

JAERI-Data/Code
98-016



プログラム並列化支援解析ツールkpx(その2)

1998年3月

渡部 弘・折居茂夫・熊倉利昌*・滝川好夫*

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは、日本原子力研究所が不定期に公刊している研究報告書です。

入手の問合せは、日本原子力研究所研究情報部研究情報課（〒319-1195 茨城県那珂郡東海村）あて、お申し越しください。なお、このほかに財団法人原子力弘済会資料センター（〒319-1195 茨城県那珂郡東海村日本原子力研究所内）で複写による実費領布をおこなっております。

This report is issued irregularly.

Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, 319-1195, Japan.

© Japan Atomic Energy Research Institute, 1998

編集兼発行 日本原子力研究所
印 刷 いばらき印刷株

プログラム並列化支援解析ツール kpx (その 2)

日本原子力研究所計算科学技術推進センター

渡部 弘・折居 茂夫・熊倉 利昌*

滝川 好夫*

(1998年 2月 9日受理)

kpxは並列処理推進のための共通基盤として開発されたプログラム並列化支援解析ツールである。kpxはプログラムの実行時間及び実行回数を計測するktool、並列化オーバヘッドを計測するptool, xtool, mtool, vtool及びktool の結果をグラフ表示する kviewから構成される。kpxは全てのUNIX系コンピュータで動作するよう設計されており、現状で Paragon, SP2, SR2201, VPP500, VPP300, Monte-4, SX-4, T94において動作が確認されている。

The KPX, a Program Analyzer for Parallelization (ver2.0)

Hiroshi WATANABE, Shigeo ORII, Toshimasa KUMAKURA* and Yoshio TAKIGAWA*

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received February 9, 1998)

The kpx is a program analyzer, developed as a common technological basis for promoting parallel processing. The kpx consists of 6 tools. The 1st tool is ktool, that shows execution time and execution counts of user's program segment. The 2nd tool named kview is a post processor for ktool. It makes ktool's output visualize in X window terminal. And the others are ptool, xtool, mtool and vtool, and they measure parallelization overhead on various machines. The kpx is designed to work for any FORTRAN source code on any UNIX Computer, and it is confirmed to work well after testing on Paragon, SP2, SR2201, VPP500, VPP300, Monte-4, SX-4 and T94.

Keywords: Analyzer, Parallel Processing, Parallelization Overhead

* WOODLAND, Ltd.

目 次

1.はじめに	1
2.プログラム並列化支援解析ツールkpxの目的と構成	3
3. k tool	5
3.1 要求仕様	5
3.2 設計及び製造	5
3.3 利用方法	10
4. kview	14
4.1 要求仕様	14
4.2 設計及び製造	14
4.3 利用方法	15
5. m tool	18
5.1 要求仕様	18
5.2 設計及び製造	18
5.3 利用方法	20
6. kpxのバージョンアップ	24
6.1 kpxの評価	24
6.2 バージョンアップ仕様	25
7.おわりに	26
謝 辞	26
参考文献	26
付 錄	27

Contents

1. Introduction	1
2. Purpose and Constitution of kpx	3
3. ktool	5
3.1 Request Specifications	5
3.2 Design and Production	5
3.3 Usage	10
4. kview	14
4.1 Request Specifications	14
4.2 Design and Production	14
4.3 Usage	15
5. mtool	18
5.1 Request Specifications	18
5.2 Design and Production	18
5.3 Usage	20
6. Version up	24
6.1 Estimation of kpx	24
6.2 Version up Specifications	25
7. Conclusion	26
Acknowledgements	26
References	26
Appendix	27

1 はじめに

近年の科学技術計算分野における並列計算機の普及は目を見はるものがある。それはメーカーの努力によりハードウェアの価格性能比が著しく改善されたこと、またOSや言語の標準化が進みソフトウェアの開発環境が整いつつあることが原因と考えられる。すなわち科学技術並列計算のインフラストラクチャは急速に整備されつつあるといえる。今後のさらなる普及発展のためには、このインフラストラクチャの上で如何に多くの優秀な並列計算プログラムが開発されるかが鍵となるであろう。

並列計算機には大きく分けて共有メモリ型と分散メモリ型があり、それぞれ一長一短がある。共有メモリ型は並列化プログラミングが比較的容易であり、自動並列化コンパイラがサポートされている場合も多い。自動並列化コンパイラ技術は現在急速に進歩しており、近い将来、小規模の並列化においては十分ユーザの要求を満足するレベルに達すると考えられる。そうなれば、現存する多くの非並列プログラムをコンパイルし直すだけで並列化プログラムとして利用できるようになり、科学技術並列計算は爆発的に普及するものと期待されている。

しかし共有メモリ型の並列計算機はそのアーキテクチャ上、メモリを共有できるCPU数は比較的少数であり、大規模な並列化には適していない。従って現状の100～1000倍以上の性能向上が必要なプログラムのプラットホームとしては向きでない、その場合には分散メモリ型の並列計算機を採用せざるを得ない。実際、現状でピーク性能が1テラフロップスを超えるスーパーコンピュータは、例外なく分散メモリ型の並列計算機である。ところが分散メモリ型の自動並列化は技術的に非常に困難で、近い将来に分散メモリ型自動並列化コンパイラが実用化されることはずまず考えられない。従って分散メモリ型の並列化には“人手による並列化”はここ当面必要不可欠である。

以上のような状況下では、プログラムの並列化時にユーザの労力を少しでも軽減するツールの存在が重要である。例えばプログラムのどの部分を並列化すれば最も効果があるか、また通信時間はどれ位消費されているか等の情報が容易に得られれば、ユーザがプログラムを並列化しチューニングする際の有効な指針となりうる。

もちろんこの種のツールは並列計算機メーカ自体が開発し、提供している場合も少なくない。しかしメーカ提供のツールは当然のことながら、その並列計算機上でしか利用できない。従って他の並列計算機にプログラムを移植する場合、ユーザはツールの利用方法を新たに修得しなければならない。このことは並列プログラム開発者に大変な負担を強いるものであり、プログラム並列化を妨げる大きな原因となっている。そこで全ての並列計算機で全く同じ方法で容易に利用できる並列化支援ツールがあれば、非常に有用であろう。

