解　説

並列処理のためのシステムソフトウェア

4. 並列化支援システム†

菊池　純男††

1. はじめに

並列処理には粗粒度(coarse-grained)から細粒度(fine-grained)までが存在する．前者はベクトルプロセッサとマルチプロセッサに代表され，並列処理の対象は主にサブルーチン，ループを対象としている．一方，後者はスーパスカラ型 (VLIW : Very Long Instruction Word も含む) RISC プロセッサに代表され，命令語のレベルで並列処理が実現される．本稿ではマルチプロセッサ向け並列化支援システムを取り上げる．

マルチプロセッサを用いた並列処理によって逐次処理を遥かに凌ぐ高速性能を実現しようとする試みは 10 年以上前からあるが，近年実用化の兆しがみえ始めている．並列処理が本格的に普及するためには，並列記述言語の標準化とともに，FORTRAN のように過去のソフトウェア資産が多い場合には従来プログラムを自動並列化するコンパイラおよび並列化作業を支援するシステムが重要な鍵を握っている．

ここ数年の並列処理システムの動向によると，ユーザにとって使いやすい共有メモリ型並列計算機の本格的な普及を待たずに，ユーザの関心は分散メモリ型並列計算機へと移ってしまった感がある．特に最近，多数の高性能 RISC プロセッサをネットワークで接続した形態やワークステーションをクラスタ化した形態がスケーラビリティとコストパーフォーマンスの観点から注目を集めている．これを反映するように科学技術計算分野のみならず，急速なダウンサイジング化を進めつつあるビジネス分野まで狙った商用機の発表がなされている．

しかし，比較的歴史の新しい分散メモリ型並列計算機はもちろんのこと，すでに 10 年以上の歴史をもつ共有メモリ型並列計算機ですら，だれでも気軽に使える道具とは言えず，依然としてほんの一握りの専門家向けといった色彩が強い．この原因は，プログラミングとデバッグの困難さにあると言われている．

これを解決する鍵は自動並列化技術を含めた広い意味での並列化支援技術であり，現在も研究が活発に行われている．一方では，共有メモリ型並列記述向け仕様の規格化や分散メモリ型並列記述向け仕様として HPF (High Performance Fortran) が提唱されている．

本稿では，逐次プログラムの共有あるいは分散メモリ型並列計算機向け並列化支援，並列プログラムの性能チューニング支援という観点で現状と今後の課題について解説する．

2. 並列化支援システムの必要性と要求技術

2.1 必　要　性

並列化とは，共有メモリ型並列計算機では処理を複数のプロセッサに分割することであり，分散メモリ型並列計算機では複数のプロセッサにデータと処理を分割することを意味する．

並列処理を普及させるには，過去の膨大なソフトウェア資産の有効利用を図り，かつユーザの並列化作業の負担を極力軽減することが必要である．そのためには従来の自動ベクトル化技術同様，並列処理においても自動並列化は基幹技術である．

一方，並列処理は並列制御やプロセッサ間通信のオーバヘッドが大きいために粒度（並列処理単位の大きさ）を大きくしないとプロセッサ台数に見合った性能が得られない．苦労して並列化しても 1 台のときより遅くなることもある．このこと

† Parallelization Assist System by Sumio KIKUCHI (Central Research Laboratory, Hitachi, Ltd.).

†† (株)日立製作所中央研究所

は，次の二つのことを意味する．

第1は並列化する箇所の選択は注意深く行われなければならないということである．選択を誤ると性能上，深刻な問題となる．

第2は粒度を大きくするために多重ループ，さらに手続きにまたがった並列化を行うことが必要となることである．

前者のためには，コンパイラによってどこが並列化されたか，ということ以外に，どこを並列化すべきか，どこを並列化すべきでないか，をユーザが判断できる材料を提供することが必要となる．

後者のためには，手続きにまたがった高度なデータフロー解析技術が必須となる．しかるに，プログラム全体にわたってデータの依存関係を完全に解析することは技術的にきわめて困難なケースがある．さらに，このような技術的な問題以前に致命的な点はソースプログラムという静的な情報からだけでは確定せず，実行時に初めて確定するようなデータ依存関係が存在しうることである．たとえば，配列添字にDOループの制御変数以外の変数が含まれておりこの変数がプログラム実行時に読み込まれるデータである場合，コンパイラはこのような添字をもつ配列データの依存関係を解析できない．また，意味解析を必要とする**図-1**のようなケースもある．本図はPerfect club benchmarks[1]と呼ばれるスーパコンピュータ用ベンチマークプログラムの一部を抜粋したものである．なお，本稿の説明では，「定義」とはメモリに値を書き込むこと，「使用」とはメモリから値を読み出すことをさす．

DO 1000を並列化するための最大のポイントは，配列RLの各要素がループ繰返しにまたがって定義と使用の関係をもたないことを解析できるか否かにかかっている．配列RSについては，DO 1110でRS (1 : 9)の範囲が定義され，同一ループ内にある条件文①で同じRS (1 : 9)の範囲が使用される．さらに，DO 1130内の条件文②でRS (6 : 9)の範囲が使用される．よって，配列RSについてはDO 1000でループ繰返しにまたがって定義と使用の関係がないことが判明する．この関係は通常の配列データフロー解析によって明らかにすることができる．

ところが，配列RLについては簡単ではない．

```
      DO 1000 I=1, NMOL
        ...
        DO 1110 K=1, 9
          RS(K)=...
1110      IF(RS(K). GT. CUT2) KC=KC+1 ·····①
        IF(KC. EQ. 9) GOTO 1100
          ...
        DO 1130 K=2, 5
          ...
          IF(RS(K+4). GT. CUT 2) GOTO 1130 ···②
          RL(K+4)=... ·····④
          ...
1130    CONTINUE
        IF(KC. NE. 0) GOTO 20 ·····③
          ...
        DO 1140 K=11, 14
          =RL(K-5) ·····⑤
1140    CONTINUE
  20    DO 1150
          ...
1150    CONTINUE
          ...
1100    CONTINUE
          ...
1000    CONTINUE
```

