1K−7

並列推論マシンPIM/kの開発(1)
— KL1処理系のデバッグ手法とツール —

沖 健治, 丹野 尚, 酒井 浩†, 仲瀬 明彦†, 武脇 敏晃‡
東芝ソフトウェアエンジニアリング(株)　†(株)東芝 総合研究所　‡情報通信システム技術研究所

1 はじめに

並列推論マシンPIMでは, 並列論理型言語KL1を複数の要素プロセッサで並列実行する処理系の研究開発が進められている. 本稿では, メモリ共有型の並列推論マシンのKL1処理系のデバッグの問題点とその支援方法を検討し, 次にPIM/kにおける実現方法について述べる.

なお, 本研究は通産省第五世代コンピュータプロジェクトの一環として行なった.

2 デバッグ上の問題点とデバッグ支援方法

(1) 並列性

並列処理の場合, プロセッサ間で, データの共有や待合せがあり, シーケンシャルにデバッグしにくい. また, 例えば, データ表示ルーチンをつけるとバグがでなくなることがある.

対策

(1-a) 要素プロセッサの連続実行とステップ実行を, 任意に組み合わせた場合にも再現性が保証されるようにする.

(1-b) データチェックや表示ルーチン等をつけた場合にも再現性が保証されるようにする.

(1-c) 疑似並列のスケジューリングには自由度を持たせることで, 処理のタイミングに関わるバグを見つけることができるようにする.

(2) ガーベジコレクション

KL1処理系では, 実時間GCと一括GCをおこなうので, GC自身のバグと通常のKL1のデータ操作のバグが絡み合って原因が見つけにくい.

対策

(2-a) 実時間GC自身のバグであるかどうか調べるには, GCをしないをモードでも処理系が動くことができるようにする.

(2-b) 一括GC自身のバグであるかどうか調べるには, コピーイングGCを使用するので, GCの直後に旧領域と新領域についてチェックするのが有効である.

(2-c) 通常のKL1のデータ操作のバグを調べるには, おかしいと思われるデータにアクセスする履歴がとれるようにし, 再実行により, エラーが起きる少し手前のデータアクセスまで戻せるようにする. 実時間GCを, offモードで実行することは, この場合にも役に立つ.

(3) 種々レベルでのデバッグ

KL1-cプログラムのバグ, KL1コンパイルパス上のバグ, KL1処理系のサブルーチンのバグがあり, アセンブラレベルのデバッグ手段だけでは不十分である.

対策

(3-a) マルチウィンドウで各レベルのソースプログラムとオブジェクトプログラムの対応がとれるようにすると良い. また, 各レベルでのステップ実行が必要である.

(3-b) KL1処理系全体の流れを知るため, 例えば, ゴールの実行履歴をとる必要がある.

(4) 大規模プログラムのデバッグ

ＰＩＭＯＳのように大きなプログラムでデバッグする時には, 常に最初から実行したのでは時間が掛かり過ぎる.

対策

(4-a) 処理中の任意の時点でメモリダンプやレジスタダンプをとれるようにし, その時点から再実行できるようにする.

(5) パフォーマンスバグ

負荷分散などの処理方式およびコーディング上のパフォーマンスバグをとる必要がある.

対策

(5-a) 処理方式上のパフォーマンスバグについては, KL1レベルの統計情報を収集する.

(5-b) コーディング上のパフォーマンスバグについては, アドレストレースをとり, 担当者が最適化の余地について検討する. [1] また, アドレストレースはソースと対応づけるようにする.

3 PIM/kにおけるデバッグ方法

KL1処理系のデバッグは, PIM/kクラスタのソフトウェアシミュレータの上で, KL1処理系, 要素プロセッサの機械語命令にコンパイルしたKL1プログラム, デバッ

Development of Parallel Inference Machine PIM/k (1) – Debugging Technique of KL1 System –
Kenji OKI, Hisashi TANNO, Toshiba Software Engineering Co. Ltd.,
Hiroshi SAKAI, Akihiko NAKASE, Toshiaki TAKEWAKI, Toshiba Corporation

グ支援プログラムを動かして行った.次にシミュレータ,デバッグ支援プログラムの概要を述べる.

3.1 PIM/k クラスタ・シミュレータ

本シミュレータは,KL1 処理系のデバッグ及びハードウェアを含むシステム全体の評価を目指したもので,現在 AS-3000/ 4000 上で稼働している.KL1 処理系デバッグ用のため,下記の特徴を有する.なお,付記した番号は,前節のどの対策と対応するかを示している.

