細粒度並列計算機お茶の水1号 － 最適化コンパイラ －

6G－3

稲垣 達氏　松本 尚　平木 敬

東京大学理学部情報科学科

1 はじめに

命令レベルの並列性を利用した細粒度並列処理を行なう場合、演算器の高速化に伴い細粒度での同期や通信のコストが相対的に大きくなる。これらに対してハードウェアによる高速なバリア同期機構 [3] やプロセッサ間通信機構 [2] を用いて同期や通信のオーバーヘッドを削減もしくは隠蔽することができる。その際、静的なスケジューリングによって同期命令の発行やデータ転送のタイミングの最適化を行なうことが重要である。

本稿ではプログラムの基本ブロック及びトレース内の演算レベルの並列性を静的タスクスケジューリングによって利用する細粒度並列処理を実現する最適化コンパイラ OP.1(Optimizing Parallelizer 1) について述べる。OP.1 は当研究室で開発中の汎用細粒度並列計算機お茶の水1号をターゲットとしており、本稿ではシミュレータ上で性能評価を行なった結果を示す。

2 コンパイラの概要

コンパイルの過程は次の六つに分けられる。

1. ソースプログラムの構文解析
2. 中間言語の生成
3. フローグラフの生成
4. タスクスケジューリング
5. elastic barrier 同期コードの挿入
6. レジスタ割り付け

ソースプログラムは簡単な手続き型言語で、構造として配列を持つ。中間言語はシングルプロセッサでのアセンブリ言語に相当し、これを等価なフローグラフに変換し、固定台数でタスクスケジューリングを行なう。通信命令はこの段階で全て配置されるが、同期命令は後のステージでクラスタ化や簡約化を行ない、スケジューリングされた gantt chart に挿入する。最後に各プロセッサ内で変数の生存期間解析を行ない、レジスタを割り付ける。

An Optimizing Compiler for a General Purpose Fine-Grained Parallel Processor OCHANOMIZ-1
Tatsushi INAGAKI, Takashi MATSUMOTO, Kei HIRAKI
Department of Information Science, Faculty of Science, the University of Tokyo
E-mail: {inagaki, tm, hiraki}@is.s.u-tokyo.ac.jp

3 タスクスケジューリング

フローグラフをプロセッサにマップするタスクスケジューリングアルゴリズムとしては B. Kruatrachue の DSH (Duplication Scheduling Heuristic)[1] を用いる。[1] では代表的なスケジューリングアルゴリズムである CP/MISF 法において、通信の遅延のために生じた空きスロットに他の命令を挿入するスケジューリングポリシーを ISH (Insertion Scheduling Heuristic) と呼んでいるが、これはそのままプロセッサ内のパイプラインのスケジューリングにナイーブに導入することができる。DSH は ISH にさらに先行タスクの複製による最適化を付加したアルゴリズムである。DSH を用いることによって通信のコストを考慮したタスクスケジューリングを行なうことができ、同時に局所性の拡大による全体の高速化が可能になる。

また、スケジューリングの対象となるタスクグラフの DAG(Directed Acyclic Graph) を大きくするため、VLIW 計算機のコンパイラ等で採用されている trace scheduling を援用する。

4 同期命令の挿入

各シュレッドにおける同期命令を決定するには次の三つの段階を経る。

まず、タスク間の一対一の先行関係をバリア同期と整合性の良い多対多の先行関係にクラスタ化する。具体的には、互いの先行タスクの開始時刻が後継タスクの開始時刻を越えない先行関係を一つのクラスタにまとめるという操作を繰り返す。

次に、先行関係で依存距離が長いものを、冗長であれば除去する。

最後に、生成された先行関係のクラスタに対応するバリア同期命令を決定し、命令を挿入する。後継タスクを持つシュレッドは pre request 命令と real request 命令、後継タスクを持たないシュレッドは approve 命令を発行する。

5 性能評価

OP.1 の生成したコードの性能をシミュレータ上で測定した結果を以下に示す。シミュレータは [4] で用いたものを使用した。シミュレータ細部のコストは、バリア命令の実行にプロセッサのパイプラインクロックで 2 clock、