図-1 Perfect club benchmarksの一例

①②③という三つの条件文の成否，配列RLに対する文④の定義と文⑤の使用との関係を注意深く観察すると，以下のことが分かる．

まず配列RL (6 : 9)が使用されるのは条件文③からKCが0のときである．ところが，KCの値が0になるのは条件文①からRS (1 : 9)の値がすべてCUT 2以下の場合である．よって，条件文②ではRS (6 : 9)がすべてCUT 2以下となるため，配列RL (6 : 9)への定義が実行される．すなわち，文④での定義範囲と文⑤の使用範囲は同一となり，しかも必ず，定義されてから使用されることが分かる．つまり，ループにまたがって定義・使用の関係がない．このことが解析できれば作業配列化変換 (array privatization)，すなわち配列RLを各プロセスごとの作業的な配列に置き換えることによって並列プロセス間のデータ依存をなくすことができ，並列処理可能となる．

ここで示した例は意味解析の領域に近く，コンパイラ技術で解析するのは困難である．考えられる一つの解は一度実行してprofile情報（実行時の情報）を収集し，その情報を利用することである．しかしながら，この方法で問題となるのは，プログラムのどこに（何に）着目し，どういう情報を集めるか，さらに収集した情報をどのように解析するかである．これをシステムで自動化する

のはきわめて困難と考えられる．

他の解は，解析できない点をユーザに支援してもらい，その情報を取り込むことである．この手段が実用的であるためには支援システムとは言っても強力なプログラム解析技術を備えていることが必須である．さもないとユーザは膨大な情報をシステムに提供しなければならず，支援作業は多大の工数を要することになる．自動化技術とは結局のところ，プログラム解析技術に尽きる．しかし，これにはおのずと限界があり，限界への実用的な対処手段を用意しておくことはきわめて重要である．しかも，有効な手段となるためにはコンパイラとの密な連携が取れていなければならない．

次節では，並列化支援システムが具備すべき機能とそのための技術について述べる．

2.2 具備すべき機能と要求技術

並列化支援システムが具備すべき機能は大別すると以下の4つと考えられる．

- 性能モニタ機能
- 対話機能
- プログラム解析機能
- データベース化機能

これら機能が統合化された環境として提供される必要がある．

2.2.1 性能モニタ機能

性能のモニタリングでは，実行時に低オーバヘッドで情報を収集する技術，それを解析し，見やすく表示する技術が要求される．低オーバヘッドで情報を収集するために専用のハードウェア機構を設けることもある．オペレーティングシステムとも深く関わりがある．

（1） プログラム実行負荷解析・表示

プログラムの各部分ごとの実行時間と全体に占める比率に関する統計情報である．ユーザは並列化対象，チューニング対象を選択するときの判断基準として利用する．支援システムとして最も基本的な機能である．この情報を収集するためには一度プログラムを実行することが必要である．通常，手続きとループごとの情報を必要とする．

（2） 並列プロセス実行状況解析・表示

並列に実行される各プロセスごとの実行開始・中断・待ち（同期，データ受信待ち）・終了を時系列で表示する．また，プロセス間のイベントの対応関係を併せて表示する．これによってプロセス間の実行関係を把握することができる．並列プログラムの性能向上を図るには，各プロセスの待ち時間を最少にすることが必要であり，プロセス実行状況解析はプログラムチューニング上，有益な情報を提供する．

（3） プロセッサ負荷バランス解析・表示

一般に並列処理では，すべてのプロセッサの負荷を均等にしたほうが性能が向上する．これは特定のプロセッサが性能上のボトルネックになることを防ぐという意味がある．そこで，各プロセッサの実効的な処理時間，つまり待ち時間を除いた時間を全実行時間に対する割合で示す．これにより各プロセッサの実効的な稼働率を知ることができる．

（4） プロセッサ間通信解析・表示

本機能は分散メモリ型並列計算機に固有である．分散メモリ型並列計算機では，アプリケーションプログラムのデータは各プロセッサに分割して割り当てられる．したがって，自プロセッサの演算に必要なデータのうち，自プロセッサが保持しないデータはプロセッサ間でデータの送受信を行わねばならない．この種の計算機では，データ通信が性能のボトルネックになるケースが多く，プロセッサ間の通信回数，通信量を極力少なくすることが性能上重要である．しかも，通信量はプロセッサ間で均一にしたほうがネットワーク競合が発生しにくい．各プロセッサの通信回数，通信量に関する情報はデータ分割戦略を決定する上で重要である[31]．

（5） プロセッサ負荷内訳解析・表示

各プロセッサごと，あるいは全プロセッサの処理時間の内訳に関する情報を提供する．内訳としては以下の項目が含まれる．

- データ送信時間
- 受信待ち時間
- 演算時間
- 入出力時間

など，これらによって並列プログラムの性能ボトルネックが明確になる．

2.2.2 対話機能

並列化作業の効率を考えると対話環境は必須である．自動並列化コンパイラによって並列化されなかったプログラム部分を並列化しようとする

場合，ユーザは以下の2点を知ることが必要となる．

- どこの，何が解析できなかったか
- 並列化できないと判断した原因はどこの，何か

「どこの」はユーザが対象ソースプログラムとの対応を取るために必要である．

これらをユーザとの対話で解決を図るためには並列化システムの内部解析情報の不明点をユーザと対話できる形式に変換すること，およびユーザから得た回答を内部解析情報に変換することが必要となる．この場合，難しいのは何をどうユーザに聞いたらよいかである．多様なユーザレベルへの対応が要求される．