(1) 要素プロセッサのスケジューリング

複数要素プロセッサによる並列処理のシミュレートでは,i 番目のプロセッサが n_i 個の命令を実行すると次のプロセッサに実行順序が移るようにした.また,ひとつの要素プロセッサに着目してステップ実行する場合も,全く同じ結果が得られるように,そのプロセッサ i が n_i 個の命令を実行した時点で他のプロセッサが指定された個数の命令を実行するようにしている.(1-a, 1-c)

ここで,各要素プロセッサに何命令実行したかを示すカウンタ(以下,命令実行数カウンタ)を設け,この値により,処理の時点を特定できるようにした.(2-c, 4-a)

(2) ブレークポイント機能の多様性

命令実行に関しては,アドレスだけでなく命令種別でも実行が中断できるようにした.例えばプロセッサ i の命令実行数カウンタが N を越えた後に proceed/ execute 命令を実行した時点で止めることができる.(3-a)

メモリアクセスに関しても,アドレスだけでなくデータの値についても条件指定できるようにしたため,例えばリダクション数がある値となった時点で止めることができる.(2-c)

(3) 局所的トレース機能

要素プロセッサが,ある特定のメモリアドレスをアクセスするごとに,プロセッサ番号,命令実行数カウンタの値,データの値を記録できるようにした.これにより,実時間 GC に絡むバグのように,メモリの不正な書換えの検出が容易になる.(2-c)

(4) デバッグ用特殊命令の追加

各要素プロセッサに命令実行数のカウントの抑止 / 解除を制御する特殊命令を追加した.これにより,データの表示やチェックルーチン,KL1 処理系に関わる実行履歴をとるルーチン等を,通常の処理の流れを乱すことなく追加することができる.なお,特に実行命令数が多いルーチンについては,同等の機能を有する特殊命令とすることで高速化をはかっている.(1-b, 2-a, 2-b, 3-b, 5-a)

(5) 性能測定用機能

各要素プロセッサで下記の命令の実行回数を表示することにより,常に性能を意識するようにした.

全命令,ロード命令,ストア命令,分岐命令(条件の成立 / 不成立を区別)

また,KL1 処理系の性能向上を支援するため,各ルーチンごとのダイナミックステップ数や実行トレースを採取できる.(5-b)

(6) ソースプログラムとの照応機能

UNIX の C 言語デバッガの dbx のように,オブジェクトプログラムファイルに stab 情報を持たせることで,スクリーンエディタ上に表示されているソースプログラムの行の単位で実行できるようにした.なお,KL1/C, KL1/B, 要素プロセッサのアセンブリ言語での対応関係を区別するため,stab 情報の識別コードを追加した.(3-a)

(7) システムダンプ

ある時点のメモリダンプとレジスタダンプ及びデバッグ環境の保存することにより,その時点からの再開できるようにした.(4-a)

3.2 デバッグ支援プログラム

本プログラムは,PIM/k クラスタ・シミュレータ上で動作し,KL1 処理系のインタラクティブなデバッグ環境を提供する.

主な機能は,次のとおりである.

(1) 初期ゴールの入力

キーボード入力された文字列から,ゴールレコードを生成し,指定されたプロセッサのゴールキューに繋ぐことができる.また,その際,アトムや変数の管理テーブルのメンテナンスを行う.

(2) KL1 処理系固有データの操作

処理の途中で,KL1 処理系固有のデータ(ゴールキュー,サスペンドスタックなど)を参照・変更できる.

(3) プログラムのインクリメンタルロード

分割コンパイルされた応用プログラムを順次ロードできるので,デバッグの準備が容易になる.

4 おわりに

メモリ共有型の並列推論マシンの KL1 処理系のデバッグの問題点とその支援方法および PIM/k における実現方法について述べた.ここで述べた方法により,これまでに 8 プロセッサで N クイーン程度のプログラムを使い,KL1 処理系のデバッグを行うことができた.今後は,PIMOS のように大規模な KL1 プログラムのデバッグについて,検討する必要がある.

参考文献

[1] 丹野ほか,"並列推論マシン PIM/k の開発 (2)-KL1 処理系の予備評価 -",第 42 回情報処理全国大会講演論文集