このような背景より、日本原子力研究所計算科学技術推進センターは平成7年度からプログラム並列化支援解析ツール kpx を開発してきた。現在 kpx は、非並列化プログラムの実行時間及び実行回数を計測する ktool、ktool の計測結果をグラフ表示するための kview、並列化プログラムの通信時間及び通信回数を計測する ptool,xtool,mtool,vtool から構成されている。

本論文は前論文“プログラム並列化支援解析ツール kpx”(参考文献 1)の続編であり、平成 8 年 1 月以降の kpx 開発状況について説明する。この期間に行われた作業は ktool のバージョンアップ、kview の開発及び mtool と vtool の開発である。mtool は MPI(Message Passing Interface)により並列化されたプログラムの並列通信時間及び通信回数を計測するツール、vtool は PVM(Parallel Virtual Machine)により並列化されたプログラムの並列通信時間及び通信回数を計測するツールである。

本論文ではこれらのツールの要求仕様、設計と製造方法及び利用方法について報告する。第 2 章で kpx の構成及び役割を概観した後、第 3 章から第 5 章で ktool、kview 及び mtool の詳細を説明する。mtool のみを取り上げたのは mtool と vtool はその目的、設計と製造方法、利用方法がほとんど同じであり、さらに MPI が並列化通信インターフェースのデファクトスタンダードとなっていることによる。第 6 章では平成 10 年度に予定している kpx のバージョンアップについて説明する。

2 プログラム並列化支援解析ツール *kpx* の目的と構成

kpx は並列処理推進のための共通基盤として開発された、プログラム並列化支援解析ツールである。計測対象プログラムは fortran77 ベースのユーザプログラムである。*kpx* は以下の 6 個のツールから構成されている。

- (1)**ktool** : ユーザプログラムの実行時間及び実行回数をルーチン単位, do ループ単位, 入出力単位で計測するツール.
- (2)**ptool** : Paragon の NX ライブラリにより並列化したプログラムの並列通信時間及び通信回数を計測するツール.
- (3)**xtool** : VPP の並列化指示行により並列化したプログラムの並列通信時間及び通信回数を計測するツール.
- (4)**mtool** : MPI により並列化したプログラムの並列通信時間及び通信回数を計測するツール.
- (5)**vtool** : PVM により並列化したプログラムの並列通信時間及び通信回数を計測するツール.
- (6)**kview** : *ktool* の計測結果をグラフィカルユーザインターフェースにより、容易かつ視覚的にグラフ表示するためのポストプロセッサ.

ktool では全ての UNIX 系コンピュータの(非並列)fortran プログラムを対象としており、Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T94 において動作が確認されている。

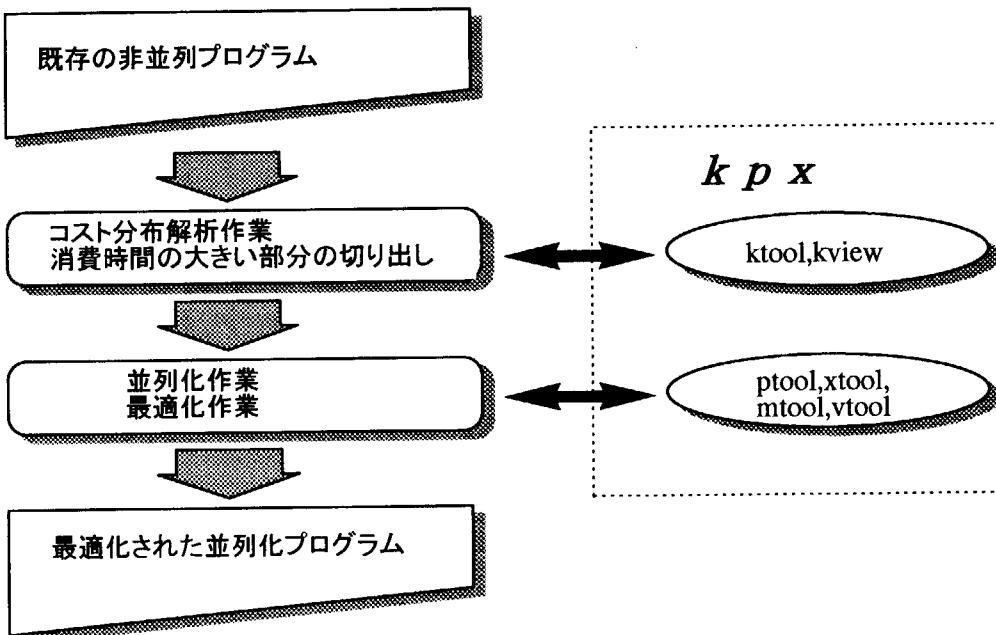


図1： *kpx* の役割

一方、各種機種及び通信インターフェースで並列化されたプログラムの並列通信時間及び通

信回数を計測するために ptool、xtool、mtool、vtool がそれぞれ用意されている。ユーザはこれらのツールを用いて並列化に伴うオーバヘッドを計測し、並列化プログラムのチューニングに役立てることができる。既存の非並列プログラムを並列化する場合に、どのようにこれらのツールが活用されるかを図示すると図 1 のようになる。

kpx の開発にあたっては以下のことを目標にした。

- 正確な計測結果を求める。
- 多くの計算機上で動作する。
- 計測のためのオーバヘッドができるだけ少なくする。
- ツールの操作性と透明性を高める。

ここでいうツールの透明性とは、ツールの動作の分かりやすさのことである。

kpx の開発経過は次の通りである。

- 平成 7 年 10 月 kpx(ktool,ptool,xtool) 開発開始
- 平成 8 年 10 月 日本原子力研究所計算科学技術推進センター内で仮公開
- 平成 8 年 11 月 ktool のバージョンアップ、mtool,vtool,kview の開発開始
- 平成 9 年 4 月 テストとバグフィックス及びドキュメントの整備
- 平成 9 年 6 月 日本原子力研究所内公開