2.2.3 プログラム解析機能

（1） データ依存関係解析

変数および配列データの参照関係(定義・使用)を解析する．大きな粒度で並列化するためには複数手続きにまたがるプログラム全体での解析が必要となる．

配列データに関する解析方法としては二つの方式が知られている．

第一の方式は，従来から行われてきた方式で，**図-2** に示すように各参照間について

a） フロー依存
b） 逆 依 存
c） 入 力 依 存
d） 出 力 依 存

を求める方式である[9]．この方式は，手続き内の解析に用いられてきた．最近，変数データに関して手続き間に拡張する方式も報告[14]されているが，これを配列データにまで拡張するのは困難と考えられる．

第二の方式は，リージョン解析と呼ばれる方式である[10)~13)]．手続きにまたがった配列データ参照解析では，このリージョン解析が用いられる．解析対象は手続きにまたがって参照されるデータである．たとえば，FORTRAN プログラムではコモンと引数に使用される配列データが解析対象となる．この解析は手続きが呼び出されたときの配列の参照される範囲（リージョン)を求めるものである．基本リージョンとしては以下に示す4つの種類がある．

- MOD (Modify)：定義される可能性のあるリージョン
- USE：使用される可能性のあるリージョン
- KILL：必ず定義されるリージョン
- EUSE (Exposed USE)：定義されずに使用（露出使用）されるリージョン

基本的な解析手順を**図-3** に示す．まず手続きに含まれる各ループ内で参照される配列データについて，上記リージョン集合を求める．ここでリージョン集合とは，参照範囲付きの配列名の集合をいう．さらに，ループごとの情報とループでない部分との情報を集約し，手続きの入口でのリージョン集合を求める．

リージョン解析には，さらに flow-insensitive な解析方式と flow-sensiitve な解析とがある．前者は，制御フローを考慮しない，すなわち配列データへの定義と使用の順序を考慮しない解析方式である．上記 MOD, USE リージョンは flow-insensitive な解析で求めることができる．

一方，後者は制御フローを考慮した解析であり，KILL, EUSE リージョンの解析が可能となる．これらのリージョンは，作業配列化 (array privatization) を行うために必須である．

（2） 並列性解析

（1）で述べた複数手続きにまたがったプログラム解析結果に基づき，ユーザが指定したプログラ

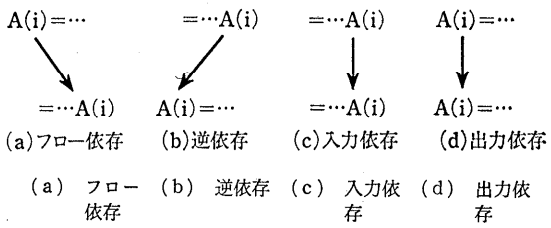


図-2 データ依存の種類

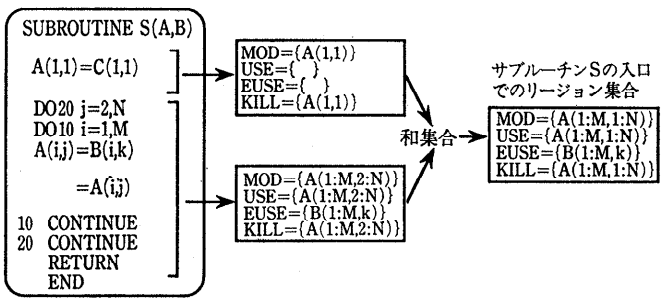


図-3 リージョン解析の基本手順

ム部分の並列性を解析する．ループ並列化条件は，異なるループ繰り返し回で，定義範囲が重ならないことおよび定義範囲と使用範囲が重ならないことである．これをリージョンで表現すると**表-1** に示すようになる．

また，プログラム解析の限界に対処するためにユーザとの対話で得た情報を取り込み，並列性解析に利用する機能が必要である．

（3） プログラム変換技術

通常，共有メモリ型並列計算機向けの並列化支援システムでは，並列化可能であると判断できる場合は，コンパイラ指示文 (directives) をソースプログラムに挿入することが行われる．支援システムからコンパイラへの情報の受渡し方はソースプログラムで行えるのがポータビリティの面でよい．しかし，共有メモリ型並列では現状 directives 以外にコンパイラとのインタフェースを取る有効な手段はない．当然，コンパイラはこの directives を理解し，ソースプログラムを並列オブジェクトコードに変換できなければならない．

以上はコンパイラと支援システムが独立している場合である．他の形態として支援システムがオブジェクトコード出力機能をもってもよい．この形態は対話型コンパイラと呼ぶことができる．現在，商用化されている並列化支援システムは，ターゲットコンパイラ向けの directives を挿入したソースプログラムを出力するトランスレータ方式が採られている．

一方，分散メモリ型並列計算機向けの並列化支援システムでは，Fortran 77/Fortran 90 で記述されたソースプログラム，あるいは HPF で記述したソースプログラムから通信文などの並列制御ライブラリを含んだソースプログラムへの変換が行われる．こちらもトランスレータ方式が採られている．最近，並列プログラムのポータビリティ確保のために並列制御ライブラリのアプリケーションプログラムインタフェースの共通化を目的とした活動 (Message Passing Interface Standard Meeting) が米国で始まっている．

（4） プログラム構造解析

ユーザが並列化を進める過程で対象プログラムの全体構造，動作の理解が必要となる場合がある．特に，他人の作ったプログラムでは必須である．

このプログラム理解のために以下の情報が有用である．

- 手続きの呼び出し関係
- 複数手続きにまたがったデータの参照関係の追跡

以上はプログラム解析の結果として得られる情報であり，この可視化技術さらにはプログラム実行の様子をアニメーション表示する技術が必要である．

（5） 静的性能予測

並列化作業を進める場合，並列化によってどれだけ速くなるのか，が知りたくなる．その際，実際に並列機で実行しなければ，どの程度性能が向上するか分からないのでは使い勝手が悪い．そこで，ワークステーション上でターゲットマシンに合わせた，およその性能を，しかも実行せずに静的に予測できることが望ましい．

