手続き間自動並列化コンパイラ WPP の評価

佐藤　真琴[1),2)]、青木　雄一郎[1),2)]、和田　清美[1),2)]、太田　寛[1),3)]、飯塚　孝好[1),2)]
1) アドバンスト並列化コンパイラ研究体、　2)（株）日立製作所システム開発研究所
3)（株）日立製作所情報コンピュータグループ
(msatoh, yu-aoki, kiyomi, iitsuka) @sdl.hitachi.co.jp, hir-ohta@itg.hitachi.co.jp

概要

本論文では、手続き間自動並列化コンパイラ WPP の特徴と評価結果を述べる。WPP の特徴は、手続きクローニングにより促進された手続き間定数伝播、連立線形不等式を用いた配列参照リージョン表現、コモン変数に対する選択的プライベート化、及び OpenMP プログラムと日立 SR8000 向けネィティブコードの生成である。代表的ベンチマークである SPECfp95 と NPB のいくつかのプログラムに対して、生成 OpenMP プログラム、ネィティブコード、及び MPI プログラムを SGI™ 社の並列計算機 Origin™ 2000 と日立 SR8000 上で評価した。生成 OpenMP プログラムはネィティブコードや MPI プログラムとほぼ同等の台数効果であった。

Evaluation of Interprocedural Parallelizing Compiler WPP

Makoto SATOH [1),2)], Yuichiro AOKI [1),2)], Kiyomi WADA [1),2)], Hiroshi OHTA [1),3)], and Takayoshi IITSUKA [1),2)]
1) Advanced Parallelizing Compiler Project,　2) Systems Development Laboratory, Hitachi, Ltd.
3) Information and Computer Systems, Hitachi, Ltd.

Abstract

In this paper, the features of an interprocedural parallelizing compiler WPP and its evaluation results are described. Its features are interprocedural constant propagation enhanced by procedure cloning, the representation of array reference regions using the linear inequality system, selective privatization for COMMON variables, and OpenMP or Hitachi SR8000 native code generation. For some programs from SPECfp95 and NPB, the evaluation results of generated OpenMP programs, native codes, and MPI programs are compared on the SGI™ Origin™ 2000 and the Hitachi SR8000. The OpenMP programs performed almost the same speedups as native codes and MPI programs.

1．はじめに

OpenMP は共有メモリ型計算機（SMP）向けの指示文などから成るプラットフォームフリーの API である。この API はユーザにとって理解が容易である反面、プログラムを解析して適切な OpenMP 指示文を挿入するのは困難でエラーを引き起こし易い。実際、実行効率の高い OpenMP プログラムを得たいユーザは、彼らのプログラムを注意深く解析し、変数に対する複雑な属性指示やバリア同期の挿入などを行なう必要がある。さらに、並列化を促進するために、ユーザはしばしばプログラムを変換する必要もある。

この問題を解決するために、我々は、手続き間自動並列化コンパイラ WPP（Whole Program Parallelizer）[1, 2, 3]を開発している。これは、逐次 Fortran プログラムを入力し、手続き間解析・並列化およびループ変換を施し、その結果を OpenMP プログラムまたは日立 SR8000 向けネイティブコードとして生成する。手続き間解析・並列化の特徴として、手続きクローニングにより促進された手続き間定数伝播、連立線形不等式を用いた配列参照リージョン表現、コモン変数に対する選択的プライベート化が挙げられる。このうち、手続き間定数伝播は、プログラムを単純化し、並列化解析の解析精度を向上するのに有効である。また、連立線形不等式表現は、複雑な添字式を表

現可能なだけでなく、手続き呼出し前後で配列次元が変化する添字でもうまく表現できる。

手続き間並列化コンパイラに関する関連研究として、以下がある。

KAP™ [4]は手続き間並列化の結果をOpenMPプログラムとして出力できる。その手続き間解析方法は、手続きをインライン展開した後に通常の手続き内解析を行なうものであり、OpenMPプログラムを生成する時は、その解析結果をインライン展開しないプログラムに挿入する。