次章以降でそれぞれのツールの要求仕様、設計製造及び利用方法について説明する。なお本論文の内容は参考文献 1 と重複する部分も多いが、その場合の説明は要点を記述するに留め、新たに追加した部分を詳述する。

3 ktool

3.1 要求仕様

ktool は非並列プログラムの実行時間及び実行回数を解析するためのツールである。その要求仕様は以下の通りである。

- プログラムの実行時間（経過時間あるいはC P U時間）をルーチン毎、do ループ毎、入出力毎に計測する機能を有する。
- プログラムの実行回数をルーチン毎、do ループ毎、入出力毎に計測する機能を有する。
- do ループの平均ループ実行回数を出力する機能を有する。
- 計測のためのオーバヘッドができるだけ少なくする。
- ユーザが指定した部分を計測する機能を有する。
- ツールのソースコードは fortran77 で作成し、fortran77 コンパイラを有する全ての並列計算機上で動作する。
- make コマンドに対応できる。

3.2 設計及び製造

3.2.1 ktool のポリシー

要求仕様でも述べた通り、ktool の製造は全て fortran77 言語で行う。この種のツール作成にはC言語もしくはC++言語を用いるのが一般的であろう。しかし我々は以下の理由で、これらの言語の採用を断念した。

- ktool インストール先の並列計算機にはCコンパイラが搭載されていない可能性がある。これはその並列計算機用のCコンパイラが存在しない場合や、Cコンパイラが有償であるためシステム管理者が導入していない場合が含まれる。
- C言語は標準化されているとはいえ、ANSI 版や K&R 版等があるように機種によって多少の違いがある。fortran77 ではそのような怖れが全くない。

以上のような理由から、多少生産性は落ちるもの、開発言語として fortran77 を採用した。しかしそれでも時刻取得ルーチンの機種による違いだけは如何ともしがたい。そこで我々は抽象的な ktool 用時刻取得ルーチンを定義し、この ktool 用時刻取得ルーチンがさらに機種固有の時刻取得ルーチンをコールするようにした。ktool はこの ktool 用時刻取得ルーチンを定義しているファイルを読み込み、kpx システムルーチンファイル @@f22.f を構成する(図2参照)。従って、ユーザは ktool 用時刻取得ルーチンファイルの 1 個所を変更するだけで、あらゆる機種で ktool による計測が可能となる。さらに動作確認済みの機種に対してはインストーラが ktool 用時刻取得ルーチンの書き換えを行うので、その場合ユーザは時刻取得ルーチン名の違いを全く意識する必要がない。

ktool のようなプログラムの各部分の実行時間や実行回数を計測するツールには、一般に 2 種類の設計手法が考えられる。一つはプログラムの要所要所に時間計測ルーチンや実行回

数カウントルーチンを挿入し直接計測する方法(直接的方法)であり、もう一つはプログラム実行時に一定の時間間隔でプログラムのどの部分が実行されているかという情報を取得し、それから実行時間や実行回数を間接的に推定する方法(間接的方法)である。

間接的方法は一般に計測のためのオーバヘッドが小さくできるという大きな長所がある。しかしコンパイラやOSに精通していないとツール開発は不可能である。今回開発したツールはできるだけ多くの計算機上で動作することを目的としているので、計算機の詳細な知識が必要となる間接的方法を採用することは得策でない。また間接的方法はあくまで実行時間や実行回数を統計的に推定するだけなので、正確な値を計測することは不可能である。

一方、直接的方法によると以下のようなメリットがある。

- 実行時間及び実行回数の計測値が正確である。
- ユーザの `fortran` ソースファイルに時間計測ルーチン及び実行回数計測ルーチンを埋め込めば良いので、コンパイラやOSに関する詳細な知識が必要ない。
- ツールが出力する計測用ソースコードがユーザにとって理解しやすい。すなわちツールの透明性が高い。
- ある特定部分を詳細に調べ、それ以外は計測しないというような利用方法が可能である。

以上より `ktool` の開発には直接的方法を採用した。しかし直接的方法を用いると、プログラムの多くの部分に時間計測ルーチンや実行回数カウントルーチンを挿入する必要があるので、どうしてもツールによるオーバヘッドが大きくなりかねない。この問題を解決するために、以下のような手法を用いた。

- プログラム実行時の `ktool` の中間計測情報は全てメモリ上に記憶し、途中経過のファイル出力は一切行わない。
- ベクトル計算機では最内側 `do` ループはコンパイラにより自動的にベクトル化される。従ってもし最内側 `do` ループに `ktool` の計測ルーチンを挿入するとベクトル化が阻害され、実行時間が大幅に増える可能性が高い。そこで最内側 `do` ループに `ktool` 計測ルーチンを挿入することは極力さける。
- 時間計測ルーチンが多数回コールされると非常に大きなオーバヘッドが生じる。そこでオプションにより `ktool` による計測を2回に分け、1回目の計測でプログラムの実行回数情報のみを取得し、その情報を元に2回目の計測で実行時間情報を取得することも可能にする。

以上から `ktool` の処理の流れは図2のようになる。

まずユーザが用意した `fortran77` プログラムに `ktool` を適用することにより、`ktool` 用の時間計測ルーチン及び回数計測ルーチンのルーチンコールを挿入し、オリジナルのユーザソースファイル名と同じ名前の `ktool` 計測用ソースファイルを作成する。同じ名前のファイルにすることにより、`make` コマンドを利用しているユーザはメイクファイルを大幅に変更す

る必要がない。あわせて ktool 用の回数計測ルーチン及び時間計測ルーチンの本体(ktool 用システムルーチンファイル@@f22.f)を作成する。ktool の動作制御は ktool 入力データファイル k_tool.dat を編集することによりなされる。