2.2.4 データベース化機能

複数手続きからなるプログラム開発を一元管理する技術はソフトウェアデータベース技術[15]と呼ばれており，統合化されたプログラミング環境を実現するために必須の技術といえる．

並列化支援システムにとっても，

- ソースプログラムと解析情報の関係

表-1 並列化可能条件

分類		並列化可能条件
ループ内に定義なし		①無（無条件で可能）
ループ内に定義あり	DO J 露出使用：ARRAY*euse*(J) 定　義：ARRAY*kill*(J) 使　用：ARRAY*use*(J)	②すべての J について ARRAY*kill*(J)=ARRAY*use*(J), ARRAY*euse*(J) あるいは， 任意の Jm, Jn について ARRAY*kill*(Jm)≠ARRAY*use*(Jn), ARRAY*euse*(Jn) ③任意の Jm, Jn について ARRAY*kill*(Jm)≠ARRAY*kill*(Jn)

- チューニング前後のソースプログラムの関係
- ソースプログラム，オブジェクトプログラム，ロードモジュールの関係
- 手続きの呼出し関係

などの管理に必要である．

特に，複数手続きにまたがったプログラム解析を行った場合には，ある手続きの変更が全体の解析結果に影響を及ぼす．したがって，手続きとその解析情報のバージョン管理は正確に行われなければならない．

3. 並列化支援システムの現状と技術

本章では，代表的な並列化支援システムを紹介する．並列化支援システムには次に示す二形態があり，いずれもトランスレータとして提供されている．

a） 自動トランスレータ

b） 対話型トランスレータ

いずれも自動並列化コンパイラで用いられている技法と重なる部分が多い．b)では，いわゆる並列化機能以外に，ユーザの並列化作業を支援するためにプログラム解析情報を視覚化する機能などを提供するものもある．この分野の研究は米国において盛んであり，並列処理研究の歴史を感じさせる．

まず，**3.1** では企業が提供するいくつかの並列化支援システムを，**3.2** では大学・公的機関のシステムを紹介し，**3.3** では筆者らが試作した並列化支援システム Parassist (Parallelization Assist System) を紹介する．なお，**表-2** にここで取り上げるシステムの一覧を示す．

3.1 企業関係

以下，4つのシステムを取り上げる．特に豊富な機能をもつ **3.1.3** FORGE 90, **3.1.4** Express については詳細に紹介する．

3.1.1 VAST (Pacific-Sierra Research Corp.)

PSR 社は 1971 年に設立され，FORTRAN/C プログラム用のベクトル，並列向け自動トランスレータ VAST を提供している．VAST はターゲットマシン向けにプログラム再構成やコンパイラ directives を挿入する機能をもつ[2]．以下，並列化のための主要な機能を示す．

- ループ以外の parallel-sections の検出
- parallel-loop と parallel-sections の大粒度 parallel-regions へのマージ
- micro-tasking directives の挿入

3.1.2 KAP (Kuck & Associates, Inc.)

Kuck & Associates 社は Illinois 大の D. J. Kuck 教授らが 1979 年に設立した会社である．KAP は FORTRAN/C プログラム用のスカラ，ベクトル，並列向けの自動トランスレータである．特に，キャッシュベースのプロセッサ向け最適化トランスレータとして注目を集めている．

プログラム解析技術やループ構造変換技術，メモリ階層向け最適化技術（タイリング，メモリ配置制御など）に優れている．

以下，並列化のために提供されている主要機能について示す[22]．

- ループ構造変換（展開，分割，融合，交換）
- 粒度の増大
- 同期頻度の低減
- プロセス fork/join 頻度の低減
- 手続きのインライン展開
- parallel-loop/paralll-sections の検出
- micro-tasking directives の挿入

3.1.3 FORGE 90 (Applied Parallel Reseach, Inc.)

以前 MIMDizer と呼ばれていた分散メモリ型並列計算機向け並列化支援システムにベクトル化機能，共有メモリ向けループ並列化機能などを追加し，Fortran 90 対応とした対話型トランスレータである．

FORGE は各種プログラム解析結果をデータベース化し，ユーザが容易にアクセスできる X-window ベースのインタフェースを提供している．まず，ユーザは並列化対象の逐次プログラムを一度実行させることによって手続きごと，ループごとの実行時間 profile 情報を得る．ユーザはこの情報を見ながら，並列化対象部分を選択する．並列化作業は自動化よりも会話形式によるユーザ指示によって進められる．この過程でユーザはデータベースの各種情報を参照することによって，プログラムを理解しながら作業を進めることができる．以下，FORGE を構成する主要サブシステムについて簡単に解説する[2),3)]．

a） プログラム解析サブシステム[3)]

最大の特徴は，複数手続きにまたがった変数および配列データの参照解析技術にあり，プログラム全体にわたったデータ参照関係がデータベースとして保持される．また，本解析によって手続き呼出しを含むループの並列性解析を可能にしている．

b） Database Viewing サブシステム[3)]

データベース化された情報に基づき，以下のViewing 機能が提供されている．

- サブルーチン，DO ループごとの実行時間表示
- サブルーチンコールトリー表示
- 複数手続きにまたがったデータの参照関係追跡（定義，使用場所の追跡）
- COMMON, EQUIVALENCE のメモリマップ状態表示

c） コード変換サブシステム[2),3)]

