4V－6　関数型プログラム実行のための細粒度並列仮想マシンコード DVMCにおける構造体の静的コピー除去

岡崎芳希　日下部茂　谷口倫一郎　雨宮真人
九州大学大学院 総合理工学研究科

1　はじめに

筆者らは汎用並列処理記述言語である超並列V言語の設計および言語処理系の開発を行っている[1]。V言語は原則として関数型セマンティクスに従っており、破壊的代入を認めないため、ある構造データを一部だけ更新する場合、概念的にはその構造データ全体のコピーをとり、新しい構造データを生成する必要がある。しかし実装においては、効率の観点から構造データのコピーの回数をできるだけ削減することが望ましい。この問題に対し筆者らは、まず機種非依存の中間コードの段階において可能な限り静的に構造データのコピー除去を行い、その後、各対象アーキテクチャを考慮した(静的／動的)コピー除去を行うアプローチをとっている。本稿では、中間コードである細粒度並列仮想マシンコード DVMC(Datarol Virtual Machine Code) に対し行う構造データの静的コピー除去の方法について述べる。また、本方式をいくつかの例題プログラムに適用した結果を示す。

2　DVMCの概要

V言語で記述されたプログラムはデータ依存解析を経てDVMCという機種非依存の中間コードに変換される。DVMCは細粒度マルチスレッド・コントロールフローグラフである。本節ではDVMCの概要について述べる。

例として、配列Aのi番目の要素とj番目の要素を入れ換えた配列を返すV言語の式

```
update(update(A,[j],A[i]),[i],A[j]) -- (1)
```

をDVMCに変換したものを図1に示す。

楕円で囲まれているのがノードであり、コピー除去を行う時点のDVMCでは、1ノードが1命令に対応する。ノード間をつなぐ矢印がコントロール・フローを表すアークである。ノードは原則として、全ての入力アークからフローを受け取ると発火し、命令の実行を終了すると全ての出力アークへフローを流す。アークはその出発ノードの命令の種類によって、破線と実線に分けられている。破線は、他のプロセッサとの通信や、関数インスタンスの生成のため遅延が大きい(split-phaseである)命令からでるアークを表し、実線は、遅延することなく後続命令へ制御を移すことができる命令から出るアークを表す。

Static Elimination of Structure Data Copyings on Fine-grain Parallel Virtual Machine Code, DVMC, for Functional Programs
Yoshiki OKAZAKI, Shigeru KUSAKABE, Rin-ichiro TANIGUCHI, Makoto AMAMIYA
Graduate School of Engineering Sciences, Kyushu University

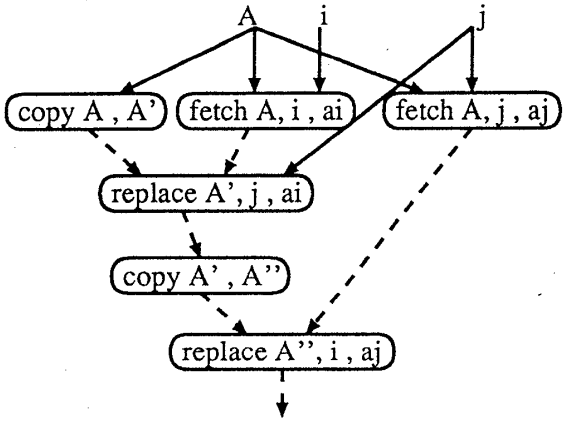

図1：(1)式のDVMC

図1のDVMCにおいて、命令 copy A, A' は、配列A全体をコピーし、新たに生成される配列をA'とする。命令 fetch A, i, ai は、配列Aのi番目の要素を読み出し、aiへ格納する。命令 replace A', j, ai は、配列A'のj番目の要素をaiの値に破壊的に書き換える。ノード中のA、i、aj等は実際にはポインタやレジスタを表す記号であるが、ここでは便宜上変数のように表現している。一番下のreplaceノードを実行した後の配列A"が、(1)式の結果となる。

3　コピー除去法

本節ではDVMCに対して行う構造データのコピー操作の除去法について、簡単な(1)式を例に述べる。実際には、条件分岐命令が存在する場合の解析や、callerから引数として渡されてきた構造データを更新する場合の関数間にまたがった解析を行っているが、それらについては割愛する。

3.1　コピー操作の除去

ここで述べるコピー操作の除去とは、具体的には、DVMC中の各copyノードが、次の条件を満たしている場合、そのcopyノードをmoveノードに変換(ノードから出るアークも破線から実線に変換)することである。命令 move A, A' は、配列Aを指すポインタをA'へ格納する。

