7M−4

高並列推論エンジンPIE64研究経過報告 − ソフトウェア −

中田秀基, 小池汎平, 吉田実, 舘村純一, 白木長武, 田中 英彦

{nakada,koike,minoru,tatemura,shiraki,tanaka}@mtl.t.u-tokyo.ac.jp

東京大学 工学部 電気工学科

1 はじめに

我々は、大規模知識処理のための高並列推論エンジン PIE64 と本機向けの言語 Fleng の研究を進めてきた。本稿では Fleng の処理系および周辺技術に関する研究 / 開発の現状に関して報告する。

2 Fleng

大規模知識処理を高並列に記述するための言語には以下の特徴が求められる。

- 問題から並列度を自然に抽出することができる。
- 複雑で非定型な問題を抽象化して記述できる。

我々はこのような特徴を持つ言語として 言語 Fleng を設計、実装してきた。Fleng は論理型言語を祖先に持つ Committed choice 型言語の一つである。他の Committed choice 型言語に対して、高速な実行を可能にするために、言語の仕様を簡潔にした点に特徴がある。

Fleng では、変数はすべて単一代入である。ヘッド部に記述したパターンに対してゴールがパターンマッチすることが、単一代入変数に対する暗黙の同期になる。このため、非常に細粒度の同期をプログラマが特に意識せずに、自然に記述できる。

Fleng の処理系は現在ワークステーション(SPARC station、LUNA 88K 等) 上で稼働している。これらの処理系は C 言語によって記述されている。このうち、LUNA 88K 上の処理系はマルチプロセッサに対応しており、最大4つの CPU を用いて並列実行を行なっている。

これらの処理系上に、マルチウインドウ・デバッガ、並列非単調推論システムなどのアプリケーションが実装されている。

3 Fleng コンパイラ

Fleng の コンパイラは Fleng 自身によって記述されている。コンパイラは大きく二つのフェイズに分けられる。中間コードを生成する部分と、中間コードからターゲット・コードを生成する部分である。後半のフェイズを差し替えることによって、さまざまなターゲットに対応することができる。

ワークステーション上での処理系ではコンパイラはターゲット・コードとして C 言語のコードを生成する。これは、以下の理由による。

- ポータビリティが優れている。
- C 言語コンパイラのオプティマイザを活用できる。

現在ターゲットコードとして稼働しているのは以下のものである。

- C 言語
- UNIRED アセンブラ
- 88000 アセンブラ

コンパイルされたコードはオブジェクト・コードとして、上述の処理系に動的にロードされ実行される。このため、コンパイルされたコードと、インタプリタ用のソースコードが処理系上で、共存できる。

4 Fleng の静的解析

高並列な実行には、実行対象の粒度が細かいことが必要である。しかし、粒度が細かいと言うことは同期をとる点が多くなるということを意味し、対象となる問題の並列度がプロセッサの数と比較して大きい場合にはこの同期はオーバーヘッドになる。

このオーバーヘッドと、並列度のトレードオフは対象となる計算機の性質と、問題の固有の並列度に依存するが、とくに、プロセッサ数の少ないワークステーション上の処理系においては、プログラムを静的に解析し粒度を粗くすることによって、このオーバーヘッドを低減することが高速実行には欠かせない。また、充分なプロセッサを持つ PIE64 においても、対象となる問題のサイズが大きくプロセッサ数以上の並列度をもつばあいには、、並列度を減らして、オーバーヘッドを低減することが重要になってくる。

我々は、これらオーバーヘッドの低減のための手法として、プログラムに対して静的に解析を行ない、粒度を上げることによって不必要になる同期のポイントを検出した。この際、粒度を上げるポリシーは、プロセッサ数と問題の性質に応じて人間が設定している。

また、計算の間に本質的な依存関係が存在し、必然的に計算がシリアライズされる場合がある。この場合には、これらの計算を同時に実行しようとしてもできず、並列度があがらない上にいたずらに同期のコストがかかってしまうことになる。このオーバーヘッドも同様に、静的解析を行なうことで検出できる。

この検出された結果をアノテーションとしてプログラムのソースコードに付加し、コンパイラへの指示とすることで、最適なコードを生成している。

5 Fleng プログラムのデバッグ

細粒度高並列プログラムにおいては、通常の逐次プログラムと異なり、制御の流れやデータの流れが分散して多数存在する。このため、逐次プログラムにおいては一般的なデバッグ手法である、プログラムの流れをトレースする方法は、細粒度高並列プログラムに適応することはできない。細粒度高並列のプログラムのデバッグには多様な情報を高度に抽象化し、多次元的にプログラマに提示する必要がある。このため、本研究室ではマルチウインドウのデバッガ HyperDEBU[1] を開発した。

HyperDEBU は、プログラマに対して提示するべき情報として、制御の流れ、データの流れ、構造データをそれぞれ最適に抽象化し表示する。そのために以下の種類のウインドウを持つ。

1. トップ・レベル・ウインドウ
   プログラムの実行をグローバルに観察・操作する
2. プロセスウインドウ
   (a) TREE ウインドウ
   プロセス内の計算木(制御の流れ)の観察・操作を行なう

Progress report of highly-parallel inference engine PIE64 – Software –
Hidemoto NAKADA, Hanpei KOIKE, Minoru YOSHIDA, Jun'ichi TATEMURA, Osamu SHIRAKI, Hidehiko TANAKA
Department of Electrical Engineering, The University of Tokyo