Polaris/OpenMP [5, 6]もインライン展開を行なって解析し、OpenMPプログラムを生成する。

SUIF [7]は、我々と同様な、インライン展開を行なわない手続き間解析を行なうが、OpenMPプログラムは生成しないようである。

本論文は以下のように構成されている: 第2章にはWPPの特徴を記述する。第3章ではWPPの評価を述べる。第4章では本研究の結論を述べる。

2. WPP自動並列化コンパイラ

図1はWPPの構造である。WPPは3フェーズから構成される：手続き間（以降、IPと書く）スカラ解析、IP配列解析、IP並列化の3つである。また、コールグラフ、クローン手続き、制御フローサマリグラフを作成する共通フレームワークも持つ。

表1:スカラ変数に対する参照集合.

参照集合	意味
MOD	定義の可能性のある変数
KILL	確実に定義される変数
USE	使用の可能性のある変数
EUSE	定義前に使用の可能性の有る変数
LIVE	ループや手続きの最後でライブな変数

2.1 手続き間スカラ解析

IP スカラ解析は、各手続きに対して、各変数の参照集合－MOD, KILL, USE, EUSE, LIVE－を決定する。各集合は、各手続きにおける各変数の定義・使用情報を要約したものである。表1に意味を示す。

IP スカラ解析は、手続きクローニングと手続き間定数伝播を行ない、プログラムを単純化することも行う。その結果、次に行なう手続き間配列解析の精度が向上が見込める。図2はこの最適化の適用例である。手続きfに手続きクローニングが適用され、その手続きの引数mに定数伝播が適用され、m, k, lの値がコンパイル時に計算される。

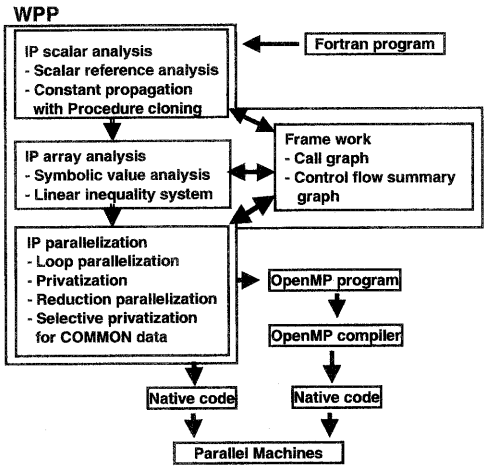


図1:WPPの構造.

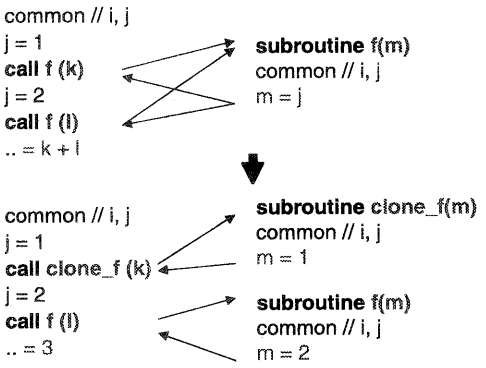


図2:手続きクローニングと手続き間定数伝播.

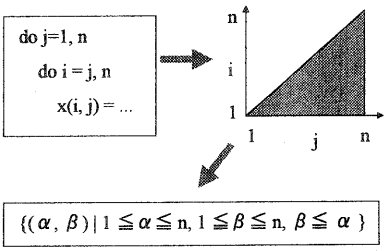


図3:連立線形不等式を用いた参照リージョン表現.

表2:参照リージョンのイタレーション種別.

