2003－ARC－154　(26)
2003／8／6

対話型 OpenMP プログラム作成支援ツールの開発

石原　誠[†]　本多弘樹[†]
弓場敏嗣[†]　佐藤三久[††]

本稿では対話型 OpenMP プログラム作成支援ツールである **iPat/OMP** の設計指針と実装方法を述べる．iPat/OMP では並列化支援機能を広く利用されているエディタである GNU Emacs に組み込むことで，ユーザは使い慣れたエディタ上でプログラムの編集作業を行いながら，対話的に並列化支援機能を実行することができる．

Development of interactive OpenMP programming assist tool

Makoto Ishihara,[†] Hiroki Honda,[†] Toshitsugu Yuba,[†]
and Mitsuhisa Sato[††]

In this paper, we propose an interactive parallelizing assistance tool for OpenMP, named iPat/OMP. The tool assists the user in transforming a sequential program into parallelized one using OpenMP. The tool is implemented as a set of functions on the Emacs editor. All the activities related to program parallelization, such as selecting a target portion of the program, invoking an assistance command, and modifying the program based on the assistance information shown by the tool, can be handled on the source program editor environment.

1. はじめに

プログラム並列化支援ツールは人手によるプログラム並列化を支援し，プログラマのプログラムの特性に関する知識と自動並列化手法を組み合わせることによりハイパフォーマンスプログラムの作成を実現している．プログラム並列化支援ツールに関しては多くの研究開発がなされている[7)~13)]．それら支援ツールのなかにはプログラムエディタと並列化支援機能を一体化した環境を提供するものがあり，これらではユーザがエディタ上でソースコード編集を行うことと同時に支援機能との対話的な並列化作業を行うことを可能にしている．

近年，逐次プログラムを並列化するための API として，指示子を逐次プログラム中に記述して並列化を命令することができる OpenMP[1)] が注目を集めている．OpenMP 指示子を記述したプログラムは逐次プログラムとしてもコンパイル可能であることや，可搬性が高いという利点を持っている．

人手による OpenMP を用いた並列化の際，プログラマはプログラムの並列化可否判断を行う必要がある．人手によるプログラムの並列性解析も可能であるが，その作業が複雑なため，解析結果に間違いを生じる可能性がある．したがって人手による OpenMP プログラム作成の際，プログラマが正確な並列化判断を行うためにはプログラムの並列性解析を行うツールが必要となる．

本稿では，人手による OpenMP を用いた逐次プログラムの並列化を対話的に支援するツール iPat/OMP について報告する．

iPat/OMP では，並列性解析，並列化の際に挿入する OpenMP 指示子作成，プログラムリストラクチャリングや実行時間解析に関する支援機能を持つこととし，それら機能を広く利用されているエディタに組み込むこととする．これによりユーザが使い慣れたエディタを利用してプログラム編集を行いつつ並列化支援機能との対話的な作業を行うことを可能とする．

また iPat/OMP ではその役割をプログラム解析とその結果提示に限定し，プログラム並列化の判断及び

† 電気通信大学大学院情報システム学研究科
The Graduate School of Information Systems, University of Electro-Communications
†† 筑波大学　電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

プログラム変更はユーザの役割としている．こうすることでユーザにとって読みやすいプログラムを維持することと，並列化対象の適切な選択を行うことを促進する．さらに iPat/OMP は，プログラムの特性に関するユーザの知識を受け入れることを可能とし，その知識を利用した支援が行える機能を持つこととする．

本論文では，まず 2 章で提案ツールの機能と設計指針を述べ，3 章では OmniOpenMP コンパイラが持つ機能の一部を利用して作成する並列化支援機能を GNU Emacs に埋め込む方法を述べる．そして 4 章では関連研究を紹介し，5 章でまとめ，6 章で今後の課題を述べる．

2. iPat/OMP の機能及び設計指針

2.1 並列化支援機能

iPat/OMP は下記の 4 つの支援機能を持つ．

- 並列性解析
  指定されたプログラム部分の並列性解析を行い，その解析結果を提示する．
- 指示子作成
  並列化可能な部分に対する OpenMP 指示子を提示する．
- プログラムリストラクチャリング
  プログラムの並列化及び最適化に必要なプログラム変換を行う．
- 実行時間解析
  指定されたプログラム部分の実行時間計測に必要なコードを挿入する．