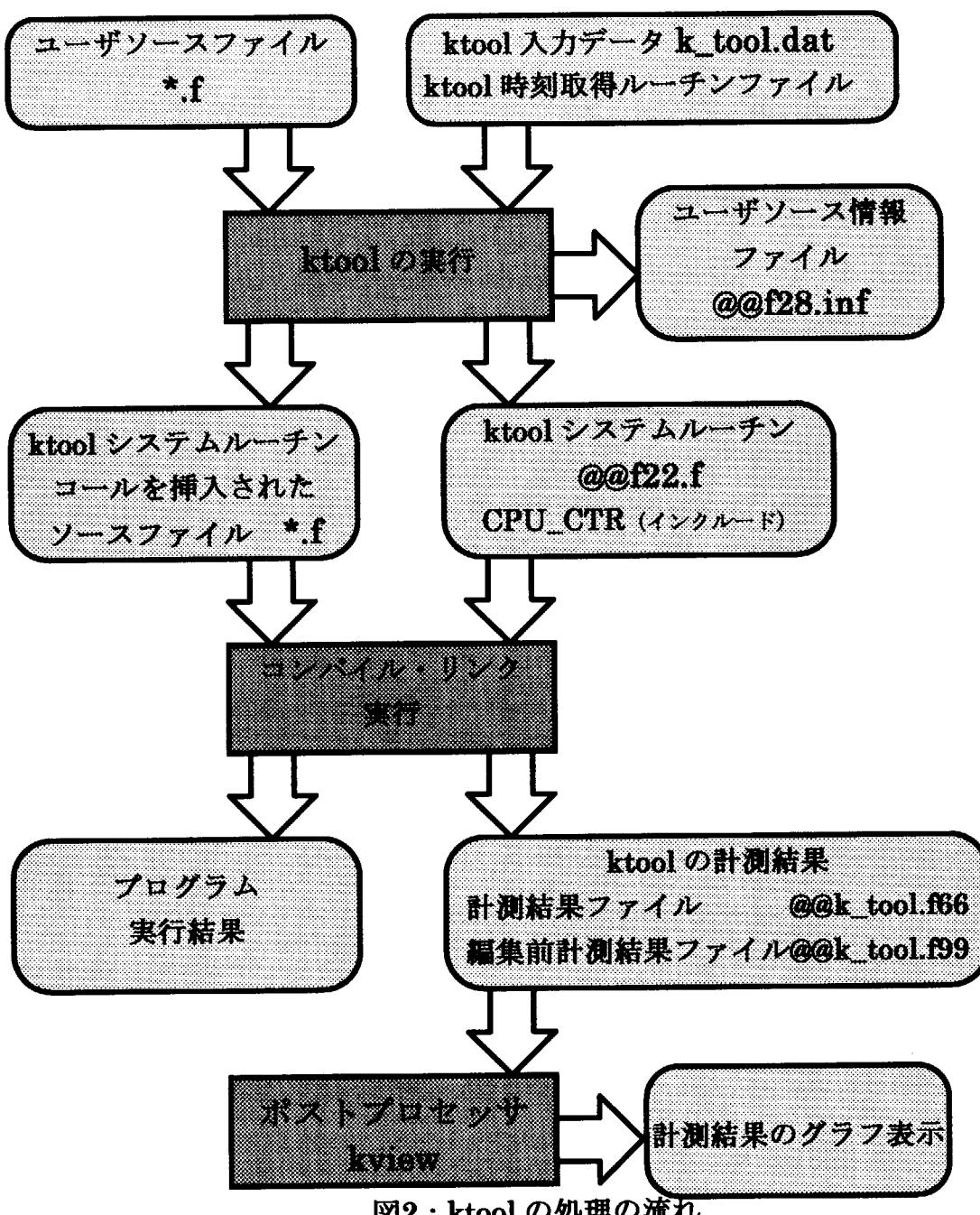


図2 : ktool の処理の流れ

次に ktool 用に修正された fortran ファイル及び ktool 用システムルーチンファイル @@f22.f をコンパイル,リンクし、実行モジュールを作成する。その後、この実行モジュールを通常通り実行させると、プログラム実行終了時に ktool 計測結果ファイル@@k_tool.f66 が 出力される。

ユーザはこの ktool 計測結果ファイル@@k_tool.f66 を見て、プログラムプロファイル情報を得る。また次章で説明する kview を用いて、グラフィカルユーザインターフェースにより簡単にプログラムプロファイル情報をグラフ表示することも可能にする。

3.2.2 メインルーチンの計測

メインルーチンに対して ktool は以下のことを行う必要がある。なお ktool はメインルーチンを program 文の存在により自動的に検知できる。

- メインルーチンの実行時間の計測する。
- メインルーチンの一番最初で ktool で用いられる各種変数の初期化や計測時間と計測回数を格納するための配列の初期化を行う。初期化ルーチンコールは実行文の先頭に挿入される。
- メインルーチンを終了する直前に計測結果を集計し、ユーザに分かりやすい形式でファイル出力する。そのため ktool 終了ルーチンはメインルーチンの最後と全ての STOP 文の前に挿入される。

3.2.3 ルーチン単位の計測

サブルーチンや外部関数(これらをルーチンと総称する)に対する計測には 2 種類の方法が考えられる。その第 1 はルーチンを呼ぶ側で計測を行う方法である。例えば以下のようなプログラムでサブルーチン sub1 の実行時間を計測するために、call 文の前後に計測ルーチンコールを行う。(図 3 参照)

```
program main
    → ここに計測ルーチンコールを挿入
    call sub1
    → ここに計測ルーチンコールを挿入
```

図 3 : ルーチン計測手法 (その 1)

この手法はサブルーチンを対象とする場合は直感的に分かりやすい。しかしあるサブルーチンが複数個所でコールされている場合には、経過時間と実行回数をそれぞれの個所で計測し、最後に合計しなければならない。また外部関数の場合にはこの手法はさらに問題が大きい。一つは外部関数にはサブルーチンに対する call 文に相当するシンボルがないので、自動的に計測ルーチンコールを挿入することが困難ということである。さらに外部関数の場合は 1 行に複数個の外部関数を呼ぶこともありうるので、ユーザソースプログラムの詳細な解析と大幅な書き換えが必要となる。