種別	意味
CUR	i 回目の繰り返し回でのリージョン
PREV	1から (i-1)回目までのリージョンの和集合
ALL	全繰り返し回のリージョンの和集合

2.2 手続き間配列解析

IP 配列解析は、各手続きやループに対して、各配列の参照添字の集合ー参照リージョンーを計算する。各参照リージョンは表1の参照集合と同じで、各参照リージョンは、各手続きや各ループにおける各配列要素の定義・使用情報を要約したものである。これは、ループに対しては、ループイタレーションのi回目 (CUR)、1回目から(i-1)回目までの和集合 (PREV)、全イタレーション回数の和集合 (ALL) に分けて計算される (表2)。

参照リージョンの表現には連立線形不等式を用いる。これは、ストライド参照や三角形状参照のような複雑な参照パターンを表現するのに有用である。図3は三角形状参照の例である。

2.3 手続き間並列化

IP 並列化は、2.1、2.2節の2つの解析で得られた情報を用いて、手続き呼出しを含むループを並列化する。本並列化自身も全手続きを解析し、プライベート化や並列リダクションのような、並列ループ数を増加させるいくつかの手法を適用する。手続き間並列化の特徴を以下にまとめる。

(1) 手続き間変数プライベート化： スカラや配列に対する特定の依存関係を手続き境界を越えて検出し、各スレッドに対してその変数のコピーを作成することで、その依存をなくす。
(2) 保証コード生成： 手続き呼出しを含む並列ループ中のプライベート変数・配列に対して、初期値・終値保証および条件付終値保証コードを挿入する。
(3) コモン変数の選択的プライベート化： 各コモンブロックからプライベート変数を検出し、プライベート変数だけを含む新しいコモンブロックを作成する。これは使用メモリ量削減に有効である。
(4) 手続き間リダクション並列化: SUM, PRODUCT, MAX/MIN, MAXLOC/MINLOC など多種のリダクションを並列化する。

2.4 OpenMP コード生成

WPP の並列化方法には以下の特徴がある。これらは OpenMP と親和性があるので、WPP の並列化中間語から OpenMP コードを生成するのは容易である。

- 並列化対象はループである。
- 並列処理部はマスタースレッドとスレーブスレッドの両方が実行する。
- 並列処理部以外の文はマスタースレッドが逐次実行する。
- スレッド生成・消滅とループ並列化を分離し、複数の並列ループのスレッド生成・消滅が1回で行なえる。

また、ループ内に現れる変数の属性についても、各ループ毎に以下の一つまたは複数に決定する。この内、最後のもの、条件付終値保証、配列プライベート化以外は OpenMP1.0 にのっとったプログラム生成に使うことが可能である。

- 共有
- プライベート
- 初期値・終値・条件付き終値保証の要否
- リダクション
- プライベートコモン：プライベートかつコモン変数で、ループ中に手続き呼び出しを含む。

また、並列化決定部は各並列ループ毎に、以下のいずれかのループスケジューリング方法を決定する。OpenMP 生成では、この内、ブロック分割のみ生成する。

- ブロック分割
- サイクリック分割
- パイプライン並列化

OpenMP コード生成部は、以上の情報から OpenMP 指示文の中間語を作成し、その後に、全中間語をソースプログラムに逆変換する。

3. 性能評価

SPECfp95 と NPB2.3-serial からいくつかのプログラムを取り上げて WPP の性能評価を行なった。特に、SPECfp95 では turb3d、NPB2.3 では EP、MG、および CG に対して詳細な評価を実施した。

評価した WPP の生成コードは、OpenMP プログラムと SR8000 ネイティブコードの両方である。

評価マシンは、SGI ™ 社の Origin ™ 2000 と日立

SR8000 である。Origin 2000 上では OpenMP プログラムを、SR8000 上ではネイティブコードを評価した。

ここで、Origin 2000 は、2CPU から成る共有メモリプロセッサ（SMP）を1ノードとする分散共有メモリマシンである。表3はその性能諸元である。本性能評価では1筐体に収まった 16CPU（8ノード）までを用いた。

一方、SR8000 は8CPU から成る SMP を1ノードとする SMP クラスタである。表4はその性能諸元である。本性能評価では8CPU（1ノード）のみを用いた。

生成 OpenMP プログラムかに対するコンパイラは、Origin 2000 では MIPS Pro Fortran を用いた。コンパイルオプションは

-mp -Ofast=ip27 -OPT:IEEE_arithmetic=3

である。

表3：Origin 2000 の性能諸元.

