4V－5　既存並列計算機を対象とした細粒度並列仮想マシンコードDVMCからのコードスケジューリング

山下欣宏　日下部茂　谷口倫一郎　雨宮真人
九州大学大学院 総合理工学研究科

1　はじめに

我々は，関数型言語をベースにした超並列V言語のコンパイラの開発を行っている[1]．コンパイラは大きく分けて機種非依存／依存の2つの部分で構成されている．コンパイラの機種非依存部では，まずプログラムをデータ依存解析して細粒度並列中間言語コードDVMCを生成し，機種非依存の範囲で最適化を行う．機種依存部では，各機種に依存するコストパラメータを受け取り，各対象計算機で効率的に実行できるコードを生成する．本稿では，特に分散メモリ型計算機AP1000を対象とした機種依存部のコードスケジューリング法について述べ，生成されるコードを評価する．

2　Vコンパイラの概要

2.1　機種非依存部

プログラムをデータ依存解析して生成されるDVMCに機種非依存な範囲で構造データのコピー除去，冗長アークの除去，および冗長に分割されたスレッドの併合などの最適化を施す．想定する抽象実行モデルでは，並行実行可能な複数のスレッドから構成される関数インスタンスを並列実行する多重実行環境によりプロセス間通信やリモートメモリアクセス等の遅延の隠蔽を行う．また，関数呼び出しは引数が全て揃わなくても揃った部分から実行できる方式をとっている．

2.2　機種依存部

機種依存部では，機種非依存の最適化を施されたDVMCと機種依存のコストパラメータを入力とし，3節で示すように並列性の調整を行う．コストパラメータよりインスタンスへの引数渡しメッセージの結合判定，インスタンスのローカル/リモート呼び出し判定を行うことでターゲットマシンで効率的に実行されるスレッド化コードを生成する．

3　並列性の調整

AP1000を対象とした場合，スレッド並列処理のための特別なハードウェア支援はなくスレッド並列処理をソフトウェアで実現している．そのため，メッセージ操作，同期処理，スレッド切替え処理などの並列処理のコストが高く実行効率は落ちる．しかし逆に，スレッドの実行順序などの制御が容易に行えるため，スケジューリングにより効率を上げることが可能である．3節で示す手法により，通信回数削減によるメッセージ操作コストの低減，スレッドの実行順序を一部決定することによる同期処理数の削減とスレッド長の増加および計算の局所性の活用による実行速度の向上を狙っている．以下にスケジューリングの際の前提と定義およびスレッドスケジューリングアルゴリズムを示す．

【前提および定義】

本手法は，同一インスタンスを構成するスレッドは同一PEで実行される，という前提で適用できる．DVMCではインスタンスへの引数渡しメッセージは未結合，インスタンス生成は全てsplit-phaseでリモート呼び出しの形をとっている．インスタンスの深さ $idep$ を，(親インスタンスの深さ+1) と定義する．ただし，一番浅いインスタンス (メイン関数) の深さは0とする．また $INS(idep), THR(ins)$ を以下のように定義する．

- $INS(idep) = \{$ 深さが $idep$ であるインスタンス $\}$
- $THR(ins) = \{$ インスタンス ins を構成するスレッド $\}$

【スレッドスケジューリングアルゴリズム】

1. 初期化
   DVMCよりインスタンス間の依存関係を含んだグラフ (スレッドグラフと呼ぶ) を作成する．スレッドグラフのノードはスレッドを表し，アークはスレッド間の依存関係を表す．次にコストパラメータより各スレッドのコストと各アークのコストを算出する．$idep := Max(idep)$ として以下のアルゴリズムを適用する．
2. インスタンスへの引数渡しメッセージの結合判定
   メッセージ結合時における引数渡し部分のメッセージ操作コスト，通信コスト，同期処理コスト，スレッド切替えコストの総和が未結合時のコストの総和より小さい時にインスタンスへの引数渡しメッセージを結合する．
3. 子インスタンスのローカル/リモート呼び出し判定
   深さ $idep := idep - 1$ のインスタンス $\forall ins \in INS(idep)$ について $T := THR(ins)$ とし3(a)以降を実行する．
   (a) インスタンス処理時間の算出
   スレッド集合 T と T から呼び出されるインスタンスより，インスタンス ins の処理時間を求める．
   (b) ローカル呼び出しの判定
   if (インスタンス ins からローカル呼び出しをしてもインスタンス ins の実行時間が長くならない子インスタンス ins_{local} が存在する)
   then　3(c) を実行．
   else　インスタンス ins について3 終了．
   (c) スレッドグラフの変形
   i. ins_{local} をローカル呼び出しとする．
   3(b) で複数の子インスタンスをローカル呼び出しに出来るときは，インスタンスinsが返値を早く要求しているものをローカル呼び出しとする．それも複数存在する時は処理時間の長いものをローカル呼び出しとする．
   ii. 変形によって実行順序が決定し，インスタンス ins_{local} からのアークで冗長なアークになるときがある．その冗長なアークを除去する．このときスレッドが併合されることがある．

Code Scheduling Strategy of Fine-grain Parallel Virtual Machine Code, DVMC, for stock parallel machines
Yoshihiro YAMASHITA, Shigeru KUSAKABE, Rin-ichiro TANIGUCHI, Makoto AMAMIYA
Graduate School of Engineering Sciences, Kyushu University