(b) I/O tree ウインドウ
プロセス内の入出力因果関係(データの流れ)の観察・操作を行なう

(c) GOAL ウインドウ
プロセス内のゴールの観察・操作を行なう

3. ストラクチャ・ウインドウ
構造データを観察するウインドウ

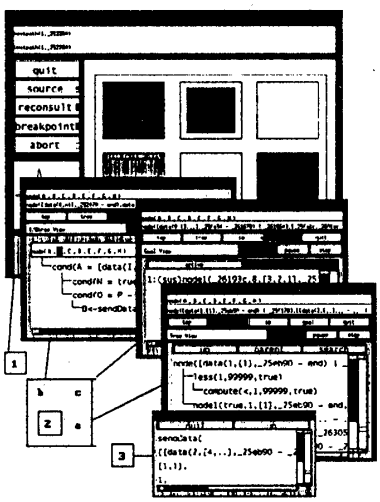

図 1: マルチウインドウ・デバッガ HyperDEBU

HyperDEBU は、すでに数年にわたって本研究室で用いられており、その有効性とアプローチの正しさが確認されている。

6 パフォーマンス・デバッギング・ツール

大規模知識処理を実現するためには、高並列言語によって実装されたシステムが効率的に稼働しなければならない。しかし、大規模なシステムのパフォーマンス悪化の原因の発見は一般に困難である。特に、高並列言語によって記述されたシステムにおいては、逐次型言語では発生しない、データ、制御の集中に起因するパフォーマンスの低下が生じることがある。このため、システムの並列度を下げる、設計時点での誤りを検出、修正するためのパフォーマンス・デバッグの支援が必要となる。

このためのツールとして、我々は パフォーマンス・デバッギング・ツール paf を開発した[2),5)]。paf は、プロセッサ数は無限大、理想的なスケジューリングを仮定した、仮想時刻を導入することで、実行環境の影響を排除したプログラムの挙動の把握を実現している。

paf は、Fleng インタープリタ pafleng が出力したトレース・データをインタラクティブに表示するマルチウインドウ・インターフェイス をもつ。paf は、さまざまなウインドウを提供することでプログラムに関して複数の側面からの視野を与え、プログラムの挙動の把握を助ける。

7 Fleng の言語拡張

Fleng の言語仕様を拡張することによって、より効率的なプログラミングを可能にしようとする研究も行なわれている。Fleng によるプログラミングをもっとも困難にしているのは、Fleng は構造データを表現する方法が貧弱であると言う点である。Fleng にオブジェクト指向を導入することでこの欠点を解決する試みがなされている。

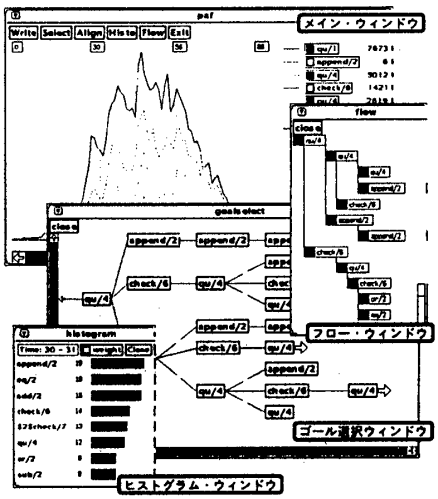

図 2: パフォーマンス・デバッグ・ツール paf

Fleng++[3)] は、Fleng で記述されることの多いパーベチュアル・プロセスを動的なオブジェクトとして明示的に記述する方法を Fleng に追加したものである。

また、F[4)] は Fleng に対して静的な単一代入のオブジェクトを追加することで記述力の向上をはかっている。

8 Fleng 処理系の PIE64 への実装

現在、Fleng 処理系の実機への実装が行なわれている。PIE64 は、Fleng の基本的実行モデルを意識して設計されているため、実機への実装は比較的容易である。すでにワークステーション上での処理系によって多数のアプリケーションが記述されていることから、ワークステーション上の処理系との互換性を意識した実装が行なわれている。

9 おわりに

PIE64 向けの言語 Fleng の特色、その処理系および周辺技術に関する研究 / 開発の現状に関して述べた。

参 考 文 献

1) 舘村純一, 小池汎平, 田中英彦. 細粒度高並列プログラムの実行の視覚化. 情報処理学会研究報告 計算機アーキテクチャ, Vol. 91, No. 64, Jul. 1991.

2) 舘村純一, 白木長武, 小池汎平, 田中英彦. 並列論理型言語 fleng のパフォーマンスデバッギング
-toward a programming environment. 情報処理学会研究報告 プログラミング−言語・基礎・実践−, Vol. 92, No. 67, Aug. 1992.

3) 吉田実他. 並列オブジェクト指向言語におけるオブジェクト内並列性. In *JSPP'91*, 1991.

4) 中田秀基, 田中英彦. 高並列言語 fleng における型システムの導入. 情報処理学会研究報告 プログラミング−言語・基礎・実践−, Vol. 92, No. 67, Aug. 1992.

5) 白木長武, 舘村純一, 小池汎平, 田中英彦. 高並列プログラムのパフォーマンス・デバッギング・ツール paf. 情報処理学会研究報告 プログラミング−言語・基礎・実践−, Vol. 92, No. 67, Aug. 1992.