数 値 解 析 33－1
（１９９０．７．６）

並列スーパーコンピュータの支援ツール

成田良一
富士通研究所

階層記憶型並列スーパーコンピュータ上のソフトウェア，特に応用プログラムの開発支援ツールについて述べる．このツールの主な目的は「並列化」の支援である．支援ツールは実行解析ツールと静的解析ツールという２つのグループからなる．実行解析ツールは，実際に実行された過程の観測と評価のためのものである．一方，静的解析ツールは，ソースプログラムを解析してプログラムの書き換えや並列性の検証の支援を行なう．支援ツールというアプローチは，粒度の大きい並列処理に対して効果的な方法の一つである．

Support Tools on a Parallel Supercomputer

Ryoichi Narita
Artificial Intelligence Laboratory, Fujitsu Laboratories Limited,
1015, Kamikodanaka, Nakahara-ku Kawasaki, 211 JAPAN

In this paper, we describe a software system used on a parallel supercomputer with hierarchical memories. The system comtains support tools for developing application programs. The main purpose of our tools is to assist in parallelizing. Our support tools clasified into two groups: execution analyzers and static analyzers. Execution analyzers give information about actual execution. Static analyzers analyze source programs, and assisit to rewriting and verification of parallelism. Our approach of support tools is an effective way to the parallel processing with coarce-grain granurality.

1．はじめに

本稿では，階層記憶型並列スーパーコンピュータ上のソフトウェア，特に並列プログラムの開発支援システムについて述べる．

科学技術計算の分野においては，年々，大容量の記憶領域を要する計算を高速に実行することが必要となってきている．そのために多くの種類のスーパーコンピュータが開発されてきた．中でも，マルチプロセッサによる並列処理と階層構造記憶の活用は有効な手段の一つである．通商産業省工業技術院大型プロジェクト「科学技術用高速計算システムの研究開発」の一環として開発されたHPP(High-speed Parallel Processor)システムもこのようなアーキテクチャを持つ計算機である．

ベクトル計算機では粒度の小さい並列処理が課題となったが，このようなシステムでは，粒度の大きいサブルーチンレベルの並列処理が有効であり，これをソフトウェアの面から見ると二つの重要な課題が生じる．一つは，アプリケーションプログラムの並列化であり，もう一つはデータ配置である．これらの問題を解決するのに，コンパイラによる自動化とユーザーによる完全なプログラム記述という両極端のアプローチが考えられる．コンパイラによる自動並列化及びデータの自動配置に関しては，多くの重要な結果が得られてはいるものの(例えば，cf. [3])，完全な解決はまだ得られていない．逆に並列処理に関するこれらすべてのことをユーザーがプログラムに記述しなければならないとしたら，そのようなシステムは利用することが困難になるであろう．

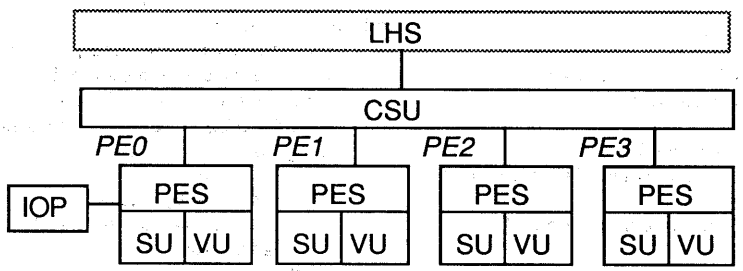
我々のアプローチは，並列プログラムの開発支援ツールを提供することで，システムとユーザーの距離を埋めようというものである．支援ツールを通じて，ユーザーが解析するには膨大すぎる情報をシステムが提供し，また逆にプログラムに記述してあることだけでは解析できない情報をユーザーがシステムに知らせる．その様な作業を通じて，より効率的なプログラム実行を行なわせることが可能となるであろう．もちろんこのようなアプローチは，コンパイラによる自動化と相反するものではなく，ツールの解析能力が増大してくれば，コンパイラへ吸収されていく性質のものである．

HPPシステムの支援ツールは，大きく２つのカテゴリーからなる．一つはソースプログラムを解析して，プログラム作成の支援を行なうものであり，「静的解析ツール」という．もう一つは実行後にシステムから得られるトレースデータを解析して，支援を行なうものである．これを「実行解析ツール」と呼ぶ．両者は相補って並列プログラムの開発を支援する．