CPU	MIPS R10000
周波数	195 [MHz]
L1 キャッシュサイズ	32 [KB]
L2 キャッシュサイズ	4 [MB]
メモリサイズ/ノード	704 [MB]
全プロセッサ数	32

表4：SR8000 の性能諸元.

CPU	独自 CPU
周波数	250 [MHz]
L1 キャッシュサイズ	128 [KB]
メモリサイズ/ノード	8 [GB]
全プロセッサ数	8

3.1 OpenMP プログラム

図4は NPB/EP と SPECfp95/turb3d から WPP が生成した OpenMP プログラムを Origin2000 上で評価した結果である。Par と IP は、各々、手続き内並列化と手続き間並列化を適用して生成されたプログラムを表す。

EP

EP には、内側に2つのループと手続き呼出しを含む主要ループが1つあり、実行時間のほとんどはこのループによって占められている。手続き間並列化では、ループ内の手続き呼出しを解析することによって、この主要ループを並列化できるため、ほぼリニアな台数効果を得ることができた。

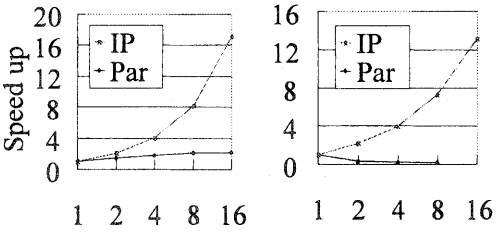


図4：OpenMP プログラムの台数効果.

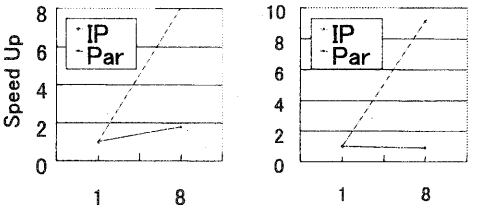
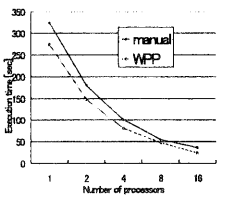


図5：ネイティブコードの台数効果.



SPEC/turb3d
図6：手続き間最適化の効果.

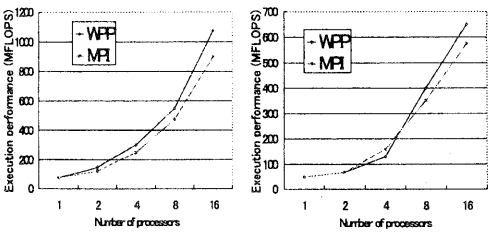


図7：MPI との比較.

一方、手続き内並列では、主要ループは並列化できないものの、主要ループ内にあるループのうち、1つは並列化できる。しかし、台数効果は見こめず、どのプロセッサ台数を見てもほぼ2倍である。これは、主要ループ内にある並列ループの実行時間がプログラム全体の約半分であるためと考えられる。

これを確かめるため、Origin 2000 上でプロファイルを取った。この結果、このループの実行比率はプログラム全体の 57%であることがわかり、上の予想が確認できた。

なお、OpenMP 1.0 では配列リダクションを認めていないので、EP では、1つのリダクション対象配列 q を 10 個のスカラ変数に変更した。これにより、スカラ変数のリダクション並列化が可能となり、手続き呼出しを含む主要ループが、並列化された。