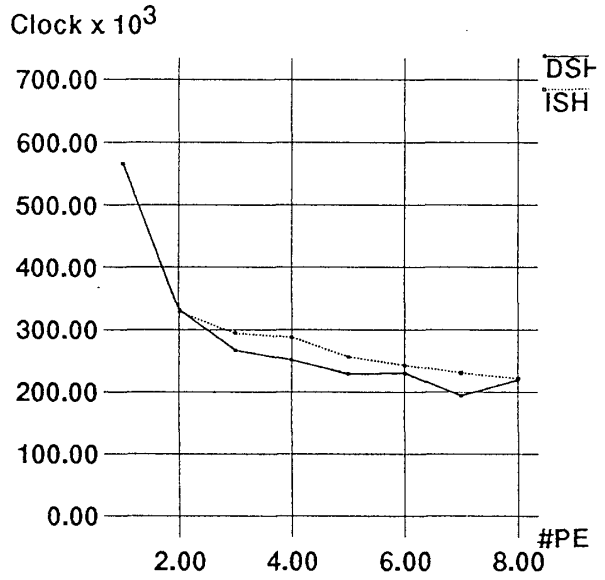

図 1: 例題 1 の実行結果

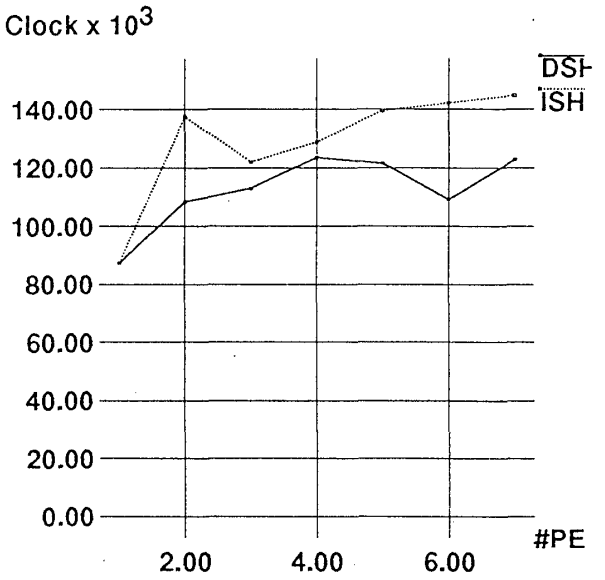

図 2: 例題 2 の実行結果

スヌープキャッシュの update プロトコルを用いてプロセッサ間通信を行なう際の遅延が 6 clock である。要素プロセッサのパイプライン構成は MIPS 社の R4000 に準拠している。

5.1 例題 1

例題 1 は [5] で用いられている級数計算を行なうプログラムである。ループ内の演算レベルの並列性が高く、プロセッサ内のパイプラインコンフリクトを起こしやすい浮動小数点の除算命令がループボディ内に数多く含まれているため、8 台で 1 台のときの約 2.5 倍の高速化を達成している (図 1)。[5] のグラフよりも少ない台数で飽和している理由は、今回シミュレーションに用いたシステムでは要素プロセッサのパイプラインによる細粒度並列性が存在し、システム全体の並列性が高いためであると考えられる。

5.2 例題 2

例題 2 は [2] で使用されている誤差拡散法のプログラムの一部である。このプログラムはループ内の演算レベルの並列性が低く、critical path に loop carried な依存があり並列化が難しい。[2] ではパイプラインステージのオーバーラップができない CISC 計算機を用いていたため、アンローリングなしで 1.67 倍の高速化を達成している。しかし、今回のシミュレーションでは演算レベルの並列性が殆んど 1 台分のパイプラインに吸収されてしまい、逆にプロセッサ台数が増えるにつれて通信や挿入された同期命令のオーバーヘッドが現れ、1 台の時より性能が悪くなっている (図 2)。

6 おわりに

現段階のコンパイラでは、トレース内の並列性が低い場合に負の台数効果を示す例があり、さらなるアルゴリズムの改善が必要である。今後、お茶の水 1 号上にコンパイラを実装し、性能を評価する予定である。

参考文献

[1] Kruatrachue, B., *Static Task Scheduling and Grain Packing in Parallel Processing Systems.* PhD thesis, Electrical and Computer Eng. Dept., Oregon State Univ., Corvallis, 1987.

[2] 松本 尚, "細粒度並列実行支援マルチプロセッサの検討," 情報処理学会論文誌, vol. 31, no. 12, pp. 1840–1851, Dec. 1990.

[3] 松本 尚, "Elastic Barrier: 一般化されたバリア型同期機構," 情報処理学会論文誌, vol. 32, no. 7, pp. 886–896, July 1991.

[4] 松本 尚, "スヌープキャッシュ制御機構の DOACROSS ループへの適用," 並列処理シンポジウム JSPP'92 論文集, pp. 297–304, June 1992.

[5] 尾形 航, 吉田 明正, 合田 憲人, 岡本 雅巳, 笠原 博徳, "スタティックスケジューリングを用いたマルチプロセッサシステム上の無同期細粒度並列処理," 並列処理シンポジウム JSPP '93 論文集, pp. 111–118, May 1993.