2．HPPシステムの概要

HPPハードウェアシステムは，PESとCSUという２段階の階層的メモリを持ち，粗に結合された４台のPE(Processor Element)からなるマルチプロセッサシステムである．各PEは，それぞれスカラーユニットとベクトルユニットを持っている．HPPシステムは10GFLOPS以上の計算速度を持つ．図１にハードウェア構成の概略を示す．(cf. [1])

HPPハードウェアシステムは，同じプロジェクトの他のグループによって開発されたLHSと接続されている．



LHS: Large High-speed Storage
CSU: Common Storage Unit
IOP: Input Output Processor

PE: Processor Element
PES: PE Storage
SU: Scalar Unit
VU: Vector Unit

図1 HPPハードウェアシステムの構成

HPPソフトウェアシステムは，並列言語Philの処理系，プログラムの実行系，プログラム開発の支援系という3つのサブシステムから構成される．図2にソフトウェア構成の概略を示す．

全体は，Philの処理系を中心にして，ハードウェアとのインタフェースとして実行系があり，支援系がユーザーとのインタフェースを受け持つ．

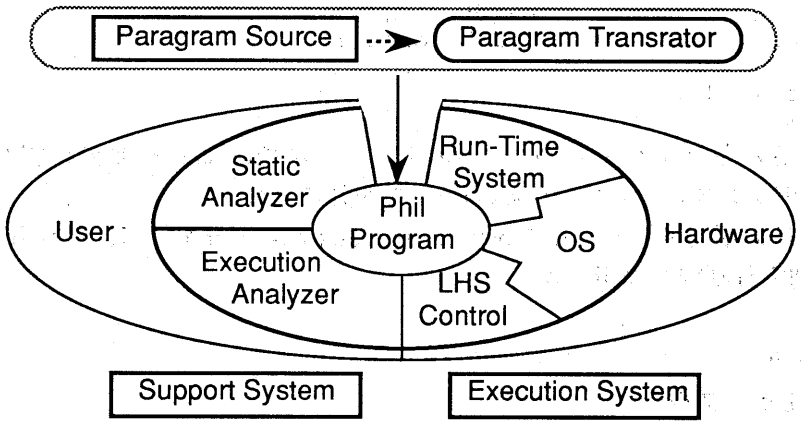


図2 HPPソフトウェアシステムの構成

3．Phil言語

Philは Fortran-likeな手続き型の言語の一種であるが，並列記述を持ち，階層記憶のアクセスを記述できように拡張されている．Philは HPPシステム上での効率的な実行と解析の容易さを目標として設計された．

並列処理の単位はPhilの表記では clusterと呼ばれるものである．図3にPhilプログラムの例を示す．この例では，プログラムは3つの clusterと共有データからなる．sumおよびmultというclusterが並列に実行される．

```
shared  DataBody
    variable
        a(10000)  :  integer
end  shared  DataBody

main  cluster  example
    procedure  readdata(a,size)
    ...
    procedure  writedata(s,p)
    ...
    procedure  example()
        use  shared  DataBody
        use  cluster sum,  mult
        variable
            size  :  integer
            s,p :  integer
    begin
        readdata(a,size)
        par
            call  sum(s,size)
            call  mult(p,size)
        end  par
        writedata(s,p)
    end  procedure  example
end  cluster  example
```

```
cluster  sum
    procedure  sum(s,size)
    use  sharedDataBody
    argument  s,size  :  integer
    variable  i  :  integer
    begin
        s = 0
        do  i=1,size
            s = s + a(i)
        end do
    end  procedure  sum
end  cluster  sum
cluster  mult
    procedure  mult(p,size)
        use  shared DataBody
        argument p,size  :  integer
        variable i  :  integer
    begin
        p = 1
        do  i=1,size
            p = p * a(i)
        end do
    end  procedure  mult
end  cluster  mult
```

図3　Phil プログラムの例

Phil の中心となる並列記述はpar~end par および pardo である．

```
par                     pardo i=1,n
    call clst1            call clst(i)
...
    call clst1
end par
```

共にfork-join型の並列実行を表わし，par はいくつかのclustrer を並列に実行するが，pardo はインデックスに応じて一つのclustrer を並列に実行する．