そのため我々は他の手法として、ルーチンの最初と最後で ktool システムルーチンをコールする方法を採用した(図 4 参照)。この方法によれば上記の難点は全て解決される。すなわちルーチンの最初の実行文の前に、実行回数のカウントアップ及びそのルーチンに入った

時刻を取得する ktool システムルーチンをコールする。またルーチン中の return 文の前に、そのルーチンの実行時間を計算する ktool システムルーチンをコールする。1 個のルーチンに return 文は複数個存在する可能性もあるので、その場合は全ての return 文の前で ktool システムルーチンをコールする必要がある。

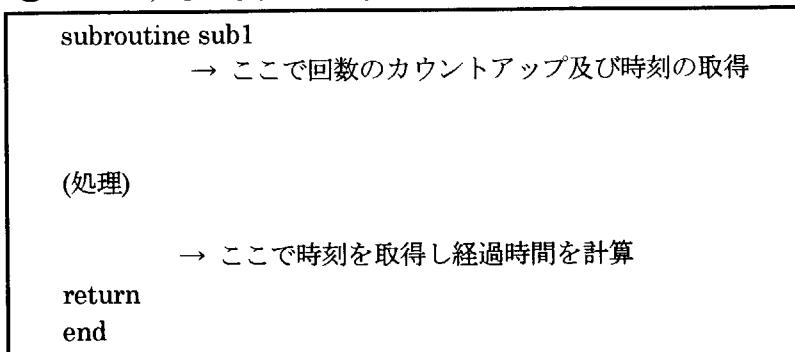


図4：ルーチン計測手法（その2）

3.2.4 do ループ単位の計測

do ループ単位では以下を計測する必要がある。

- do ループブロックの実行回数。
- do ループ内部の実行回数。これを回転数と呼ぶことにする。
- do ループ計算に要した時間。

do ループブロックの実行回数を計測するのは、その do ループブロックの直前で実行回数をカウントアップすればよいので問題無い。しかし回転数と計算時間を求めるのは注意を要する。

まず do ループを 2 種類に分類する。一つはその do ループの実行直前に回転数が確定できるタイプであり、もう一つは確定できないタイプである。前者を確定型 do ループ、後者を未確定型 do ループと定義する。未確定型 do ループの例としては do ループ内部に goto 文や return 文を含み do ループ外に制御が移る例が考えられる。

確定型 do ループの場合は do ループの直前と直後に時刻を計測することにより経過時間を計算できる（図5上参照）。また回転数も do ループの直前に決定できる。すなわち do ループ内部には ktool システムルーチンのコール文を挿入する必要がない。このことは特にベクトル計算機においては do ループのベクトル化を阻害しないという大きなメリットをもたらす。

未確定型の場合は do 文の直後に回転数を格納する変数のカウントアップを行う必要がある（図5下参照）。また経過時間を求めるために、do ループの直前直後及び do ループ内部の return 文や goto 文の直前にも時刻の取得を行わなければならない。このことは do ループ内部に ktool システムルーチンコール文が挿入されることを意味し、ベクトル化を阻害することになりかねない。しかし未確定型 do ループの場合はループ内部に return 文や goto

文があるため元々ベクトル化ができていない可能性が高いので、ktool 適用による大幅な時間増加の怖れは少ない。

```

→ ここで回数のカウントアップ、回転数の計算、時刻の取得
do I=1,n
    (処理)

end do
→ ここで時刻を取得し経過時間を計算

→ ここで回数のカウントアップ、時刻の取得
do I=1,n
    → ここで回転数のカウントアップ
    (処理)
    if (....) then
        (処理)
            → ここで時刻を取得し経過時間を計算
    return
    endif
end do
→ ここで時刻を取得し経過時間を計算

```

図 5 : do ループ計測手法 (確定型(上)と未確定型(下))

なお上記の例はブロック if 文の中に return 文がある場合であるが、算術 if 文や論理 if 文で分岐する場合も考えられる。そのため ktool は算術 if 文や論理 if 文をブロック if 文に一旦書き換えて ktool システムルーチンコールを挿入している。

3.2.5 入出力単位の計測

ファイル入出力に関しては実行回数及び経過時間を計測する必要がある。その計測値はユーザの便を考え、ファイル毎にも集計される。計測方法自体は read 文と write 文の前後に経過時間及び実行回数計測の ktool システムルーチンコールを挿入すればよい。

3.3 利用方法

3.3.1 インストール方法

kpx をインストールするために、動作確認済機種対応インストールシェルスクリプト kpxinst と汎用インストールシェルスクリプト custinst を用意した。動作確認済機種にインストールする場合は kpxinst を用いる方が簡単である。なお現状では、動作確認済機種は Paragon、SP2、SR2201、VPP500、Vpp300、Monte-4、SX-4、T94 である。これ以外の機種にインストールする場合は custinst を用いる。以下ではそれぞれの場合のインストール方法を説明する。

(kpxinst によるインストール方法)

- kpx のソースコード及びメイクファイルを含む資源ファイル(kpx.tar)を、対象コンピュータのホームディレクトリに取り込む。
- 対象コンピュータのホームディレクトリに kpx 用ディレクトリを作成し、kpx.tar をそのディレクトリ下で展開する。
- インストーラに機種名の引数を与えて起動し、kpx のロードモジュールを作成する。現状で受付可能な機種名引数は以下のうちのどれか一つ。インストーラは引数で与えられた機種の fortran コンパイラを起動し kpx 実行モジュールを作成する。さらにその機種の時刻取得ルーチンによる ktool 用時刻取得ルーチンファイルも作成する。

```
host> kpxinst HOST
      (HOST はホスト名 paragon,sp2,sr2201,vpp500,vpp300,monete4,sx4,t94)
          paragon : Paragon(インテル)用ロードモジュールを作成
          sp2 : SP2(I BM)用ロードモジュールを作成
          sr2201 : SR2201(日立)用ロードモジュールを作成
          vpp500 : VPP500(富士通)用ロードモジュールを作成
          vpp300 : VPP300(富士通)用ロードモジュールを作成
          sx4 : SX-4(N EC)用ロードモジュールを作成
          t94 : T94(C R A Y)用ロードモジュールを作成
          monte4 : MONTE-4(モンテカルロ専用コンピュータ)用ロードモジュールを作成
```