Turb3d

Turb3d には、6つの手続き呼出しのみからなる4つの主要ループがあり、実行時間のほとんどはこれらのループによって占められている。これらの各ループは、ループ毎に同じ手続きを呼び出しているが、各呼出し毎に実引数の定数値が異なる。WPP による手続き間並列化では、同じ手続きでも実引数値が異なるごとに別のクローン手続きを作成することによって、これら実引数値を呼出し先手続きに伝播させ、IF 文の削除、配列添字の単純化をコンパイル時に行なうことができた。更に、連立線形不等式を用いた強力な手続き間配列解析により、サブルーチン turb3d にある4つの主要ループ全てが並列化された。よって、手続き間解析では、ほぼリニアな台数効果を得ることができた。

一方、手続き内並列では、主要ループは並列化できないものの、主要ループ内から呼び出されるいくつかのループは並列化できる。しかし、プロセッサ数が増加するにつれ、性能は悪化した。これは、並列化したループの粒度が小さく、プロセッサ数が増加するにつれ、並列化オーバヘッドが増加したためと考えられる。

3.2 ネイティブコード

図5は、EP と turb3d から WPP が生成したネイティブコードを SR8000 上で評価した結果である。

本評価では EP のソースプログラムに修正を施していないが、配列に対してリダクション並列化が適用されたため、主要ループが並列化された。台数効果の傾向は図4と同様である。また、turb3d についても台数効果の傾向は図4と同様である。図4と図5より、WPP が生成する OpenMP プログラムの実行性能はネイティブコードの性能に匹敵すると言える。

3.3 手続き間最適化

図6は、turb3d から WPP が自動生成した OpenMP プログラム (WPP) とその OpenMP プログラムにおける OpenMP 指示文を元の turb3d プログラムに挿入しただけのプログラム (manual) との Origin 2000 上での実行時間を比較した結果である。1プロセッサから 16 プロセッサまで、すべて WPP が生成した OpenMP プログラムの実行性能が勝っていることが分る。

表5: manual 版と WPP 版との実行時間の比較[sec].

台数	1	2	4	8	16
manual	324.9	179.7	101.3	54.4	36.0
WPP	274.6	148.4	79.8	46.6	23.9
WPP / manual	84.5%	82.6%	78.7%	85.6%	66.5%

表6: manual 版と WPP 版とのプロファイルの比較.

順位	手続き名	サンプル数		差	総和比%
		manual	WPP		
1	dcft	277351	232485	44866	86.5
2	dcopy	12423	12523	-100	90.3
3	lin	7363	7272	91	92.6
4	turb3d	5990	6032	-42	94.5
5	drcft	3812	3862	-50	95.7
総数	—	320787	274171	46616	100

表5は、両者の実行時間とその比率を比較したものである。WPP の生成プログラムの実行時間は manual に較べて、最低で 15.5%、最高では 33.5%高速である。この原因を確かめるために、Origin 2000 の1プロセッサでプロファイルを取った。

表6は、manual 版と WPP 版とのプロファイルの比較結果である。この表では、プロファイルによるサンプル数の多い手続きから順に5つだけ並べている。manual でも WPP でも手続きの順位は同一であった。但し、WPP 版については手続き dcft が6つのクローンに分れているのでそのサンプル数の和を取った。尚、サンプリングは 1[msec]ごとに実施される。また、表6における「総数」はプログラム全体のサンプル数を、「総和比」は、順位1から順にサンプル数を加えたものを総数で割った比を表わす。

表６より、手続き dcft とプログラム全体に対して、manual と WPP のサンプル数の差がほぼ同一で、手続き dcft のサンプル数がプログラム全体の約９割を占めていることから、WPP 版が高速であるのはこの手続きが高速になったためと考えられる。また、手続き dcft のサンプル数の差は全体の 16%で、これはプロファイル非取得時の実行時間の差 15.5%とほぼ同じなので、プロファイルの結果はプロファイル非取得時の実行をほぼ反映している。

次に、手続き間定数伝播の効果か否かを調べるために、手続き dcft 中の変数で WPP が定数化したものを変数に戻して実測した。この結果、実行時間は元のプログラムと同様になった。また、WPP と manual のキャッシュミス回数も 1%の差しかない。よって、現在のところ、WPP 版の高速化の原因は不明である。