また，Phil は同期制御構文としてlock, barrier, reset-post-wait を持ち，これらを用いて低レベルの並列実行制御を表現できる．

4．支援ツール

4．1 並列化サイクル

支援ツールの主な目的は，アプリケーションプログラムの並列化の支援である．既存の多くのプログラムは逐次プログラムであり，それをHPPシステム上で走らせることが必要である．また，並列に記述されたプログラムであっても，特定の並列計算機上で実行する際には，チューニングの余地がある．

支援ツールが想定するユーザーの並列化手順は以下のようなものである．

(1) 作成 (Making)

ユーザーはPhilプログラムを作成する，あるいは他言語で書かれたプログラムをPhilに変換する．

(2) 実行 (Execution)

Philコンパイラにかけて，HPPシステム上で実行する．

(3) 観測 (Observation)

プログラムのどの部分に時間がかかっているかを調べる．

(4) 評価 (Evaluation)

プログラムのどの部分を分割すると効果が上がるかを評価する．

(5) 分割 (Partitioning)

ソースプログラムを分割して，並列記述を入れる．

(6) 検証 (Verification)

並列に実行しても逐次の場合と同じ結果が得られるかどうかを検証する．

(7) 決定 (Decision)

(6)の判定結果が肯定的であれば，(2)に戻って再び実行する．判定結果がNGであれば，(4)に戻って他の部分を並列化することを試みる．あるいは同期制御文を導入する．

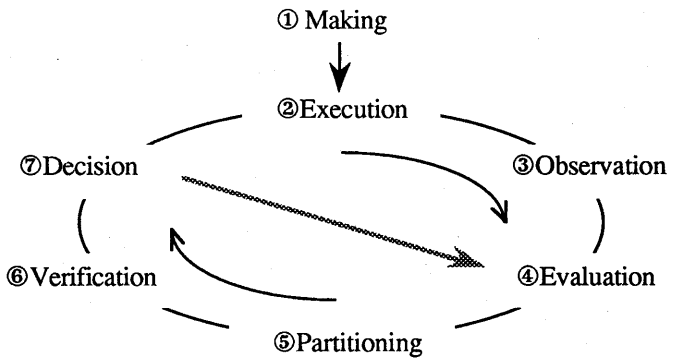


図4 並列化サイクル

以上の並列化の過程を「並列化サイクル」と呼ぶことにする．

HPPソフトウェアシステムの支援ツールは２つのグループからなる．実行解析ツールと静的解析ツールである．両者は合い補って，並列化の支援を行なう．実行解析ツールは並列化サイクルの(3)と(4)を支援し，静的解析ツールは，(5)と(6)を支援する．

4．2 実行解析ツール

実行解析ツールは，実際に実行された過程の観測と評価のためのものである．HPPシステムは，実行時にトレース情報を収集する．実行解析ツールがそのトレース情報を解析する．実行解析ツールは，「実行モニター」と「コスト解析ツール」という２つのサブツールからなる．

ユーザーはまず実行モニターを用いて実行のどの部分に時間を費やしているかを知る．実行モニターは，各PE上での計算実行のタイミングチャートをグラフィカルに表示する．タイミングチャート上には，各種のイベント（プロセスの生成，消滅，待機，データ転送のタイミングと転送量）が表示される．これにより，ユーザーは，どのプロセスにコストがかかっているかの概観を得ることができる．

実行コストがかかっているプロセスを発見すると，次にユーザーが行なうことは，そ

のプロセスの分割である．コスト解析ツールは，cluster呼び出しにかかるコスト，do文にかかるコスト，データ転送文にかかるコストを算出する．さらに，それをソースプログラムに対応させて表示する．またコストのソートなどの機能を持つ．ユーザーはこれを用いて，ソースプログラムのどの部分を並列化すれば効果的であるかを評価できる．

図5は実行解析ツールの動作画面の例である．

4．3 静的解析ツール

静的解析ツールは，静的解析ツールのターゲットは，ソースプログラムそれ自身である．静的解析ツールはソースプログラムのデータフロー解析を行ない，その情報を元にして，対話的にユーザーの並列プログラム開発の支援を行なう．

静的解析ツールも並列化サイクルのフェーズに従って，2つの部分にわかれる．(5)を支援するツールが「クラスタ化支援ツール」であり，(6)を支援するツールが「並列性検証ツール」である．

Philプログラムに対して，分割フェーズで行なうことは，ソースプログラムの逐次実行される断片をcluster呼び出しに変換し，そのようなcluster呼び出しを複数個集めて，par ~ end par （又は，元がdoループの本体であれば，pardo）で囲むという作業である．クラスタ化支援ツールは逐次実行される断片をcluster呼び出しに変換する作業の支援を行なう．いわゆるサブルーチン化と同様の作業である．