以下の機能が提供されている．

- micro-tasking directives の挿入
- 手続きのインライン展開
- ループ構造変換（展開，分割など）

表-2 並列化支援システム一覧

分　類	システム名	特　徴
企　業	VAST： Pacific-Sierra Research Corp.(米)	ベクトル，共有メモリ型並列向け自動トランスレータ ●プログラム変換（ループ構造変換など） ●コンパイラ directives 挿入
	KAP：Kuck & Associates, Inc.(米)	スカラ，ベクトル，共有メモリ型並列向け自動トランスレータ ●プログラム変換（ループ構造変換など） ●プロセス fork/join 頻度低減 ●parallel-loop/parallel-sections 検出 ●micro-tasking directives 挿入
	FORGE 90： Applied Parallel Research, Inc.(米)	ベクトル，共有・分散メモリ型並列向け対話型トランスレータ ●手続き間プログラム解析 ●プログラム解析結果の Database 化 ●プログラム変換（ループ構造変換など）
	Express：Parasoft Corp.(米)	共有・分散メモリ型並列向けプログラミングシステム ●communication profiling tool ●event profiling tool ●executing profiling tool ●memory access visualization tool ●自動並列化トランスレータ
大学・公的機関	ParaScope Editor：Rice 大 (米)	共有・分散メモリ型並列向け対話型トランスレータ ●手続き間プログラム解析 ●データ依存解析表示・編集 ●プログラム変換（ループ構造変換など） ●メモリ参照局所化，最少化変換
	SUPERB：Bonn 大 (独)	分散メモリ型並列向け半自動並列化トランスレータ ●対話型プログラム解析 ●ユーザへの解析情報提示 ●並列化変換
	PIE：Carnegie Mellon 大 (米)	性能チューニングシステム ●プログラム実行にともなう並列度状況表示 ●並列度別の実行時間分布表示 ●CPU 利用内訳表示 ●スレッドのプロセッサへのスケジューリング状況表示
	ParaGraph： Oak Ridge National Laboratory (米)	分散メモリ型並列プログラム性能チューニングシステム ●プロセッサ利用効率表示 ●プロセッサ負荷バランス表示 ●プロセッサ間通信頻度，通信量，総通信パターン表示 ●送受信プロセッサ関係表示

- サブルーチンコールを含むDOループでのサブルーチンへのDOループ押込み

特に，第4点目は手続きにまたがったループ再構成であり，高度な機能を提供している．

3.1.4 Express (Parasoft Corp.)

Parasoft 社は 1987 年にカリフォルニア工科大学出身の研究者らによって設立された．

Express は共有／分散メモリ型並列計算機，ワークステーション network-cluster 向けの並列プログラミング環境であり，マルチベンダ対応の実行時環境と以下のサブシステムから構成される[20]．なお，以下のa)，b)，c)を合わせて PM (Performance Monitor) と呼んでいる．

a) CTOOL (Communication profiling TOOL)

並列プログラムの各プロセスごとの CPU 時間内訳を解析し，その結果を視覚化するサブシステムである．これによってプロセス間通信のオーバヘッドや OS のシステムコール，I/O などの占める時間を知ることができる．

b) ETOOL (Event profiling TOOL)

OS やユーザのイベント（同期制御，通信など）モニタリングシステムである．結果は時系列で視覚的に表示されるため，ユーザはアプリケーションプログラムを実行する複数プロセスの制御関係を視覚的に理解することができる．

c) XTOOL (eXecution profiling TOOL)

サブルーチンごと，ソースプログラム文ごとの CPU 消費時間を表示する．プログラムチューニング用の最も基本的な機能である．この情報によってユーザアプリケーションの実行負荷の重い部分を把握することができる．

d) VTOOL (memory access Visualization TOOL)

逐次プログラムのメモリアクセス状況をアニメーションで表示するサブシステムである．配列データを2次元マトリックスにマッピングし，配列参照順に対応する要素位置を高輝度表示する．このサブシステムは，ユーザがメモリ階層を意識してプログラムをチューニングする場合や逐次プログラムの並列化，特に分散メモリ型向けにデータ分割方法を決定する場合に重要な情報を提供する．

e) ASPAR (Automatic and Symbolic PARallelization)[21]

逐次プログラムを並列プログラムに自動変換するトランスレータである．プログラム解析の結果としてプログラムの制御フローやデータ依存を視覚的に表示する機能をもつ．特に，プログラムで用いているアルゴリズムを認識し，並列向きのアルゴリズムに変換する点に特徴がある．

3.2 大学・公的機関関係

米国の大学を中心に多くのシステムが発表されている．以下，いくつかのシステムを紹介する．特に永年にわたり，研究を続けている RICE 大のシステムについては **3.2.1** で詳細に紹介する．

3.2.1 ParaScope Editor[4),5)]

並列プログラミング分野で古くから研究を続けている米国 RICE 大のシステムである．これまで R^n (Rice Programming Environment)[8)], PFC (Parallel Fortran Converter)[7)], PTOOL (Parallel programming assistant)[6)] といったシステムを開発してきており，ParaScope はこれらを発展させたシステムである．

(1) プログラム解析

手続き間解析を含むデータ依存解析を行う．配列データは負荷の軽い簡単な解析機能と dependency testing と呼ぶ負荷の重い詳細な解析機能とが用意されている．通常は高速で簡単な解析を行っておき，必要に応じて詳細な解析が行えるようになっている．

(2) データ依存関係表示・編集

データ依存解析の結果を表示・編集する機能である．一度に多くの依存が表示されてもユーザは理解することができない．そこで特定の依存のみを表示するフィルタリング機能が用意されている．

また，ユーザが依存を消去できる機能を備えている．データ依存解析では二つの配列が同じメモリ領域を参照するか不明な場合，通常同一メモリを参照しうると考える．その結果，実際には存在しない依存関係が解析結果に反映され，並列化を阻害することになる．したがって，ユーザが依存なしと判断できる場合には，依存を消去しデータ依存解析結果を編集することができるようになっている．データ依存解析を完全に行うことは困難であるため，その対処手段としてこのような機能

を提供している.