2.2 設 計 指 針

2.2.1 エディタ組み込み型支援機能

iPat/OMP では並列化支援機能を広く利用されているエディタに組み込むこととする．

2.2.2 ツールの可搬性と拡張性

iPat/OMP では，エディタに組み込む支援機能は特殊なライブラリや実行アプリケーションを用いずに実装可能とし，可搬性の高い支援ツールを実現する．

また iPat/OMP では新たな解析方法等を容易に追加できる枠組みを持つこととし，拡張性のある支援ツールを実現する．

2.2.3 ユーザとツールの役割分担

iPat/OMP では，ユーザとの役割を以下のように分担することとする．(**図 1**)

- iPat/OMP はユーザからの要求に応じてプログラムの解析を行い，その解析結果をユーザに提示する．

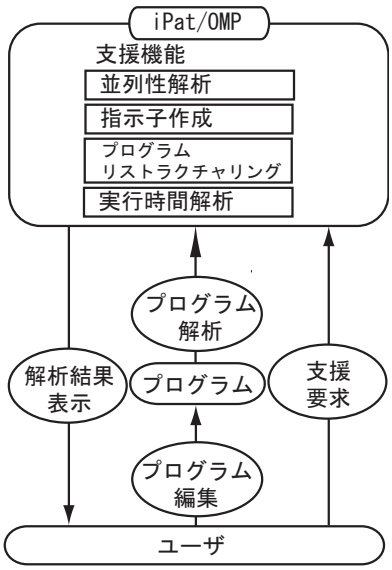

図 1 iPat/OMP とユーザの役割
Fig. 1 Role assignments between iPat/OMP and user.

- ユーザは iPat/OMP に支援を要求し，iPat/OMP が提示した結果を基にプログラム並列化の判断を行い，プログラムに指示子を挿入して並列化を行う．

このように役割分担することにより，以下の 2 つの利点を持つことが期待できる．

(1) 読みやすいプログラムの維持
自動的なプログラムの変更は，ユーザにとって読みづらいプログラムの作成を行う可能性がある．

(2) 並列化対象の適切な選択
自動的な並列化は，オーバヘッドを増加させる無用な並列化を行う可能性がある．

2.2.4 ユーザとの情報交換

iPat/OMP はユーザに情報を提供するだけではなく，プログラムの特性に関するユーザの知識を受け入れることを可能とし，その情報を利用した支援を行うことを可能とする．

ユーザはプログラムに関する知識を iPat/OMP に提供するために，コンパイラ指示子を利用してその知識をプログラム中に記述することとする．

プログラム中に指示子を記述して情報を提供することは，必要な情報をソースプログラム中に書き込むという OpenMP の思想に基づいたものである．

2.2.5 対象ユーザ

iPat/OMP では，プログラム並列化の初心者からプログラム並列化の専門家までのユーザを支援可能と

する．そのために iPat/OMP では各ユーザのプログラム並列化に関する知識レベルに合わせて支援方法を容易にカスタマイズできることとする．

3. iPat/OMP の実装

iPat/OMP では並列化支援機能を組み込むエディタとして GNU Emacs を選択する．

Emacs では EmacsLisp プログラムによって，新しい機能を追加することが可能である．

iPat/OMP は２つのコンポーネントによって構成される．ひとつは並列化支援機能であり，もうひとつはそれら支援機能を Emacs に組み込むための機能である．以下にこの２つのコンポーネントの実装方法を述べる．

3.1 並列化支援機能の実装

3.1.1 OmniOpenMP コンパイラ

iPat/OMP の並列化支援機能では OmniOpenMP コンパイラ[2] が有する機能を用いてプログラムの解析や変換を行う．

OmniOpenMP コンパイラは C と Fortran のためのフロントエンド C-front と F-front を持つ．それらは C と Fortran を中間コードである Xobject[3] へ変換する．また C-front は Xobject を C へ逆変換する機能を持つ．

