再帰的非ストリクト構造データ生成の効率化

4X－6

稲永健太郎　日下部茂　雨宮真人
九州大学 システム情報科学研究科 知能システム学専攻

1 はじめに

関数型のセマンティクスを持つ非ストリクトな言語は，他言語に比べより柔軟なプログラム記述を可能とする．この種の言語が持ついくつかの非ストリクト性のうち，特に構造データの非ストリクト性が有用であるといわれている．

非ストリクトな構造データの生成において，我々が従来用いてきた手法では構造データの要素を生成する計算体間の動的なスケジューリングが実行時のオーバヘッドの主な原因の１つであった．しかし，array comprehension のようなある要素を同一構造データの他要素によって定義する再帰的な非ストリクト構造データの生成では，要素間の参照依存関係により要素を生成する計算体間の実行順序を静的に決定することができる場合がある．本稿ではそのような場合に計算体をループ化する手法について述べる．

2 ループ化決定法

2.1 要素間の依存関係の解析

ループ化の可能性を判断する材料となる要素間の依存方向ベクトルを，参照される要素ごとに生成する．

解析対象となる配列を n 次元の配列 A，各次元を $j(1 \leq j \leq n)$，各次元の添字変数を i_j とし，定義される配列要素を $A[i_1, \cdots, i_j, \cdots, i_n]$，参照される配列要素を $A[f_1(i_1), \cdots, f_j(i_j), \cdots, f_n(i_n)]$ と表す．(ただし，関数 f_j は変数 i_j の線形式で表される)．この時，参照される配列要素ごとに方向ベクトル $\vec{\theta}$ を次のように定める．

$$\vec{\theta} = (\theta_1, \cdots, \theta_j, \cdots, \theta_n)$$

ここで，方向ベクトルの成分 $\theta_j(1 \leq j \leq n)$ は以下の表に従い決定される．

条件式	θ_j	次元 j での実行順序
$f_j(i_j) < j$	$<$	昇順
$f_j(i_j) > j$	$>$	降順
$f_j(i_j) = j$	$*$	任意
上記以外	$\times$	決定不可能

Efficient Generation of Recursively-defined Non-strict Structured Data
INENAGA Kentaro, KUSAKABE Shigeru, AMAMIYA Makoto
Department of Intelligent Systems, Kyushu University
6-1 Kasuga-Koen, Kasuga, Fukuoka 816, JAPAN

2.2 計算体のループ化決定法

参照される要素ごとに生成した方向ベクトルをもとに，以下に述べる方法により要素を生成する計算体をまとめるループ化が可能かどうかを決定する．

ループ化決定法

1. 各方向ベクトルを集合 L または 集合 NL のいずれかに分類する．

   集合 L：
   全ての成分 $\theta_j(1 \leq j \leq n)$ が，$>$，$<$，$*$ のいずれかの値を持つ方向ベクトルの集合

   集合 NL：
   ある成分 $\theta_k(1 \leq k \leq n)$ が $\times$ であるような k が存在する方向ベクトルの集合

2. 1. において分類された方向ベクトルのうち，集合 L に属する方向ベクトルが次の 2 つの条件を満たすかを調べる．

   (a) 参照する全ての要素ごとに生成された方向ベクトルのが集合 L に属している

   (b) 集合 L に属する全ての方向ベクトルが等しい
   ここで，$*$ は $>$ や $<$ に置き換えて方向ベクトルの比較が可能である．

これら条件を満たしていれば，そのようなベクトルを持つ要素への参照を含む計算体をまとめてループ化可能である．そうでなければ，各計算体の実行が途中で中断する可能性が残っているため，我々が従来用いている計算体間の動的なスケジューリングに従って実行する [1]．

3 評価および考察

3.1 ループ化決定法の評価

以下に示す 3 つの言語で記述されたプログラムを汎用ワークステーション上で実行させ，実行時間を比較することにより今回述べたループ化の効果を評価する．

1. V: 関数型言語 V 言語 (非ストリクト言語)[2]
2. C: 手続き型 C 言語
3. Sisal: 関数型言語 Sisal (ストリクト言語)

今回の例題として，V 言語で記述した場合は簡潔に配列を再帰的に定義できる wavefront プログラムを用いる．この wavefront プログラムは，要素間の依存関係の解析により要素を生成する計算体をまとめてループ化することが可能である．評価結果を図 1 に示す．