(3) プログラム構造変換

対話型プログラム構造変換機能である. ユーザはプログラムの一部を選択し,そこに施す変換をメニューから選択する. 変換はデータ依存をチェックし,問題がなければ実行するが,もし実行結果が不正となってしまうような変換の場合にはユーザにそのデータ依存を知らせる. ユーザはこれを無視して,変換を指示することもできる.

以下の4つに分類されたプログラム変換機能が用意されている.

a) reordering transformation

主にループ粒度を大きくするためのループ変換(交換,融合,一重化など).

b) dependence breaking transformation

並列化を阻害する要因を削除,分離するための変換(ループ分割,ループ端点展開,変数の配列化,配列名リネイムなど).

c) memory optimizing transfomation

メモリ参照の局所化,最少化のための変換(配列要素の変数化,ストリップマイニング,ループ展開など).

d) miscellaneous transformation

定数伝播,逐次ループと並列化ループとの相互変換など.

3.2.2 SUPERB (SUprenum ParallelizER Bonn)[16]

SUPRENUM プロジェクトの一環でドイツのボン大学が開発した半自動の並列化システムである. ターゲットマシンは SUPRENUM と呼ぶ分散メモリ型並列計算機である. SUPERB は対話型トランスレータであり,逐次プログラムを SUPRENUM FORTRAN プログラムに変換する. SUPRENUM FORTRAN は FORTRAN 77 に並列向け言語拡張を施したものである.

本システムのデータ依存解析では解析精度を高めるために,まず手続きにまたがって定数伝播を行う. これによって配列添字に現れる変数やループ繰返し範囲の上下限を定数として求める. もし,定数伝播できない場合には,ユーザとの対話によって変数の値や範囲を決定する.

本システムでは,本質的にデータ依存解析は完全には行えない,したがって並列化は完全には自動化できないという考え方をとっている. そこでユーザへの解析情報の提供や並列化変換システムの提供に力点を置いており,並列化に関する知識をもったユーザを対象としている.

3.2.3 PIE[17),18)]

本システムは,性能チューニングあるいは性能デバッギングのためのシステムであり,いわゆる並列化システムではない. アプリケーションプログラマや並列システムの設計者(スレッドのスケジューラ設計者など)の支援が目的であり,並列プログラムの実行状況をモニタリングし,その結果を視覚化表示する. ユーザレベルの計算と OS カーネルの振舞いの性能を評価することができるよう以下の機能が提供されている.

- プログラム実行にともなう並列度の変化
- 並列度別の実行時間分布
- CPU の利用内訳
- スレッドのプロセッサへのスケジューリング状況

3.2.4 ParaGraph[19)]

分散メモリ型並列プログラムの挙動のグラフィカルアニメーション表示とそのプログラム実行性能を把握するためのシステムであり,豊富な表示機能を備えている.

本システムの入力はトレースファイルであり,トレース情報は PICL (Portable Instrumented Communication Library) と呼ばれる低オーバヘッドのマルチベンダ対応のライブラリによって収集される. トレース情報は以下に示すような種々の角度からグラフィカル表示される.

- プロセッサ利用効率
- プロセッサ負荷バランス
- プロセッサ状態 (busy, idle, overhead) の時間経過にともなう台数推移
- プロセッサ間通信頻度,量,総通信パターン
- 送受信プロセッサ関係のマトリックス表示

3.3 Parassist

Parassist は筆者らが試作した共有メモリ型並列計算機向け並列化支援プロトタイプシステム[24)]である. 図-4 に本システムの構成を示す. 以下, 3.3.1 では4つの主要なコンポーネントの機能概要を紹介し, 3.3.2 で並列化効果を示す.

3.3.1 システム構成と機能

（1） 自動並列化コンパイラ

a） 多重 DO ループ自動並列化

サブルーチン呼出しのない DO ループを自動並列化する.

b） directives に従った並列化

- 並列ループ文
- Parallel Section 文
- 同期排他制御文

など.

c） 並列プロセス制御の自動化

- プロセスの生成，消滅
- プロセス起動，待ち

d） 並列化診断情報出力

- 並列化ループに関する情報
- 並列化不能要因

など.

（2） 解析システム[23),25),32)]

flow-sensitive な複数手続きにまたがった変数，配列のデータ参照関係の解析を行う．解析は手続き内と手続き間とに分けて行い，おのおのの解析結果はデータ参照解析情報としてファイルに出力する．このとき，コモンや引数となっている配列データについてはリージョンを解析する．解析手順は，まず変数について手続き内，手続き間解析を行い，ついでこの結果を利用して配列の手続き内，さらに手続き間解析を行う．この際，配列解析に先だって手続きにまたがった定数伝播を行う．これによって配列データの添字内変数やループ繰返しの上下限が定数値に置き換えられる場合もあり，配列データの依存解析精度の向上を図っている.

（3） 並列化システム[26),27)]

ユーザが選択した DO ループの並列性を解析し，並列化判定を行う．並列化可能ならば，コンパイラが並列化するのに必要な情報を directives としてソースプログラムに挿入する.

a） 並列実行負荷表示

手続き，ループごとの占める実行時間を表示する.

b） 並列性解析

ユーザが選択した DO ループについて並列性を解析する．ループ内に出現する手続きにまたがるデータ（コモン，引数）については手続き間データ参照解析情報を取り込み，並列性を解析する.