Xobject は OmniOpenMP コンパイラで使用している可読性の高い抽象構文木構造の中間言語である．そしてこのコードの解析や変換を行うための機能として OmniOpenMP コンパイラは Exc toolkit[4] と呼ぶ Java のクラスライブラリ群を備えている．

3.1.2 支援ライブラリの実装

iPat/OMP 用に作成される支援機能を支援ライブラリとしてまとめることとする．支援ライブラリは支援関数群とシェルスクリプトで構成される．支援関数群は，Xobject を解析可能な Exc toolkit を使用するため Java で実装し，シェルスクリプトはそれら支援関数群を制御するために用いる．シェルスクリプトや支援関数群からの情報表示は標準出力へ表示するように定義される．

このような構成にすることで，ユーザは支援ライブラリを Emacs 上の iPat/OMP 環境で使用する以外にも，ターミナルのコマンドライン上から使用することができる．

支援関数の例として，ループの並列性解析と OpenMP の指示子作成の機能を以下に示す．

ループ並列性解析関数

この関数は for ループ内のデータ依存解析を行い，その結果の提示を行う．依存解析手法としては，線型方程式と線形不等式についての解の有無を評価して依存の有無を判断する *Power Test*[5][6] を用いた．

指示子提示関数

この関数は並列化可能な for ループに対して OpenMP 指示子である *#pragma omp for* とそれに付随する *shared* や *private* 節の提示を行う．現時点では，*shared* や *private* 節にリストする変数を下記の通り定義している．

- 配列変数:*shared*
- ループ制御変数:*lastprivate*
- それ以外でループ内に現れるスカラ変数:
  - 最初の参照が読み込み参照の場合:*firstprivate*
  - 書き込み参照をしている場合:*lastprivate*

この関数はループ内に存在するスカラ変数のプライベート変数化を試みる．

3.2 Emacs 組み込み機能の実装

Emacs に並列化支援機能を組み込むために，EmacsLisp 言語を使用した iPat/OMP のモード (Emacs 上で特定の作業を行うための環境) を作成する．モード作成のためには関数及びローカルキーマップの定義が必要となる．

現在 iPat/OMP では大きく分けて 3 つの関数を実装している．

iPat/OMP 起動関数

iPat/OMP 起動関数では，Emacs に iPat/OMP 環境を構築することを行う．この関数は Emacs 上のフレーム内のウインドウを上下二つに分け，上部をプログラム編集の環境を提供するメインウインドウ，下部を iPat/OMP への命令の受け付けを行い，支援機能からの情報を提供するサブウインドウと定義する．

またこの関数では，以下に示している関数及び今後新たに作成する関数を簡単なキー入力によって実行可能にするローカルキーマップの定義も行う．

支援命令関数

支援命令関数は，ユーザが命令した支援内容に対応している支援ライブラリ内のシェルスクリプトを実行する．その際この関数は，シェルスクリプトが標準出力へ表示した文字情報をサブウインドウへ表示するように定義する．