(custinst によるインストール)

- kpx のソースコード及びメイクファイルを含む資源ファイル(kpx.tar)を、対象コンピュータのホームディレクトリに取り込む。
- 対象コンピュータのホームディレクトリに kpx 用ディレクトリを作成し、kpx.tar をそのディレクトリ下で展開する。
- インストーラ custinst をエディタで編集し、コンパイラを指定する。custinst 内の “set KPXCOMP=f77” という行を探し、'f77'の部分を対応機種の fortran コンパイラ名に変更する。
- インストーラを起動し、kpx のロードモジュールを作成する。インストーラは ktool 用時刻取得ルーチンファイルの雛形も作成する。
- 生成された ktool 用時刻取得関数ファイルの雛形に、その機種固有の時刻取得ルーチンを書き込むことにより、その機種で動作する ktool 用時刻取得ルーチンを作成する。

3.3.2 利用の仕方

ktool の実行のためには利用者の便を考え、ktool 実行用シェルスクリプト krun が用意されている。ここでは krun を利用した ktool 実行方法について述べる。

- ktool を利用するにあたり、2 つの方針が考えられる。
- 時間コストと実行回数を同時に求める。
 - 時間コストと実行回数を別々に求める。すなわち 1 回目の実行で実行回数を求め、2 回

目の実行では実行時間を求める。

第1の方針が簡単であるが、プログラムの大部分に実行回数及び実行時間計測ルーチンを挿入するので、非常に時間がかかる場合もある。その場合には第2の方針に従う。第2の方針では1回目のktool実行でプログラムの各部分の実行回数を求め、そのデータに基づき2回目の時間コスト解析を行う。2回目の解析ではある一定回数(デフォルトは10000回)以上実行されている部分の計測は行わないので、比較的短時間で解析結果が得られる。計測されない部分の時間は上位のサブルーチンやdoループにその消費時間が加算される。

以下では2つの方針による解析方法について各自説明する。

(時間コストと実行回数を同時に求める方法)

- krun を用いて ktool 用のソースファイル及び@@f22.f を作成する。

```
host> krun -0 *.f
```

- 書き換えられたソースファイル及び新たに生成された@@f22.f をコンパイルし、実行モジュールを作成する。

```
host> f77 *.f ... *.f の中には@@f22.f も含まれる
```

- 作成された実行モジュールを実行する。

```
host> a.out
```

- 実行モジュールが終了すると同時に、時間コスト及び実行回数の計測結果を記した計測結果ファイル @@k_tool.f66 が出力される。

(時間コストと実行回数を別々に求める方法)

- krun を用いて実行回数計測用ソースファイル及び @@f22.f を作成する。

```
host> krun -1 *.f
```

- 書き換えられたソースファイル及び新たに生成された@@f22.f をコンパイルし、実行モジュールを作成する。

```
host> f77 *.f ... *.f の中には@@f22.f も含まれる
```

- 作成された実行モジュールを実行する。

```
host> a.out
```

- 実行モジュールが終了すると同時に、実行回数の計測結果を記した計測結果ファイル @@k_tool.f66 が出力される。

- 再度 krun を使用し、実行時間解析用ソースファイル及び @@f22.f を作成する。@@f22.f も新たに生成されることに注意する。ktool は前回の実行回数結果ファイルを参照しながら、実行回数が大きい部分の時間計測は行わない実行時間解析用ソースファイルを出力する。

```
host> krun -2 ... ソースファイルを指定する引数は必要ない
```

- 再度コンパイル、実行する。

```
host> f77 *.f      ... *.fの中には@@f22.f も含まれる  
host> a.out
```

- 実行モジュールが終了すると同時に、実行時間解析結果を記した **@@k_tool.f66** が出力される。

最後に計測結果の確認方法であるが、**ktool** では計測結果ファイル **@@k_tool.f66** による方法と **ktool** 用ポストプロセッサ **kview** による方法がある。**@@k_tool.f66** については付録で説明することとし、**kview** による方法を次章で詳述する。

4 k v i e w

4.1 要求仕様

kview は ktool の計測結果をグラフィカルユーザインターフェース(以下ではG U Iと略す)により、容易かつ視覚的にグラフ表示するためのポストプロセッサである。以下にその要求仕様を記述する。

- ktool の出力結果を容易に読み込める機能を有する。
- できるだけ多くの機種上で動作する。特に計算科学技術推進センターが所有する全ての並列計算機上で動作すること。
- ルーチン単位、do ループ単位、入出力単位の実行時間を棒グラフにして表示する。
- ルーチン単位、do ループ単位、入出力単位の実行回数を棒グラフにして表示する。
- 操作はマウス入力を中心とし、キーボード入力は必要最小限にする。

4.2 設計及び製造

kview の開発にあたり、まず計算科学技術推進センターの所有する並列計算機のグラフィックス開発環境を調査した。その結果、全ての並列計算機でCコンパイラとXライブラリがサポートされていることが判明した。そこで我々は kview をCとXライブラリで開発することとし、より便利なG U I構築ツールである motif 等はあえて利用しないことにした。

G U Iで最も重要なのは画面間の遷移関係である。これが直感的に分かりやすいものであれば、そのG U Iは使いやすいものとなる。kview の画面遷移関係は図 6 の通りである。