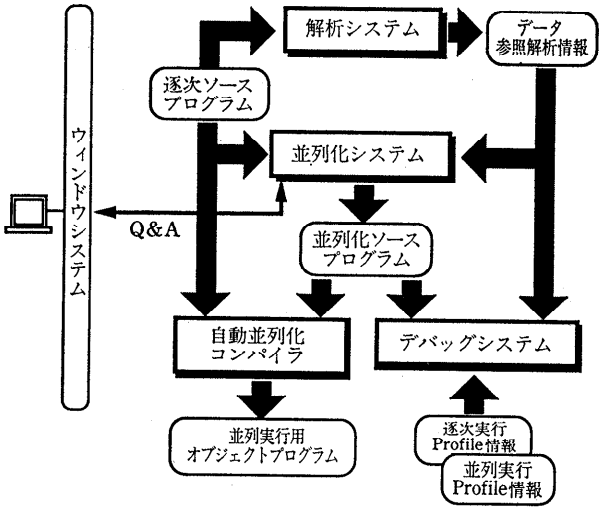


図-4 Parassist の構成

c） 並列化判定

並列性解析の結果から並列化できるか否か，さらに粒度を考慮し並列化すべきか否か判定する．粒度不足と判断できる場合には，並列化可能でも並列化しないと判定する.

静的な解析結果だけでは並列化判定できない場合には，並列化条件を作成し，ユーザとの対話処理にて解消を図る.

d） ユーザとの対話

並列化判定する上で必要な情報（データ依存，ループ長など）のうち，不明なものについてユーザとの Q&A を行い，これを解消する．その結果，並列化条件を満たせば並列化可能と判定する.

e） ソースプログラム変換

並列化可能と判定された場合，コンパイラの並列化診断情報に照らして，directives をソースプログラムに挿入する.

（4） デバッグシステム[28)~30)]

a） 並列構造表示

ソースプログラムから directives および DO ループ文を抽出し表示する．この表示は b） 以降の機能を利用する際の共通インタフェースである.

b） 不正並列化 directives の検出

逐次実行結果との値比較による不正なユーザ directives を一括検出する.

c） 並列性検査

逐次実行時のメモリ参照情報を利用した並列化

阻害要因の検出とメモリアクセス状況をアニメーション表示する．複数プロセスの同一メモリ領域への参照履歴情報を用いて並列性を検査する．

d） 並列実行プロセス表示

並列実行時のプロセスに関するprofile情報を解析し，プロセスの実行状況を表示する．

- プロセスの実行，待ち状態表示
- POST, WAIT の対応表示
- 並列実行のネスト時，プロセスをグルーピングし，マクロ表示

3.3.2 並列化効果

図-5 に本システムを用いて並列化した場合と自動並列化コンパイラのみで並列化した場合の並列化率の評価結果を示す．評価対象は Perfect club benchmarks の中から6題を選択した．図に示すように，並列化システムによって並列化率の平均を55%から83%へと向上させることができる．

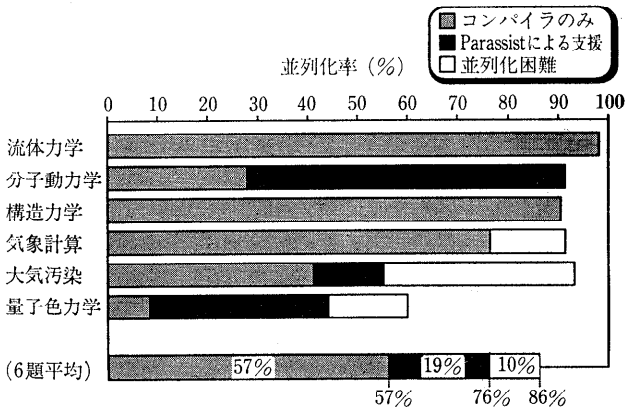


図-5 Perfect club benchmarks での評価

4. 今後の課題

今後，分散メモリ型並列計算機の普及につれ，並列化支援システムの重要性は一層増すと予想される．逐次プログラムからの並列化は並列処理普及のキーポイントである．そのためには，プログラム解析技術の一層の高度化を追及するとともに，その対話化が必須である．ユーザとの対話の接点をどこに求めるかは実評価を踏まえて検討が加えられなければならない．この際，並列処理に対するユーザの知識や経験に応じた対話方法も考慮することが必要である．

また，多くのプロセッサによって実行されるプログラムの挙動をどのように視覚化するか，いわゆる program-visualization 技術も重要である．

5. おわりに

以上，並列化支援システムの研究の現状を中心に紹介した．今後，コストパーフォーマンスに優れた並列計算機の登場と相まって並列化支援システムも発展していくと考えられる．並列計算機が気軽な道具として使えるようになることを期待したい．

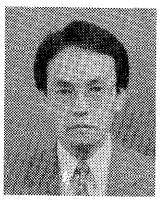
謝辞 本稿を執筆するにあたり，ご支援，ご助言をいただいた当所ユーザインタフェース部宮本俊介部長ならびに関係各位に深謝いたします．

