マスター&ワーカー
アルゴリズム	プロセッサファーム
並列化手法	パラレルリージョン
インタラクション	なし
BACS	C^t

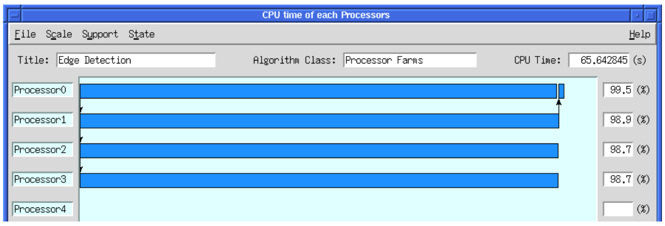
表4: エッジ抽出の事例修正による評価

(行数)

	スレッド	同期	タスク分割	単位計算
再利用	8	1	6	60
修正	0	0	0	5
新規	0	0	15	0

5　並列実行の可視化

各プロセッサの実行，待ち状態を可視化することにより，負荷均衡，同期などの理解を手助けし，並列プログラミングを支援するツールを作成中である．実行中，待機中，同期，稼働率，拡大/縮小の機能がある．図4にエッジ抽出の並列実行状況を示す．



Title: Edge Detection
Algorithm Class: Processor Farms
CPU Time: 65.642845 (s)

Processor0 99.5 (%)
Processor1 98.9 (%)
Processor2 98.7 (%)
Processor3 98.7 (%)
Processor4 (%)

図4: エッジ抽出の並列実行状況

6　おわりに

文字列照合(KMP法)から3次スプライン，さらにエッジ抽出の並列プログラミングを示した．プロセッサファームは構造が単純であるので，スレッド，同期は完全に再利用でき，タスク分割は修正が必要である．現在，システム全体の構築を進めると共に，巡回セールスマン問題の事例修正による並列プログラミングを行っている．タスク分割の修正の軽減，及び並列アルゴリズムの可視化が今後の課題である．

参考文献

[1] 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．

[2] 山崎，松田，安藤，朝倉"事例ベース並列プログラミング"，第38回プログラミングシンポジウム，1997．