またユーザーが Emacs に備わっているマーカー機能を用いてメインウインドウ上の複数行に渡るプログラム領域を選択してこの関数を呼び出した場合，この

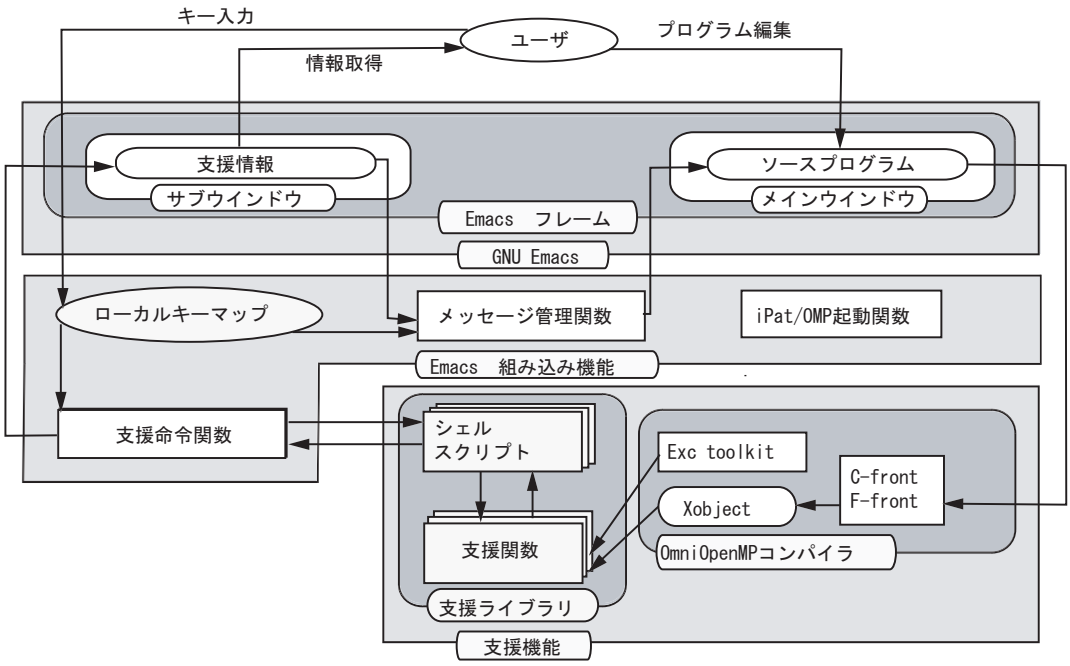

図 2 iPat/OMP の構造
Fig. 2 Structure of iPat/OMP.

関数はその領域内についてのみ解析等を行うようシェルスクリプトに引数を与える．

メッセージ管理関数

メッセージ管理関数では，サブウインドウ上に表示される支援機能からの情報とメインウインドウ上のプログラムの対応付けを行う．この関数はサブウインドウ上のカーソルがある行の支援情報について解析を行い，その支援情報が指し示しているメインウインドウ上のプログラム行に矢印を印字し，そのプログラム部分を表示する．

3.3 iPat/OMP を用いた並列化ワークフロー

ここでは，iPat/OMP を使用するユーザ，Emacs 上の iPat/OMP 環境，支援機能のそれぞれについてのワークフローを説明する (**図 2**)．

ユ ー ザ

- Emacs 起動中:ユーザは Emacs 起動中に iPat/OMP 起動関数を呼び出す．
- 支援要求時：ユーザは支援対象とするメインウインドウ上のプログラム範囲をマークし，サブウインドウにカーソルを移動して目的に合わせた支援コマンドを入力する．
- 支援要求後：ユーザは二つのことを行う．
  (1) メッセージ管理関数を実行し，支援情報が指し示しているプログラム部分を参照する．
  (2) サブウインドウ上の支援情報を基に並列化の判断を行い，指示子挿入などのプログラム編集を行う．

Emacs 上の iPat/OMP 環境

- iPat/OMP 起動時：iPat/OMP 起動関数はウインドウ分割及びローカルキーマップ設定を行う．
- 支援実行時：支援命令関数は支援ライブラリ内にあるシェルスクリプトを実行し，シェルスクリプトが実行中に標準出力へ表示した情報をサブウインドウ上に表示する．
- 支援実行後：メッセージ管理関数はユーザの命令に従ってサブウインドウ上のメッセージを解析し，メインウインドウ上のプログラムとの対応付けを行う．

支 援 機 能

- 支援実行時:シェルスクリプトは C-front を用いてメインウインドウ上のプログラムを Xobject へ変換し，支援関数を呼び出す．支援関数は Xobject の解析等を行い，その結果を標準出力へ表示する．

