4E－2

逐次言語との複合プログラミングを行うデータフロー並列処理言語の実現

松浦 健一郎, 近山 隆
東京大学 工学系研究科

1. はじめに

並列処理を簡単かつ正確に記述できる言語の必要性は高い。しかし、既存の知識や資産を活かしつつ、安全で効率的な並列処理を可能にするのは難しい。

本研究では、逐次型言語を組み入れてプログラミングを行う新しい並列型言語「POOH」を実現する。POOHはシンプルな言語仕様を持ち、習得が容易な並列型言語である。加えて、使い慣れた既存の逐次型言語との複合により、並列プログラミングを簡便にする。同時に、単一代入を利用したデータフロー管理機構や、自動負荷分散機構を装備することにより、並列処理の安全性や効率の向上を図る。POOH処理系は、現在SolarisおよびWindows95/NT上にて実装・評価中である。

逐次型言語プログラムを並列型言語によって制御するという考え方はCoordination Programming[1]に近いが、本研究は、並列型言語プログラムの内部に逐次型言語プログラムを直接記述するなど、両プログラムの密着性が高いという特徴を持つ。

単一代入を用いてC/C++プログラムの並列実行制御を行う言語としてはCC++[2]がある。しかしCC++では単一代入が同期の補助的手段として位置づけられているのに対し、本研究では単一代入に基づくデータフロー管理機構を言語の核としている点で、方向性が異なる。

2. 言語

POOHでは、逐次型言語との複合プログラミングを行う。現時点ではC/C++であるが、将来は他の逐次型言語にも対応する予定である。POOH自体は独自の言語仕様

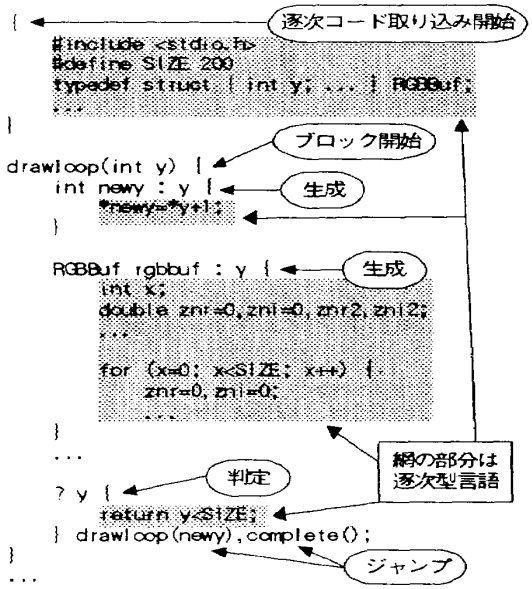

図 1：POOHプログラムの例

を持つが、スタイルはC/C++に近い。POOHプログラムの例を図1に示す。網の部分はC/C++プログラムであり、それ以外はPOOHプログラムである。

POOHは、データフローに注目した並列型言語である。すなわち、様々な処理を一般に**「幾つかのデータを参照しつつ、一つの新しいデータを生成する」**という単位に分割することによって扱う。POOHではこれを**生成**と呼ぶ。生成文は以下のような形式を持つ。

```
new : ref0, ref1, ... refN { program };
```

この文は**「ref0,ref1,...refNを参照しつつ、programに記述された処理を行い、その結果newを生成する」**という意味である。programは具体的な処理の内容であり、既存の逐次型言語を用いて記述する。programでは参照データを使用しつつ、新データを作成する。

逐次型言語プログラムから参照や変更が可能なのは、POOHの生成文にて指定されたデータのみである。他のグローバルデータへのアクセスは禁止される。また逐次型言語プログラム内部では一時的なローカルデータの使用が可能だが、このローカルデータが他の生成文

A Dataflow Parallel Language for Hybrid Programming with Sequential Language
Ken-ichiro Matsuura and Takashi Chikayama
{makken, chikayama}@logos.t.u-tokyo.ac.jp
School of Engineering, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

に影響を及ぼすことは無い。従って、各生成文が伴う逐次型言語プログラムは、並列実行が可能となっている。さらに、逐次型言語プログラムにおいては、アクセス可能なデータが制限されることを除けば、並列実行されることに関して一切配慮する必要がない。

また、POOHでは単一代入を採用している。すなわち、同一の名前を持つデータを複数回生成することを禁止している。単一代入の採用によって、並列処理のためのデータフロー管理が容易になった。単一代入では膨大な数の名前が必要にならないようにとの工夫が必要であるが、POOHでは「ブロック」と呼ぶ単位でプログラムを分割し、ブロック内でのみ単一代入を義務づけることによって、問題を回避している。

POOHは、生成やブロックの他に、ジャンプ、判定、宣言といった言語機能を持つ。

3. 処理系

3.1 構成

POOH処理系は、コンパイラとランタイムルーチンから構成される。コンパイラは、POOHプログラムをC/C++プログラムに変換する。このC/C++プログラムを既存のC/C++コンパイラにてコンパイルすることにより、実行形式を得る。また、リンクするランタイムルーチンを選択することにより、複数の実行環境に対応する。

POOHは共有メモリ型密結合マルチプロセッサシステムを想定している。スレッドライブラリが用意されている環境ならば、移植は容易である。

3.2 データフロー管理機構

POOHでは、データフローを管理することにより、生成の実行タイミングを制御する。生成を実行する際には、まず「参照データの値が既に決定しているかどうか」を調べる。もしも全てが決定されているならば、生成を実行する。すなわち、各生成文が伴う逐次型言語プログラムを実行する。

もしもいずれかの参照データの値が未決定の場合には、実行せずに、値の決定を待つ。そして値が決定した後に、改めて実行する。値の決定を待つ間に、他の実行可能な処理を先に実行することにより、プログラムが持つ並列性を活かした性能向上を図っている。

3.3 自動負荷分散機構

マルチプロセッサシステムの場合、各プロセッサに対する処理の分担方法が問題となる。POOHではこのような負荷分散を自動的に行う。負荷分散は実行時に動的に行われるので、プロセッサ数にかかわらず、実行環境ごとの性能を生かすことができる。

負荷分散は、生成文を単位として行われる。各生成文の処理の大きさが不均一かもしれないという問題は、動的負荷分散により軽減される。また、ユーザ側は負荷分散に関してあまり考慮することなく、並列化の恩恵を受けることができる。特に、既存の逐次型言語プログラムの並列化が容易になるであろう。

4. おわりに

現在POOH処理系は、SolarisおよびWindows95/NT上にて動作している。いずれの環境でもマルチプロセッサシステムにおける並列実行が可能である。

10CPUのUltra SPARCマシンにてマンデルブロ集合を計算するPOOHプログラムを実行したところ、ほぼプロセッサ数に比例した速度の向上が見られた。マンデルブロ集合では座標により計算量が大きく異なるので、動的負荷分散が速度向上に貢献していると考えられる。なお、POOHからC/C++へのコンパイル時間は、C/C++プログラムのコンパイル時間に対し数分の一程度である。

今後、より多くのプログラムの作成を通じて、性能や使いやすさの改善を重ねていく予定である。また言語機能の充実や、コンパイラによる最適化処理なども併せて進めていく予定である。

参考文献

[1]J.-M.Andreoli, C.Hankin, D.Le Metayer: "Coordination Programming —— Mechanisms, Models and Semantics", Imperial College Press, 1996.
[2]Paul A.G.Sivilotti, Peter A.Carlin: "A Tutorial for CC++", Caltech CS-TR-94-02, http://globus.isi.edu/ccpp/tutorial/tutorial.html, 1994.