5F－9

Committed-Choice 型言語 Fleng による並列数式処理

馬場 恒彦，日高康雄，小池汎平，田中英彦
東京大学工学部*

1 はじめに

現在では、REDUCE や Mathematica に代表される汎用数式処理システムが工学・理学を始めとする様々な分野で実用的に利用されてきている。一方、処理速度の向上を目的とした、数式処理システムの並列化の研究が行なわれてきている。しかし、従来の逐次型アルゴリズムの並列が容易な部分を並列 C を用いて部分的に並列化した粗粒度なシステムが多く、数式処理システムを全面的に並列化した例はまだ見受けられない。

そこで、本研究で用いた当研究室で開発した Fleng [Nilsson 86] と PIE64 を用い、細粒度並列処理言語上にシステムを構築し、数式処理の全面的な効率化をはかった。本稿ではその概要について示す。

2 研究の背景

本章では本研究で用いた Fleng と PIE64 について述べる。

Fleng は当研究室で開発された細粒度並列論理型言語である。並列 C や並列 Lisp とは異なり、同期等の並列処理を行なうための命令を記述する必要がなく、並列に実行可能な部分が存在すれば並列に実行される特徴をもつ。int の範囲を越える bignum は処理系で現段階ではサポートされていない。そのため、本研究では 2^{10} 進数のリスト表現を用いた。

また、PIE64 も Fleng 同様、当研究室で開発された並列推論マシンで、64 台の要素プロセッサ (PE) をもち、Fleng を高速実行するように設計されている。

3 並列数式処理のアルゴリズム

数式処理では、多項式の演算が中心となる。本研究ではこの多項式演算を並列化の対象とした。本章では実装中のシステムに使用したアルゴリズムについて述べる。

3.1 数値演算アルゴリズム

数値演算のうち冪乗・階乗・bignum 型の数の乗算について並列化を行なった。

冪乗・階乗の演算は 2 分割し再帰的に演算する手法が知られている。この手法を用いると、演算量を減らすことができるのと同時に、1 つの演算を 2 つの演算に分割しているため、並列に実行することが可能である。また、bignum 型の数の乗算も被乗数と乗数の各桁への乗算が互いに独立に実行可能であるので、同様な並列化を行う。

以上の点を改良したアルゴリズムと、順番に乗算を行なう逐次的アルゴリズムの実装とその評価を行なった。

3.2 GCD アルゴリズム

GCD アルゴリズムとして擬剰余を用いた Euclid の互除法があるが、逐次的アルゴリズムであり、また Knuth の例 [Knuth 69] に見られる中間表現膨張問題を生じる。

そこで、並列化可能なモジュラアルゴリズムを用いる。モジュラアルゴリズムは入力 f を元の代数的空間 D より単純な代数的空間 D' に写像 (ϕ) し、D' 内でその写像 f' を入力として、出力 g' を計算する。最後に g' を D へ逆写像 (ϕ^{-1}) して出力 g を得るアルゴリズムで、D から D' への写像は準同型写像であるので、一意に出力 g を求める為には、複数の代数的空間が必要となるが、各代数的空間の計算は互いに独立であるので並列に実行することが可能である。(図 1)

本研究ではモジュラ GCD アルゴリズムとして Brown [Brown 71] の展開した素数を法とするアルゴリズム GCD.MODINT を用いた。以下に、並列化のための具体的な改良点を示す。(逐次的アルゴリズムは文献 [Sasaki 81] を参照した)

- 簡単な代数的空間 D' 上の並列化
  モジュラアルゴリズムには、計算結果が有効ではないアンラッキな法が存在する。法がアンラッキであ

*Pararell Computer Algebra by Committed-Choice Language Fleng
Tsunehiko BABA, Yasuo HIDAKA, Hanpei KOIKE, Hidehiko TANAKA,
Faculty of Engineering,the University of Tokyo

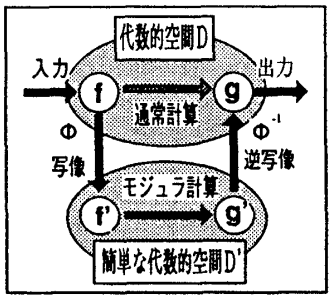

図 1: モジュラアルゴリズムの原理

る場合、その法での計算結果が棄却される。そこで、幾つかの法でのGCD演算を並列に実行し、解の探索を行う。

- 逆写像 ϕ^{-1} の並列化
  複数の代数的空間 D'_n の出力 g'_n を逆変換 ϕ_n^{-1} する際、多項式の各項の数係数は規格化により、2つの値をとりうる。故に、出力 g' が n 次の密な多項式であれば、2^{n+1} 個の写像 g の候補が存在する。そこで、各候補に対する処理を独立に実行し、並列化をはかることが可能である。また、候補に解が含まれない場合があるので、探索の終了を検知に、並列論理型言語でのショートサーキットと呼ばれる並列全解探索の手法を用いる。

4 結果・考察

数値演算アルゴリズムの冪乗・階乗・bignum型の乗算について、並列処理用に改良したアルゴリズムと逐次的に計算を行うアルゴリズムとを用い、演算に要した時間の比較と台数効果による速度の比較を行なった。それぞれの結果を、表1と表2に示す。(表2の数値はPE数が1の時の実行速度を基準(1.00)ししている)

逐次的アルゴリズムでは数分の演算時間要するが、並列化したアルゴリズムでは数秒と高速になっている。

また、逐次的アルゴリズムの速度の向上はリスト操作等のその他の処理に細かな並列性が存在するためである

表 1: 演算時間 (単位:sec)

	冪乗		階乗	
	3^{200}		100!	
PE	逐次	並列	逐次	並列
1	488.8	14.2	707.7	74.6
64	190.2	2.05	292.4	4.24

表 2: 演算速度の向上

	冪乗		階乗	
	3^{200}		100!	
PE	逐次	並列	逐次	並列
1	1.00	1.00	1.00	1.00
2	1.04	1.60	1.04	1.81
4	1.46	2.61	1.27	3.28
8	2.19	4.12	2.11	4.85
16	2.41	5.48	2.29	9.35
32	2.52	6.36	2.38	13.7
64	2.57	6.93	2.42	17.6

が、著しい速度の向上はみられず、PE64台使用時でも2.5倍程度の速度向上しか得られない。一方、並列化したアルゴリズムでは 3^{200} ではPE数32台まで、100!ではPE数64台まで速度の向上が見られた。

5 おわりに

並列推論マシンPIE64上でのCommitted-Choice型言語Flengによる、数式処理の並列化について、数値演算の細分化と多項式GCDアルゴリズムでの改良点を示し、数値演算については評価を行なった。

現在、改良されたモジュラGCDアルゴリズムの実装を行っている。

参考文献

[Nilsson 86] Nillson,M. and Tanaka,H.: Fleng Prolog - The language which turns Supercomputers into prolog Machines, In Wada,E.(Ed.): Logic Programming '86,LNCS 264, Springer-Verlag, 1986.

[Knuth 69] D.E.Knuth: The Art of Computer Programming, Vol.2, Seminumerical Algorithms, Addison Wesley, §4.6.1, 1969.

[Brown 71] W.S.Brown: On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisiors, J. ACM 18, No.4, p478, 1971.

[Sasaki 81] 佐々木建昭: 数式処理, 情報処理叢書 7, §1.1 〜§2.4 , 情報処理学会, 1981.