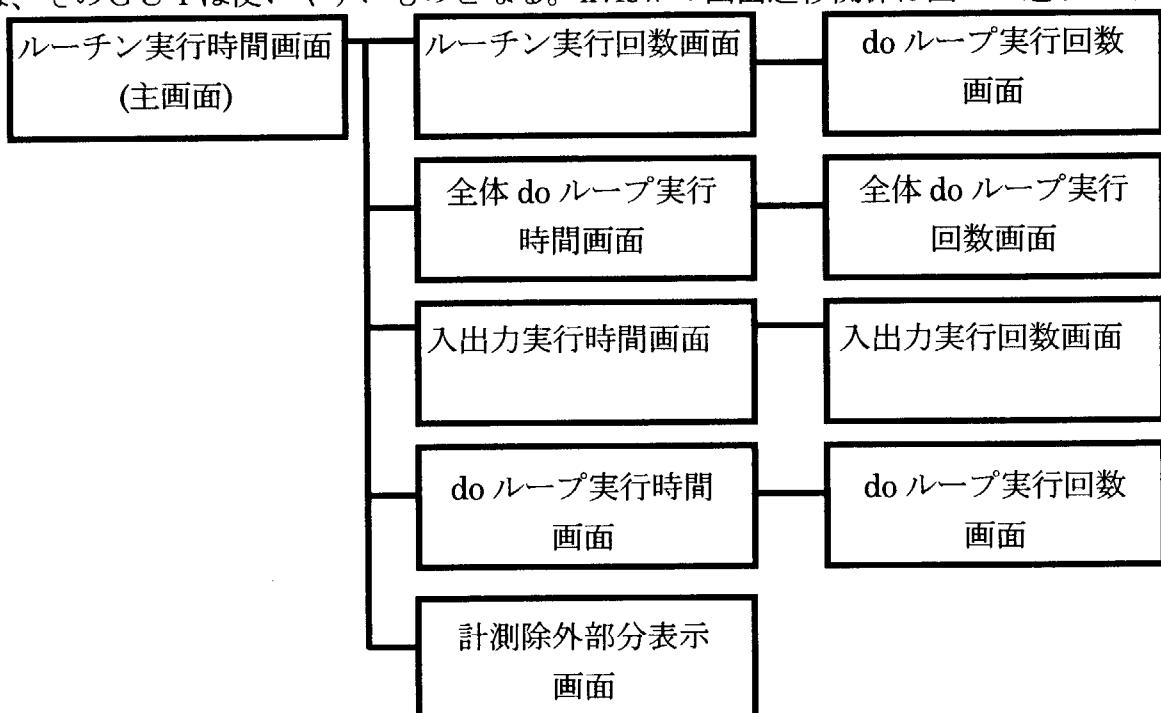


図 6 : kview 画面の遷移関係

kview を起動するとルーチン実行時間画面が表示される。これは kview の主画面であり、プログラム全体でのルーチン実行時間が表示される。主画面の「File」メニューで計測結果ファイルの読み込みや kview の終了を行う。また「View」メニューで do ループ単位や入出力単位の計測結果を表示する画面に遷移し、「Ignored」メニューで計測除外したルーチン、do ループ、入出力の一覧を表示する。さらに「Execution Times」ボタンでプログラム全体でのルーチン実行回数が表示される。この「Execution Times」ボタンは各実行時間画面に存在し、その機能は各時間コスト画面で共通している。

4.3 利用方法

4.3.1 インストール方法

kview をインストールするために、動作確認済機種用のメイクファイル及び汎用メイクファイルを用意した。動作確認済機種にインストールする場合はその機種用のメイクファイルを用いてコンパイルを行えばよい。一方それ以外の機種では汎用メイクファイルを編集の上コンパイルする必要がある。編集すべき部分は主にグラフィックライブラリの種類とその格納場所である。以下で動作確認済機種の場合のインストール方法を説明する。

- kpx のソースコード及びメイクファイルを含む資源ファイル(kview.tar)を、対象コンピュータのホームディレクトリに取り込む。
- 対象コンピュータのホームディレクトリに kview 用ディレクトリを作成し、kview.tar をそのディレクトリ下で展開する。
- 対象コンピュータ用のメイクファイルでコンパイルを行う。

4.3.2 利用の仕方

対象計算機のコマンドラインから kview と入力し計測結果ファイルを指定すると、主画面であるルーチン実行時間表示画面が現れる(図 7 参照)。「Automatic Scale」「Manual Scale」ボタンは、グラフの表示スケールの種別を選択するものである。前者はグラフが上下にスクロールされた時、画面の最上段に表示されるグラフが最大値になる様常に動的にスケールが変化する状態を指定する。後者は画面の表示状態に関わらず、ユーザの指定値がグラフの最大値になる様スクロールによりスケールが変化しない状態を指定する。これらは全ての画面で共通である。「Manual Scale」ボタンがマウスでクリックされた場合には、スケールの最大値をキーボードから入力する。

画面左下にある「Execution Times」ボタンを押すと新たに実行回数画面が表示される(図 8 参照)。実行回数画面にも「Automatic Scale」と「Manual Scale」ボタンが存在する。この画面を消去するには「Close」ボタンを押す。