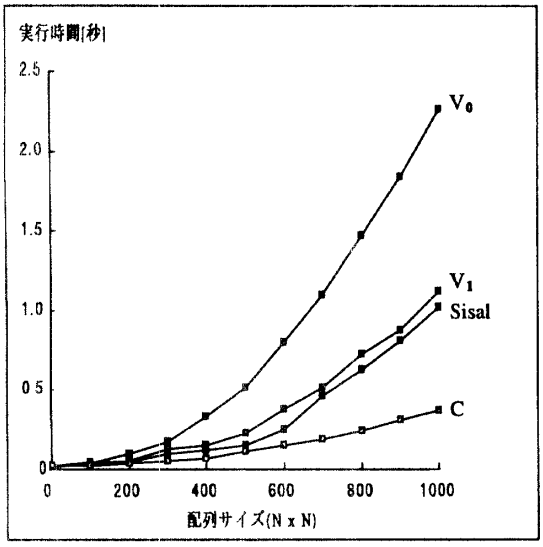

図 1: wavefront プログラムの実行時間 (その 1)

図 1 より，各計算体を従来の方法で実行させた場合 (V_0) に比べてループ化決定法を適用した場合 (V_1) は，約 50% の実行時間の短縮が見られ，ループ化による効果が確認できる．また V_1 の実行時間と Sisal で記述されたプログラムの場合の実行時間がほぼ同等であり，V 言語のような非ストリクト言語でも，Sisal のような実行時システムのオーバヘッドが比較的小さいストリクト言語と同等の性能が得られることが確認できた．

V_1では，手続き型言語 C 言語で直接記述されたプログラムの場合の実行時間に及ばなかった．この実行時間の差は，V 言語の構造データへのアクセスにかかるコストによるものと考えられる．V 言語では，構造データの要素に同期処理機構を備えることで，プログラム中のデータの生産者／消費者間の並列性を最大限に抽出できる．しかし，今回評価を行っている逐次計算機のように何の並列性も生かせず静的にループ化可能な例題においては，要素ごとに同期処理機構を備えるメリットはなくオーバヘッドだけが生じてしまう．

3.2 逐次実行に特化した最適化の評価

今回の評価では逐次計算機であるワークステーションを用いているため，さらに逐次実行に特化した最適化を試みる．具体的には，V_1 で用いている構造データを C 言語の配列と同様に実現する．この場合 (V_2) の実行時間を図 2 に示す．

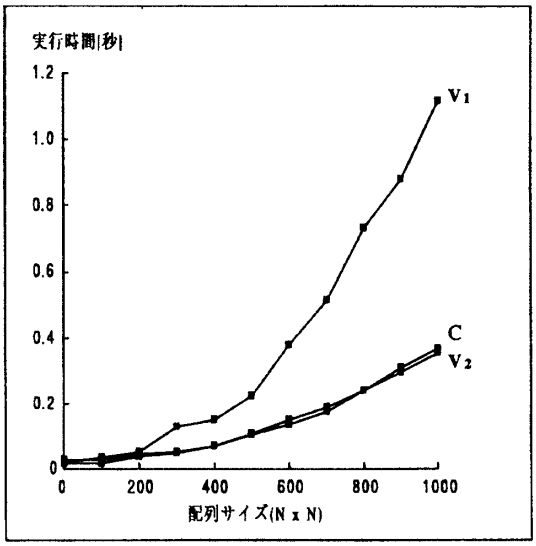

図 2: wavefront プログラムの実行時間 (その 2)

図 2 より，まず V_1 に比べて V_2 は，最大約 60%の実行時間の短縮が見られる．さらに，V_2 と C との実行時間の差はほとんどなく，逐次実行に特化した構造データの実現方法を用いることで手続き型言語のプログラムと同等の性能を引き出すことが可能であるという知見が得られた．

4 おわりに

本稿では、再帰的非ストリクト構造データ生成において要素を生成する計算体間のスケジューリングのコストを下げるためのコード生成法について述べた．例題を評価した結果，非ストリクトな言語についても計算体をループ化することでストリクトな言語と同様の性能が得られ，さらに逐次実行に特化した最適化を適用することで手続き型の言語と同様の性能が得られることが示された．今後は，より効率的な実行を実現するため，実装対象の計算機の特性を生かした構造データへのアクセスのコストを下げるための処理系の改良を行う予定である．

参考文献

[1] 日下部茂, 森本徹夫, 雨宮真人. “非ストリクトデータフロープログラム実行におけるスタックフレームの利用”. In *IPSJ SIG notes PRO 14-13*, Aug 1997.

[2] 日下部茂, 雨宮真人. “超並列 V 言語”. 重点領域研究「超並列原理に基づく情報処理基本体系」第 4 回シンポジウム予稿集, pp. 2-143 - 2-190, 1994.