クラスタ化支援ツールの入力は，ソースプログラムおよびcluster化すべきそのプログラムの範囲である．元のプログラムはクラスタ化支援ツールによって，指定された範囲がcluster呼び出しに置き換えられる．また呼ばれるclusterが新たに生成される．この際に指定された範囲にある変数は，引数又は共有データとして受け渡される．これらの変数は，データフロー解析の結果により，新しいclusterに関してなるべく局所的になるように配置される．またこれは，ユーザーの指定によっても変更することが可能である．

クラスタ化と並列記述の挿入によって，並列実行するように書き換えられたソースプログラムは，ここで並列性検証ツールに渡される．並列性検証ツールは，逐次実行と結果が一致するかいなかを調べる．すなわち，par ~ end par あるいはpardoによって並列に実行されるクラスタ間で共有されるデータに対して，アクセス競合が起きないかどうかを検査する．ここでは，粒度の大きいclusterが対象であるため，手続き間にまたがる変数の追跡が必要である．

並列性検証ツールは，アクセス競合する可能性のある変数とそのアクセスの一および呼び出しパスを結果として帰す．ユーザーはこれを元にアクセス競合の解消を試みることになる．

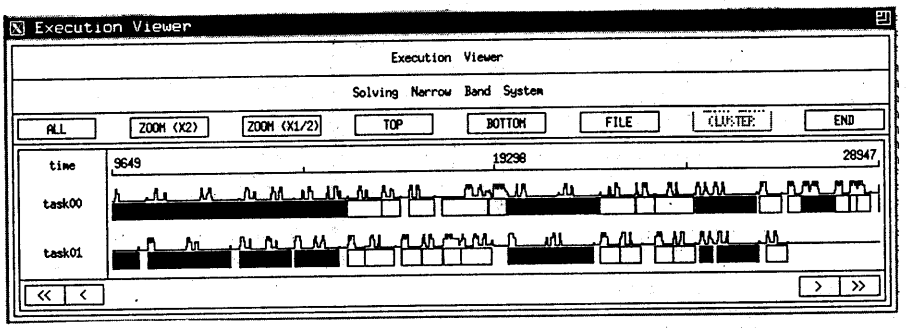
図6は静的解析ツールの動作画面の例である．

4．4 支援ツールの動作環境

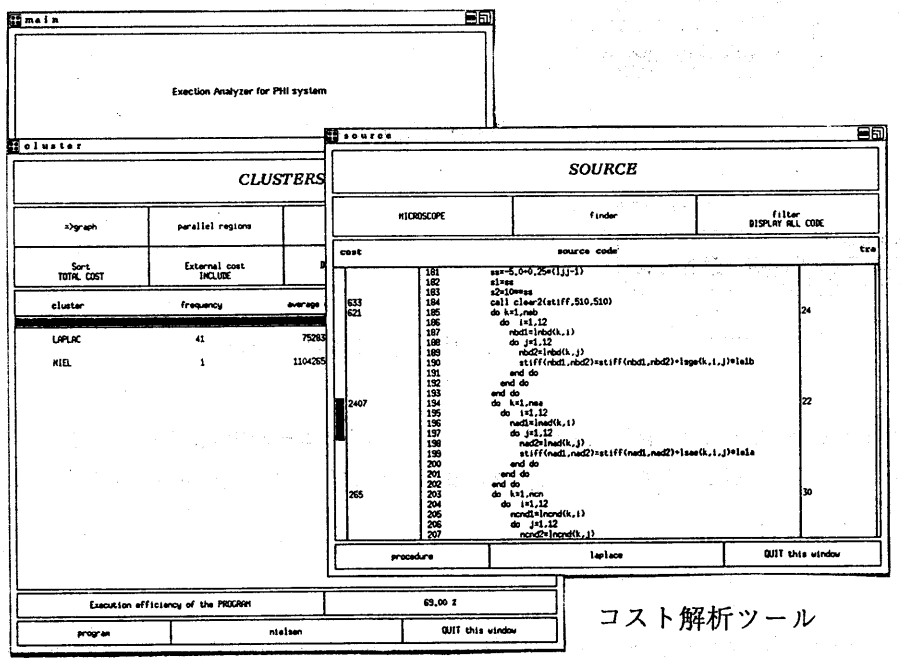
上記の支援ツールは，Sun4/110上で動作する．ユーザーインタフェースはX Window Systemを用いて作成した．HPPシステムのトレース情報は，フロントエンドのM780を経由して，LANにより転送される．実行解析ツールはCで書かれた．静的解析ツールは，各部分をCommon Lispで，ユーザーインタフェース部分をCで作成した．