【コピー操作が除去可能な条件】

あるcopyノードについて、同じ構造データに対する他のアクセスノードが、次のいずれでもないとき、そのコピー操作は除去可能である。

(1) そのcopyノードの実行以降に実行される
(2) そのcopyノードと並列に実行される

図1のDVMCの場合、左下のcopyノードは上の条件を満たしており、このコピー操作を除去できる。一方、左上のcopyノードは、上の条件を満たしていないので、このコピー操作は除去できない。

単純にデータ依存関係しか反映していないDVMCには上の条件を満たすcopyノードはまだ少ない。除去されるコピー操作を増やすために、DVMCのノードを並び換える操作(node-reordering)を行う。ここで、node-reorderingの中で用いるDVMCの深さを次のように定義する。

【DVMCの深さ】

定義： あるノードにおけるDVMCの深さを、グラフの最上ノードからそのノードまでたどる際に通過する破線のアークの最大本数と定義する。

図1の部分DVMCの場合、一番下のreplaceノードの深さは3である。

3.2 node-reordering

あるcopyノードと並列に実行される、同じ構造データに対するアクセスノードの一つ一つからそのcopyノードへACA (Artificial Control Arc)を挿入する。これによりcopyノードの実行は他のアクセスノードの実行以降に延期される。ただし、ACAの挿入により並列性が損なわれるため、ACAの挿入によるコピー除去は活用できる並列度とのトレードオフを考慮して行う必要がある。そこで、DVMC全体の深さが大きくならなければ並列性を損なっていないとみなし、次の条件を満たしている場合のみ、ACAを挿入する。

【ACA挿入の条件】

あるcopyノードと並列に実行される、同じ構造データに対するアクセスノードの深さの最大値が、そのcopyノードに対応するreplaceノードの深さ未満である。

図1のDVMCの場合、右上の2つのfetchノードから左上のcopyノードへACAが挿入される。(1)式について、node-reorderingを行い、コピー操作の除去を施したDVMC(冗長なアークは取り除いてある)を図2に示す。

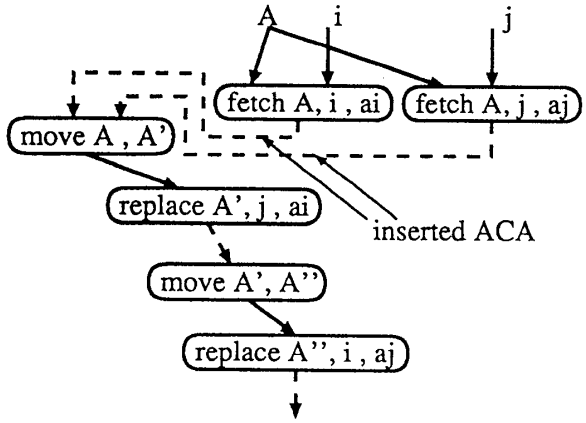

図2：(1)式についてコピー除去を行ったDVMC

図2のDVMCから分かるように、一番下のreplaceノードの深さは2である。図1のDVMCに比べ、全てのコピー操作を除去できており、DVMC全体の深さも1減っている。

4 結果

前節で示した方法をいくつかの例題プログラムに適用した結果を表1に示す。

プログラム	初期数	除去数		残数
		ACA無	ACA使用(本数)	
quicksort.v	2	1		1
bubblesort.v	2	1	1 (2)	
gauss-jordan.v	2		2 (5)	
inverse.v	4	2	2 (4)	
loop02.v	1		1 (3)	
loop10.v	10	1	9 (9)	
loop19.v	2		1 (1)	1
loop20.v	2	1	1 (4)	

表1：除去されたコピー操作の数(空欄は0を表す)

適用した例題プログラムは、クイックソート、バブルソート、ガウス-ジョルダン法、逆行列、およびリバモアループ(2, 10, 19, 20番)である。

表2から、かなりのコピー操作を除去できることが分かる。また、node-reorderingによるACA挿入の効果が大きいことも分かる。クイックソートは並列アルゴリズムであるため、本質的に除去されないコピー操作がある。このコピー操作は、後の機種依存の最適化によっても除去することはできない。

類似研究に適用型言語SISALのコピー除去[2]があるが、コピー除去率を高めることにのみ主眼をおいており、並列性の損失を考慮していない。そのため、コピー除去による実行時間短縮の保証はないと思われる。

5 おわりに

本稿では、関数型プログラム実行のための、機種非依存の中間コードDVMCにおける構造データの静的コピー除去の方法について述べた。また、本方式を用いることにより、並列性を損なうことなく、かなりのコピー操作が除去できることを示した。

参考文献

[1] 日下部, 他:"超並列V言語とその実行方式", 並列処理シンポジウムJSPP'94論文集, pp.41-48, May 1994.

[2] Steven M. Fitzgerald:"Increasing Parallelism for an Optimization that Reduces Copying in IF2 Graphs",Proceedings SISAL'93, pp.74-84, Oct. 1993.