3.4 MPI との比較

図７は、NPB2.3-serial から取った MG と CG から WPP が自動生成した OpenMP プログラムと MPI を使って書かれた NPB2.3β の MG と CG との実行時間を Origin 2000 上で比較した結果である。ここで、NPB2.3-serial から取った MG には、MPI のためのプロセッサ間通信のバッファリング処理が含まれているので、これを削除して実行した。両プログラムに対して、WPP 版は MPI 版とほぼ同等な実行性能であることがわかる。

3.5 SPECfp95 に対する台数効果

図８は、SPECfp95 のうち良好な台数効果が得られるプログラムから WPP が自動生成した OpenMP プログラムを Origin 2000 上で性能評価した結果である。swim, hydro2d, mgrid, turb3d は、8 台で 1 台のほぼ 8 倍の性能向上が得られた。このうち、手続き間並列化の効果があったのは、turb3d のみであり、他は手続き内並列化の効果である。

4. 結論

手続き間自動並列化コンパイラ WPP が生成する OpenMP プログラムと日立 SR8000 向きネイティブコードを、代表的ベンチマークである SPECfp95, NPB2.3 を用いて Origin2000 と SR8000 上で評価した。以下の結果が得られた。

- NPB と SPEC の中で手続き間並列化の効果が見込める EP と turb3d に対して、それらの主要ループの自動並列化が可能となり、WPP が生成する OpenMP プログラムとネイティブコードともにほぼリニアな台数効果が得られた。
- NPB の MG と CG から WPP が生成する OpenMP プログラムは NPB2.3β の MPI プログラムとほぼ同等な性能となった。
- turb3d から WPP が生成した OpenMP プログラムは、turb3d に OpenMP 指示文のみ加えたプログラムより、14%から 33%速くなった。

謝辞

本研究の一部は新技術情報処理機構の委託を受けて行なった。

参考文献

[1] Y. Aoki, M. Satoh, T. Iitsuka, S. Satoh, S. Kikuchi, Prototyping of interprocedural parallelizing compiler WPP -performance evaluation-, SIG Notes of IPSJ (in Japanese), 98-ARC-130, pp. 43-48, 1998.

[2] http://www.rwcp.or.jp/trc/kikaku/activities/achievements/MP/hitachi/PDC-hitachi-e.html

[3] Research exhibits and brochures on the corner of RWCP (Real World Computing Partnership) at Supercomputing '99.

[4] Kuck and Associates, Inc., http://www.kai.com/

[5] Polaris/OpenMP, http://polaris.cs.uiuc.edu /

[6] W. Blume, R. Doallo, R. Eigenmann, J. Grout, J. Hoeflinger, T. Lawrence, J. Lee, D. Padua, Y. Paek, B. Pottenger, L. Rauchwerger, P. Tu, "Parallel Programming with Polaris", *IEEE Computer*, pp.78-81, Dec. 1996.

[7] M. Hall, J. Anderson, S. Amarasinghe, B. Murphy, S. -W. Liao, E. Bugnion, M. Lam, "Maximizing Multiprocessor Performance with the SUIF Compiler", *IEEE Computer*, pp.84-89, Dec. 1996.

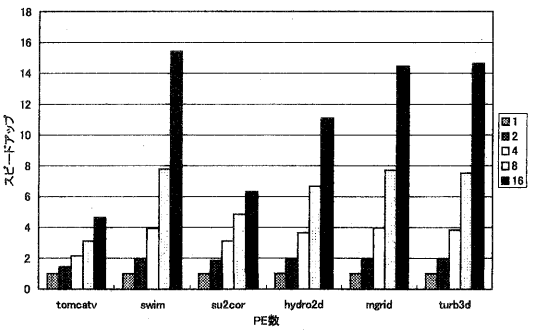


図8: SPECfp95 に対する台数効果