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

6Q−3

白木長武　舘村純一　小池汎平　田中英彦
東京大学工学部

1　はじめに

並列プログラムにおいて十分なパフォーマンスが得られないことがあるが、その理由はさまざまである。今回、アルゴリズムの問題を発見するのに有用と思われるパフォーマンス・ツールを設計し一部を実装したので、その詳細および使用経験について報告する。

2　パフォーマンス・デバッギングの問題点

並列プログラムのパフォーマンスが出ない原因はさまざまである。通信などのオーバーヘッドの増加や、プロセッサへの不適切な負荷分散がその原因となることもあるが、アルゴリズムそのものに逐次実行部分を多く含んでしまうために並列プログラムの性能が得られないという事態を経験することも多い。一般にこれらの要因は互いに独立ではなく、ある部分の修正は他に影響をおよぼす。

パフォーマンス低下の原因を探る手法としては、プログラムの部分ごとにパフォーマンス・データを集計するプロファイリングと、プログラムの実行に沿って各タスクの挙動を記録し検証するトレーシングとがある。プロファイリングではプロセッサの負荷などの大域的な情報が得られるが、タスク間の依存関係などの細かな情報は得られない。一方トレーシングでは、各タスクの挙動を観察し、パフォーマンス低下をひきおこす部分を発見することができる。

実際のマシン上で実行しその様子を観察するパフォーマンス・モニタを使って、プロセッサの負荷や通信量などを測定し、その解析によりパフォーマンスを向上させる手法がある。その手法では、実行速度を決定するさまざまな要因の混在した状況での実行結果が観察されるので、パフォーマンスが出ない原因の絞り込みが難しい。また、特定の実行環境でのチューニングであるので、チューニングされたプログラムが、他の実行環境においても良いパフォーマンスが得られるとは限らない。

3　Fleng

本ツールの対象としている言語は Committed-Choice 型言語 (CCL) Fleng[NT86] である。

Fleng は他の CCL に較べてその言語仕様が簡潔な点が特徴である。Fleng はガードゴールを持たず、ヘッドのみがガードの働きをするため、ヘッド・ユニフィケーションだけで定義節がコミットされる。

Fleng の実行は、ゴール生成とそのリダクションによって進行し、変数の束縛によって実行が制御される。したがって Fleng には、tree 状のコントロール依存関係と、複雑なデータ依存関係とが存在する。

4　Paf

実行環境の影響を少なくする、並列プログラムのパフォーマンス・デバッギングのスタイルは

1. まずアルゴリズムの問題を解決
2. その後に他の問題を処理

というステップとなる。そのために、アルゴリズムに起因するパフォーマンス・バグを発見できるツール paf を開発している。

実行環境の影響を受けずにアルゴリズムの問題点を発見するために、paf では以下に述べる実行モデルを用いている。

4.1　Paf における Fleng 実行モデル

プログラマが自分の設計した並列アルゴリズムに内在する問題点を明解に把握するためには、並列アルゴリズムの性能を、プロセッサ台数やスケジューリング戦略などシステム・パラメータと独立に評価できることが望ましい。

そのようなパラメータの制約を受けないモデルは

- プロセッサ数は無限大
- 理想的なプロセス・スケジューリング

である。このモデルでは、実行可能なゴール・リダクションはすぐに実行される。

4.2　仮想時刻

以上のようなモデル上でのプログラムの実行を観察するために、仮想時刻および仮想並列度という概念を導入する。

仮想時刻とは以上のようなモデル上でプログラムを実行をしたときの時刻であり、仮想時刻から計算される並列度を仮想並列度と呼ぶ。

Fleng プログラムの実行において、ゴール・リダクションの仮想時刻、プログラム実行の仮想並列度は以下のように計算される。

あるゴールがリダクションされた結果生成されたゴールは、それをヘッドに持つ定義節の引数のユニフィケーションを待つ。ヘッド・ユニフィケーションに成功した定義節のうち 1 つがコミットされ、ボディ・ゴールのリダクションが開始される。実行ではゴールの生成および変数のユニフィケーションがおこなわれる。図 1 にゴールが生成されてから、そのリダクションが終了するまでの様子を示す。

ゴールが生成され (T1)、サスペンション・チェックがおこなわれる (T2,T3,T4)。この図ではチェックにかかる時間は 1 と仮定

Paf: Performance Debugging Tool for Highly Parallel Programs
Osamu SHIRAKI, Jun'ichi TATEMURA, Hanpei KOIKE, Hidehiko TANAKA
The University of Tokyo