5．おわりに

階層記憶型マルチ・プロセッサ上の並列処理では，粒度の大きい並列処理が必要とされる．支援ツールという我々のアプローチは，このような並列処理に対して，効果的な方法である．現在の支援ツールは，並列化サイクルという観点から構成されているが，その視点から見ても最小の構成である．トータルな支援ツール群の構成としては，並列化のみならず，従来の逐次プログラムの開発支援ツールも包含するものでなければならない．また，粒度の大きい並列性の認識をツールとしてもサポートしていく必要があるだろう．これらの研究を今後の課題としたい．

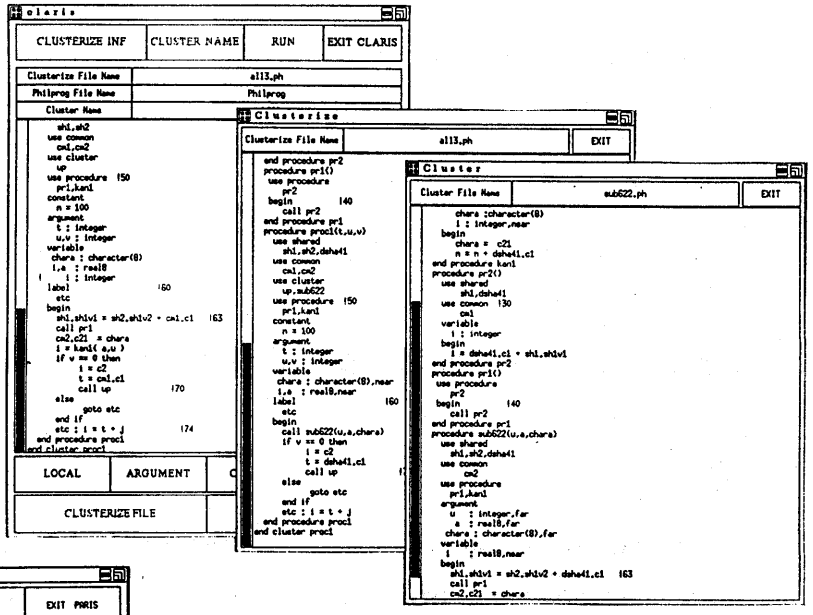


実行モニター

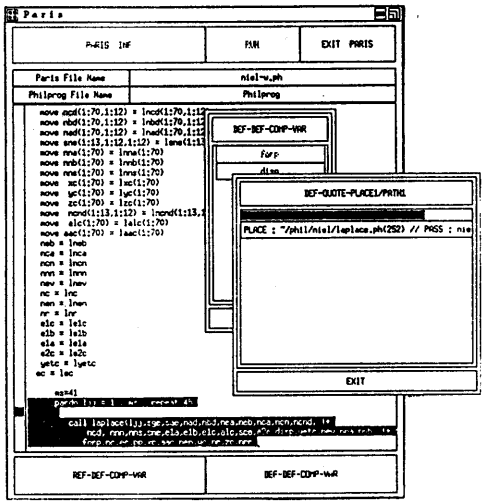


コスト解析ツール

図5　実行解析ツール



クラスタ化支援ツール



並列性検証ツール

図6　静的解析ツール

付記

本研究は通商産業省工業技術院大型プロジェクト「科学技術用高速計算システムの研究開発」の一環として新エネルギー・産業技術総合開発機構（NEDO）からの委託を受けて実施したものである.

References

[1] S. Hashimoto, S.Kamiya, N. Kasuya, N. Kuribayashi, R. Narita, Hierarchical Memory Connected Multi-Processor, 電子情報通信学会, コンピュータシステム研究会

[2] R. Narita, The Concept of Φ and Tools for Developing Parallel Programs, Info. Japan '90 (submitted)

[3] D. A. Padua and M. J. Wolfe, Advanced Compiler Optimizations for Supercomputers, Communication of the ACM, Vol. 29, No. 12, pp. 1184-1201(1986).