実行中のエディタ画面の例を (**図 3**) に示す．

4. 関 連 研 究

これまでの支援ツールには，プログラムデバッギングツール[7]，3D プログラム構造表示ツール[8]，実行時

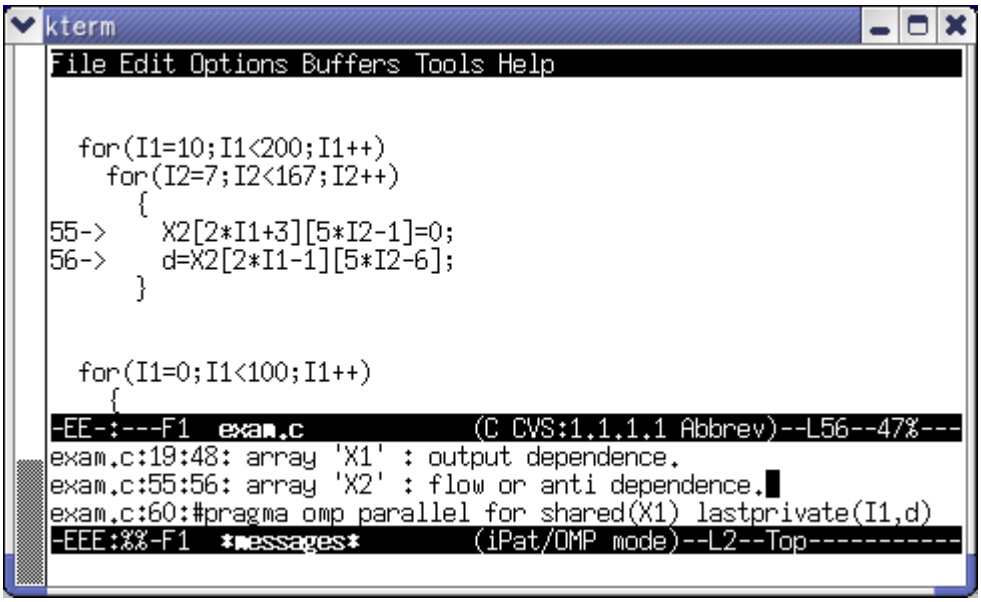

図 3 iPat/OMP 実行中のエディタ画面の例
Fig. 3 Example of editor screen in running iPat/OMP.

トレース情報表示ツール[9] などがある．それらは様々なアプローチでプログラマに情報を与えることで，人手によるプログラム並列化を支援する．

ParaScope エディタ[10] はプログラム編集環境とプログラムリストラクチャリング及び依存解析機能が一体となった Fortran 用並列化支援ツールである．このツールは依存解析の結果とプログラムの対応関係をユーザに判りやすく示すために，依存解析の結果をエディタの下部に表示し，結果の指し示すプログラム部分のフォントを変更する．

ParaWise[11] はプログラム解析とプログラムリストラクチャリングを行う並列化支援ツールであり，関数のコールグラフやデータ依存関係を図で表現する．

ParaScope エディタおよび ParaWise はプログラムに対して直接リストラクチャリングを施すツールである．

URSA MINOR と INTERPOL[12] は OpenMP プログラミングを支援するツール群である．URSA MINOR はプログラム実行時のパフォーマンス評価の結果を表示し，INTERPOL では Poraris コンパイラを実行して作成したプログラム解析結果を表示する．ユーザはそれらの結果を基に INTERPOL を用いて手動でプログラムの並列化を行うことができる．

PTOPP[13] は Emacs エディタ上でパフォーマンス評価機能等を利用可能にしたプログラム並列化支援ツールである．Emacs を利用するという PTOPP の考え方は iPat/OMP と同様のものである．

5. ま と め

本稿では，OpenMP プログラム作成支援ツールである iPat/OMP の設計指針と実装方法について報告した．

iPat/OMP では並列化支援機能を広く利用されているエディタに埋め込むことにより，ユーザが使い慣れたエディタ上でプログラム編集を行いながら並列化支援機能との対話的な作業を行うことを可能としている．また iPat/OMP ではその役割をプログラム解析とその結果を提示することに限定し，並列化の判断を行うこととプログラムの変更を行うことをユーザの役割としていることで，ユーザにとって読みやすいプログラムの維持と，適切な並列化選択を促進する．

iPat/OMP の実装では広く利用されているエディタとして GNU Emacs を選択し，そこに並列化支援機能を埋め込むことを行った．この実装により，iPat/OMP は次のような特徴を持つ．

- iPat/OMP は Emacs が提供しているプログラム編集環境を利用している．そのためユーザは普段から使い慣れているキー操作等でのプログラム編集ができる．
- iPat/OMP は Emacs が持つローカルキー定義の機能を利用している．そのためユーザは簡単なキー入力で支援命令の発行ができる．