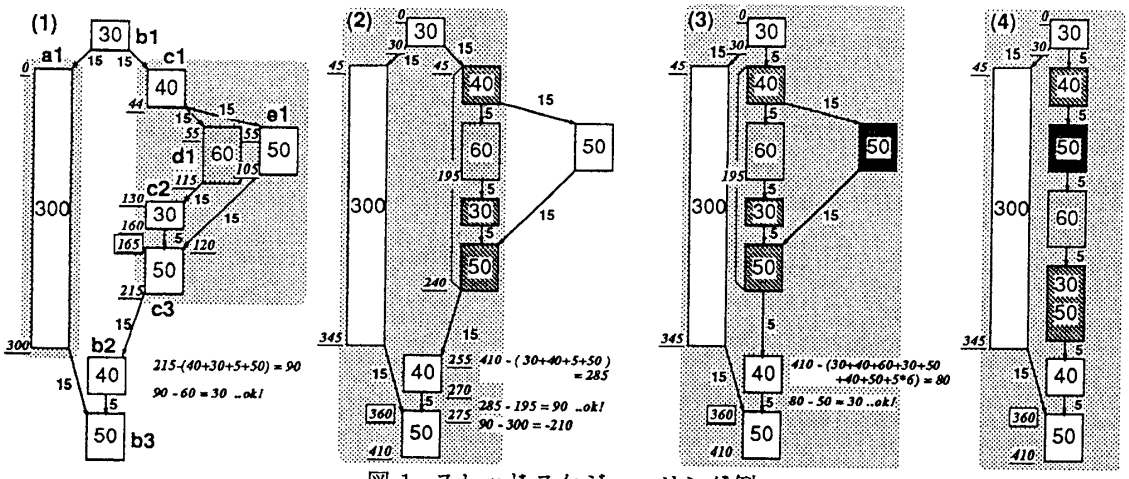

図 1: スレッドスケジューリング例

iii. $T := T \cup THR(ins_{local})$ とし，3(a) から繰り返す．

4. if ($idep \geq 0$) then 3. から繰り返す．
   else スレッドスケジューリング終了．

step3以降での簡単なスレッドグラフのスレッドスケジューリング例を図1に示す．実際のコードには図には描かれていないメッセージを受信するスレッドが存在する．この例では，プログラムは A, B, C, D, E の 5 つのインスタンスで構成されており，$THR(A) = \{a1\}$, $THR(B) = \{b1, b2, b3\}$, $THR(C) = \{c1, c2, c3\}$, $THR(D) = \{d1\}$, $THR(E) = \{e1\}$ である．このグラフでは $Max(idep) = 2$ であり，$idep = 1$ でのインスタンス A と C について，それぞれstep3(a) を適用する．A から呼び出されるインスタンスは存在しないので変形されないが，C は D, E を呼び出しているのでローカル呼び出しの判定を行う．この例では，D, Eのどちらをローカルに呼び出しても C の処理は遅くならないがスレッド $c2$ が $c3$ に先行しているため Dをローカル呼び出しとする．スレッドグラフを変形した後 (図 1(2))，次に $idep = 0$ の Bについて同様にstep3以下を適用する．このとき，Cが Bからローカルに呼ばれるように変形され (図 1(3))，Eは step3(c)iii より Cからのローカル呼び出しの判定の対象となる．実際，Eもローカルに呼び出されることとなり，それにより Eからのアークが冗長となり除去され，これによって同期の必要がなくなったスレッド $c3$ はスレッド $c2$ と併合される (図 1(4))．

4 評価および考察

PE数 64 台の AP1000 上で N-Queen パズルの全解探索プログラム ($N = 9$) について本スケジューリング方式を適用した結果を表1に示す．O0 はDVMC であり，O1 は 3節 *Algorithm*2インスタンスへの引数渡しメッセージの結合判定まで，O2 はstep 全てを適用したものである．AP1000 はメッセージ長がそれほど長くない場合，メッセージ操作コストがメッセージ長にほとんど影響されず，また他の並列処理コストに比べメッセージ操作コストが大きいため，O1 では全てのインスタンスへの引数渡しのメッセージが結合されメッセージ数が減少している．またその結合により呼ばれ側のスレッドが併合されて同期処理数，スレッド切替え数が減少し実行時間が短縮されている．O2 ではかなりのインスタンス生成がローカル呼び出しとなり，メッセージ数が減少し実行時間が短縮されている．

	実行時間 (s)	メッセージ数	スレッド切替え数	同期処理数
O0	2.421285	2,377,030	3,445,499	2,007,094
O1	1.048923	719,202	1,078,803	136,714
O2	0.445617	136,714	936,337	136,714

表 1: 実行結果

関連研究に *North Carolina State* 大学の Threshold Scheduling(TS)[2] がある．TS はタスクグラフを用いて静的にタスクをスケジューリングする手法である．タスクグラフのノードであるタスクは，どのPE でも実行可能となっている．よって計算量爆発を防ぐためタスクのスケジューリングは，着目するタスク以降に生成されるタスクの情報を用いずヒューリスティックによりスケジューリングしている．これに対し本方式は環境管理コスト削減と計算の局所性の活用のため同一インスタンス内のスレッドは同一PE での実行としている．このため生じるスレッドグラフの制約を利用し，スケジューリング区間を決めボトムアップに下位の情報を考慮したスケジューリングが可能となり，計算量爆発を防ぎ効率良く実行されるコードを生成することができる．

5 おわりに

本稿では，インスタンスへの引数渡しメッセージの結合判定，インスタンスのローカル/リモート呼び出し判定を行うコードスケジューリングにより並列性の調整を行う手法を示した．この手法を適用することにより効率良く実行されるコードを生成することを示した．

今後は，PE 間の距離情報を考慮したコードスケジューリング法の検討を行う．

参考文献

[1] 日下部, 他:“超並列V言語とその実行方式”, 並列処理シンポジウム JSPP'94 論文集, pp.41-48, May 1994.

[2] S.S.Pande, et al.: “Threshold Scheduling Strategy for Sisal on Distributed Memory Machines”, J. of Parallel and Distributed Computing Vol 21,pp.223-236 (1994).