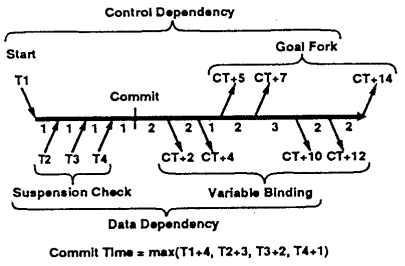

図 1: Fleng のゴール・リダクション

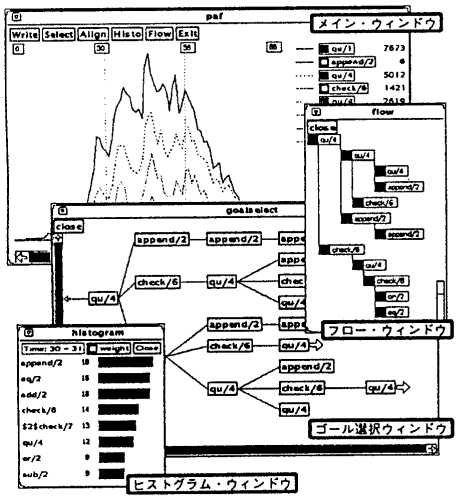

図 2: Paf のディスプレイ

している。チェックの結果コミットされると (CT)、ボディの実行を開始し、ゴールの生成と変数の束縛がおこなわれる。

各ゴールの生成された時刻からリダクション終了まで時間をカウントし、ボディ実行におけるゴールの生成および変数の束縛について、その時刻を記録する。仮想時刻は、実際の処理系で処理された時刻とは無関係になる。

4.3 Paf のディスプレイ

Paf は、各ゴールについて以上のように計算されたトレース・データを視覚化する。

Paf のディスプレイは、メイン・ウィンドウ、ゴール選択ウィンドウ、フロー・ウィンドウ、ヒストグラム・ウィンドウで構成される (図 2)。

メイン・ウィンドウ Paf を起動すると最初に表示されるウィンドウで、並列度の表示をおこなう 。メイン・ウィンドウは、並列度表示部、表示選択部、メニュー部で構成される。並列度表示部には全体および sub tree の並列度が表示される。表示選択部では、後述するゴール選択ウィンドウで選択されたゴール以下の sub tree の並列度を、表示するか否かの指定をおこなう。また、そのゴール以下の処理時間 (並列度表示部の面積に相当する) を表示する。

並列度表示により全体のおおまかな挙動を観察でき、また sub tree の並列度表示によりプログラムの部分的な挙動を知ることができるので、プログラムの各部分の動作の関連を発見することができる。

ゴール選択ウィンドウ ゴール選択ウィンドウは、ゴール・リダクションの tree を表示する。ウィンドウにはゴールが tree 状に表示され、これをたどることで並列度を表示させたいゴールに行きつくことができる。

フロー・ウィンドウ ゴール・リダクションの様子を、時刻を反映した形式で表示する 。並列度の低下している部分での挙動を細かく視覚化することができ、ボトル・ネックとなっている部分の発見に有用である。

ヒストグラム・ウィンドウ 指定した範囲でリダクションされたゴールの数やリダクションに要した時間を表示する。ある時間内で多く実行されたゴール・リダクションを知ることができ、パフォーマンス改善の方針をたてるのに有用である。

5 適用例

我々の研究室では Fleng で記述された比較的大規模なアプリケーションとして非単調推論システムを試作している。この推論システムの改良に paf を使用した。

この推論システムは、プロダクション・システムと ATMS (Assumption-Based Truth Maintenance System) で構成されている。しかしプロダクション・システム部は、「照合→競合解消→実行」というサイクルで動作するため、あまり並列度が高くならないということが予想されていた。

そこで paf を使って、推論システムのパフォーマンスを調べてみると、並列度が脈動し間欠的にピークが現れるのが観察された。また、競合解消部とルール実行部との関係をグラフ化し、実行部の処理は競合解消部の処理を待っていることを観察でき、競合解消部がパフォーマンスを低下させている原因となっていることが確認できた。このままでは、照合部や ATMS 部の並列性を高めた効果が薄いので、競合解消をおこなわずにルールの発火をおこなう、並列発火というメカニズムを導入した。その結果、全体の並列度が上昇し、実行終了時刻が早くなった。

6 おわりに

本稿では、細粒度高並列プログラミング言語 Fleng のパフォーマンス・デバッギング・ツール paf の設計と実装、およびその適用例について述べた。

今後の課題としてデータ・フローの視覚化を検討している。

参考文献

[NT86] Martin Nilsson and Hidehiko Tanaka. FLENG Prolog – turning supercomputers into Prolog machines. In *Proceedings of Logic Programming Conference '86*, Tokyo, June 1986.

[SIR92] 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, August 1992.