- メッセージ管理関数を用いることにより，ユーザは支援情報が指し示しているプログラムの部分を容易に参照することができる．
- iPat/OMP は Emacs の 1 フレーム上でプログラム編集環境とプログラム並列化支援環境を提供している．よってユーザは iPat/OMP をシリアルコンソールやターミナルなどの文字端末上で使用することができる．

また，iPat/OMP の可搬性及び拡張性については以下のような特徴がある．

- iPat/OMP は EmacsLisp と Java とシェルスクリプトを利用して構成されており，iPat/OMP が実装できる計算機環境は広範囲で，可搬性が高い．
- Exc toolkit を利用した支援関数とそれを制御するシェルスクリプトを作成し，iPat/OMP モードに新たな支援命令関数を定義することにより，iPat/OMP に新たな支援機能を追加することが可能であり，機能拡張性が高い．

6. 今後の課題

現在 iPat/OMP を実用的なプログラム並列化支援ツールとするために，iPat/OMP モード中で使用する EmacsLisp 関数や並列性解析関数及び指示子表示関数の機能拡張を進めている．

今後支援ライブラリの機能追加を行うことを検討している．具体的にはプログラムに関するユーザの知識を iPat/OMP が取得する機能やプログラムリストラクチャリングを行う機能，そして並列化対象や並列化方式の選定に必要な実行時間計測を行う機能を実装する予定である．

ユーザにとって読みやすいプログラムを維持するために，リストラクチャリングの指示はコンパイラ指示子をプログラム中に記述することで行うこととする．そして実際のリストラクチャリングは，プログラムを実行ファイルにコンパイルする直前になんらかの機能によって行うことを検討している．

またこれら支援ライブラリが C だけでなく Fortran プログラムの解析を行なえるように改良し，iPat/OMP が行う支援の幅を拡げることを計画している．

さらに各ユーザのプログラム並列化に関する知識レベルに合わせて支援内容を容易にカスタマイズできる機構を今後検討し，それを実装していく予定である．

謝辞 本研究の一部は文部科学省科学研究費 (基盤, (A), (1), 14208026) によって行われた．日ごろから御助言・御協力をいただいている本科研グループの方々に感謝いたします．

参 考 文 献

1) http://www.openmp.org/
2) M. Sato, S. Satoh, K. Kusano and Y. Tanaka, Design of OpenMP Compiler for an SMP Cluster, EWOMP '99, pp.32-39, October 1999.
3) Xobject File Format http://phase.etl.go.jp/Omni/htdocs.ja/XobjectFileFormat.html
4) Exc compiler tools kit http://phase.etl.go.jp/Omni/htdocs.ja/exc-java.html
5) U. Banerjee, Loop Transformations for Restructuring Compilers The foundations, Kluwer Academic Publishers, January 1993.
6) M. Wolfe and C.-W Tseng, The Power Test for Data Dependence, IEEE Transactions on Parallel and Distributed Systems pp.591-601, September 1992.
7) J. Cownie and S. Browne, Portable OpenMP Debugging with TotalView, EWOMP2000 June 2000.
8) 羽田昌代, 笹倉万里子, 城和貴, NaraView を利用した実アプリケーションの並列化事例, 情報処理学会研究会報告 HPC-81-8, pp.39-44, June 2000.
9) W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach, VAMPIR: Visualization and Analysis of MPI Resources. Supercomputer, 12(1), pp.69-80, January 1996.
10) K. Kennedy, K. S. McKinley, and C.-W. Tseng, Analysis and Transformation in the ParaScope Editor. In Proc. ACM Int'l. Conf. Supercomputer. pp.433-447, June 1991.
11) Parallel Software Products Inc, ParaWise http://www.parallelsp.com/
12) I. Park, M. J. Voss, S. W. Kim and R. Eigenmann, Parallel Programming Environment for OpenMP, Scientific Programming, Vol. 9, No. 2/3, pp.143-161, 2001.
13) R. Eigenmann and P. McClaughry, Practical Tools for Optimizing Parallel Programs. 1993 SCS Multiconference, March 1993.