参考文献

1) Perfect Club Benchmark Suite : Documentation, Mar. 1989, Center for Supercomputing Research and Development, Univ. of Illinois at Urbana-Champaign, Email : warmbier@csrd. uiuc. edu.
2) Pacific Sierra Research Corp. : An Interactive Tool for Vectorization and Parallelization of Fortran Programs (Jan. 1989).
3) Levesque, H. and Williamson, J. : A Guidebook to Fortran on Supercomputers, San Diego, CA : Harcourt Brace Javanovich (1989).
4) Kennedy, K., McKinley, K. S. and Chau-Wen Tseng : Analysis and Transformation in the ParaScope Editor, in Proc. ACM International Conf. on Supercomputing, Cologne, Germany (June 1991).
5) Kennedy, K., McKinley, K. S. and Chau-Wen Tseng : Interactive Parallel Programming Using the ParaScope Editor, IEEE Transactions on Parallel and Distributed Systems, Vol. 2, No. 3, pp. 329-341 (July 1991).
6) Allen, J. R., Baumgartner, D., Kennedy, K. and Porterfield, A. : PTOOL : A Semi-Automatic Parallel Programming Assistant, in Proc. 1986 Int. Conf. Parallel Processing, IEEE Computer Society Press (Aug. 1986).
7) Allen, J. R. and Kennedy, K. : PFC : A Program to Convert Fortran to Parallel Form, in Supercomputers : Design and Applications, IEEE Computer Society Press, pp. 186-205 (1984).
8) Cooper, K., Kennedy, K. and Torczon, L. : The Impact of Interprocedural Analysis and Optimization in the Rⁿ Programming Environment, ACM Trans. Programming Languages Syst., Vol. 8, No. 4, pp. 491-523 (Oct. 1986).
9) Kuck, D. J., Kuhn, R. H., Padua, D. A., Leasure, B. and Wolfe, M. : Dependence Graphs and

Compiler Optimizations, in Proc. 8th ACM Symp. Principles on Programming Language (Jan. 1981).
10) Burke, M. and Cytron, R.: Interprocedual Dependence Analysis and Parallelization, in Proc. SIGPLAN '86 Symposium on Compiler Construction, pp. 162-175 (July 1986).
11) Triolet, R. et al.: Direct Parallelization of Call Statements, in Proc. of the ACM SIGPLAN '86 Symposium on Compiler Construction, pp. 176-185 (June 1986).
12) Callahan, D. et al.: The Program Summary Graph and Flow-Sensitive Interprocedural Data Flow Analysis, in Proc. of the ACM SIGPLAN '86 Conf. on Programming Language Design and Implementation, pp. 47-56 (July 1988).
13) Balasundaram, V. and Kennedy, K.: A Technique for Summarizing Data Access and Its Use in Parallelism Enhancing Transformations, in Proc. of the ACM SIGPLAN '89 Conf. on Programming Language Design and Implementation, pp. 41-53 (June 1989).
14) Forgacs, I.: The Precise Determination of Define-Use Pairs in the Interprocedural Case, in IFIP (Sep. 1992).
15) Katz, R. H., Chang, E. and Bhateja, R.: Version Modelling Concepts for Computer Aided Design Databases, in Proc. 1986 ACM SIGMOD Int. Conf. on management of Data, Washington DC (May 1986).
16) Hans, P., Heinz-J. BAST and Michael GERNDT: SUPERB: A Tool for Semi-Automatic MIMD/SIMD Parallelization, Parallel Computing, Vol. 6, pp. 1-18 (1988).
17) Segall, Z. and Rundolpf, L.: PIE—A Programming and Instrumentation Environment for Parallel Processing, IEEE Software, Vol. 2, No. 6, pp. 22-37 (Nov. 1985).
18) Lehr, T., Segall, Z. F, Vrsalovic, D. F., Caplan, E., Chung, A. L. and Fineman, C. E.: Visualizing Performance Debugging, IEEE Computer, pp. 38-51 (Oct. 1989).
19) Heath, M. T., Etherridge, J. A.: Visualizing the Performance of Parallel Programs, IEEE Software, pp. 29-39 (Sep. 1991).
20) Express Programmer's Toolkit ver 2.0, Nippon Steel Corp. (1992).
21) Ikudome, K., Fox, G., Kolawa, A. and Flowers, J.: An Automatic and Symbolic Parallelization System for Distributed Memory Parallel Computers, in Proc. of the 5th Distributed Memory Computing Conf. (Apr. 1990).
22) Davies, J., Huson, C. T., Leasure, B. and Wolfe, M.: The KAP/205: An Advanced Source-To-Source Vectorizer for the S-1 Mark IIa Supercomputer, in Proc. of the 1986 International Conf. on Parallel Processing, IEEE Press, New York (1986).
23) 木村他：手続き間解析機能の検討，情報処理学会第 41 回全国大会，5-70 (1991).
24) 菊池他：並列化支援システム "Parassist" の試作—機能と構成—，情報処理学会第 44 回全国大会，6-85 (1992).
25) 飯塚他：並列化支援システム "Parassist" の試作—プログラム解析方法—，情報処理学会第 44 回全国大会，6-87 (1992).
26) 岩澤他：並列化支援システム "Parassist" の試作—並列化支援方法—，情報処理学会第 44 回全国大会，6-89 (1992).
27) 黒澤他：並列化支援システム "Parassist" の試作—手続き間並列性解析方法—，情報処理学会第 44 回全国大会，6-91 (1992).
28) 佐藤他：並列化支援システム "Parassist" の試作—不正並列化検出方法—，情報処理学会第 44 回全国大会，6-93 (1992).
29) 橋本他：並列化支援システム "Parassist" の試作—並列性検査方法—，情報処理学会第 44 回全国大会，6-95 (1992).
30) 佐々木他：並列化支援システム "Parassist" の試作—並列プロセス表示方法—，情報処理学会第 44 回全国大会，6-97 (1992).
31) 山本他：超並列計算機向けデータ分割の自動評価方式，並列処理シンポジウム JSPP '92 (1992).
32) Iitsuka: Flow-Sensitive Interprocedural Analysis Method for Parallelization, in Conf. on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism (Jan. 1993).

(平成 5 年 5 月 14 日受付)



菊池 純男（正会員）

昭和 27 年生．昭和 53 年東京工業大学大学院理工学研究科電気工学専攻修了．同年(株)日立製作所入社．現在，同社中央研究所ユーザインタフェース部主任研究員．言語コンパイラ，並列化支援システムの研究に従事．ACM, IEEE 各会員．