またルーチン名か棒グラフのところをマウスでクリックすることにより、ルーチン内 do ループ時間コスト画面が表示される(図 9 参照)。

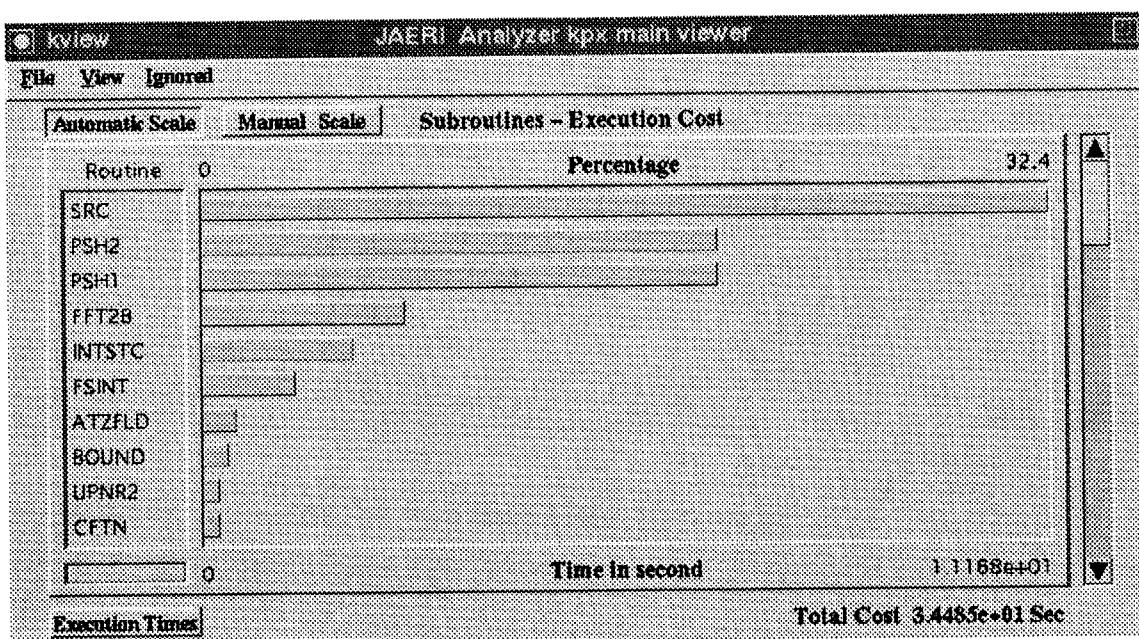


図 7：実行時間画面（主画面）

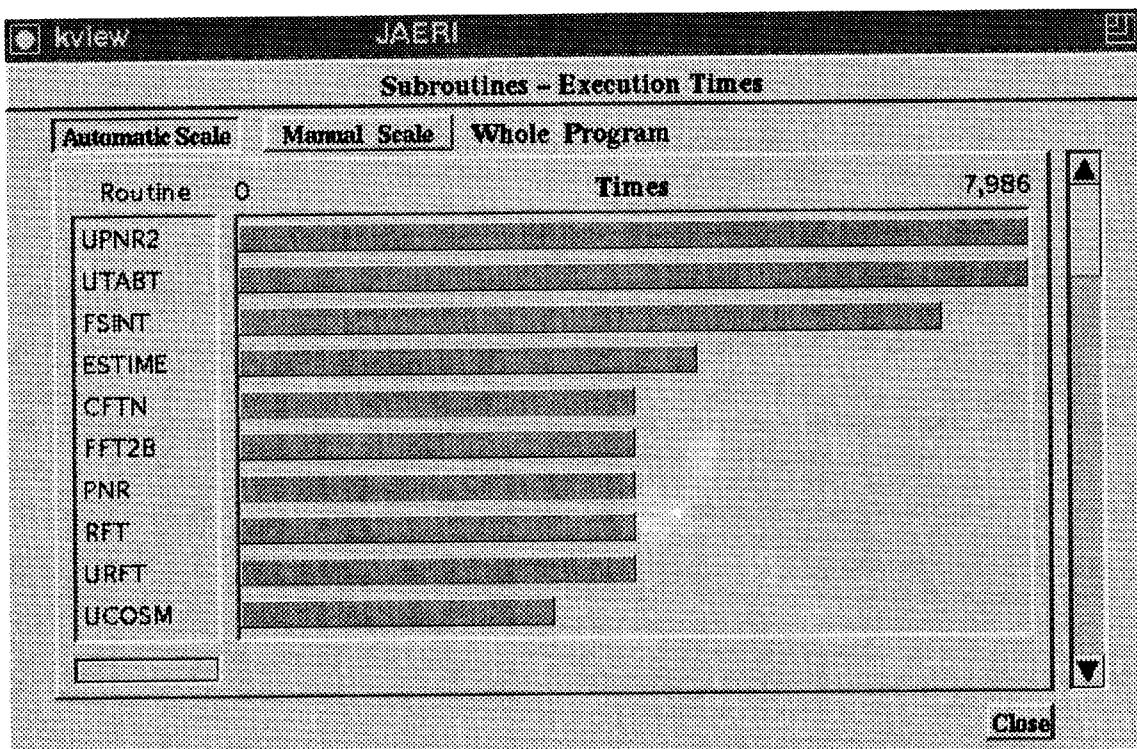


図 8：実行回数画面

主画面には「File」 「View」 「Ignored」 の各メニューが存在する。「File」 メニューには「Open」 と「Quit」 があり、計測結果ファイルの指定及び kview の終了を行う。

「View」 メニューには、「Do Loops」 と「Input/Output」 があり、前者はプログラム全体の do ループ実行時間結果(図 9 参照)、後者はプログラム全体の入出力実行時間結果の画

面が表示される。両者ともほとんど同じ表示形式である。do ループ実行時間画面については後で詳述する。

「Ignored」メニューは除外対象になったルーチン、do ループ及び入出力の一覧を表示するためのものである。

図9はプログラム全体での do ループ時間コストの画面である。do ループ時間コスト画面には、プログラム全体を対象に表示する画面と特定のルーチン内のループを対象に表示する画面の2種類があり、前者は主画面の「View」メニューから、後者は主画面のルーチン名部分もしくは棒グラフ部分をマウスピックすることにより表示できる。両者とも全く同じ表示形式である。図の例はプログラム全体の do ループを対象としている。画面下の「Execution Times」ボタンが押されると、プログラム全体での do ループ実行回数(do ループの回転数ではなく、その do ループ単位が何回実行されたかを示す)の画面が表示される。「Close」ボタンが押されるとこの画面が閉じられる。

do ループの表示画面ではそれが存在しているルーチン名と、その do ループが記述されている fortran ソースのファイル内ライン位置とそのファイル名が表示される。ルーチン名、ファイル名またはライン位置や棒グラフをマウスの左ボタンでクリックする事によりその部分のソースコードを確認できる。また同様の場所でマウスの右ボタンをクリックすることでループの総回転数や平均回転数の画面が表示される。この画面はマウスのボタンが押されている間のみ表示される。

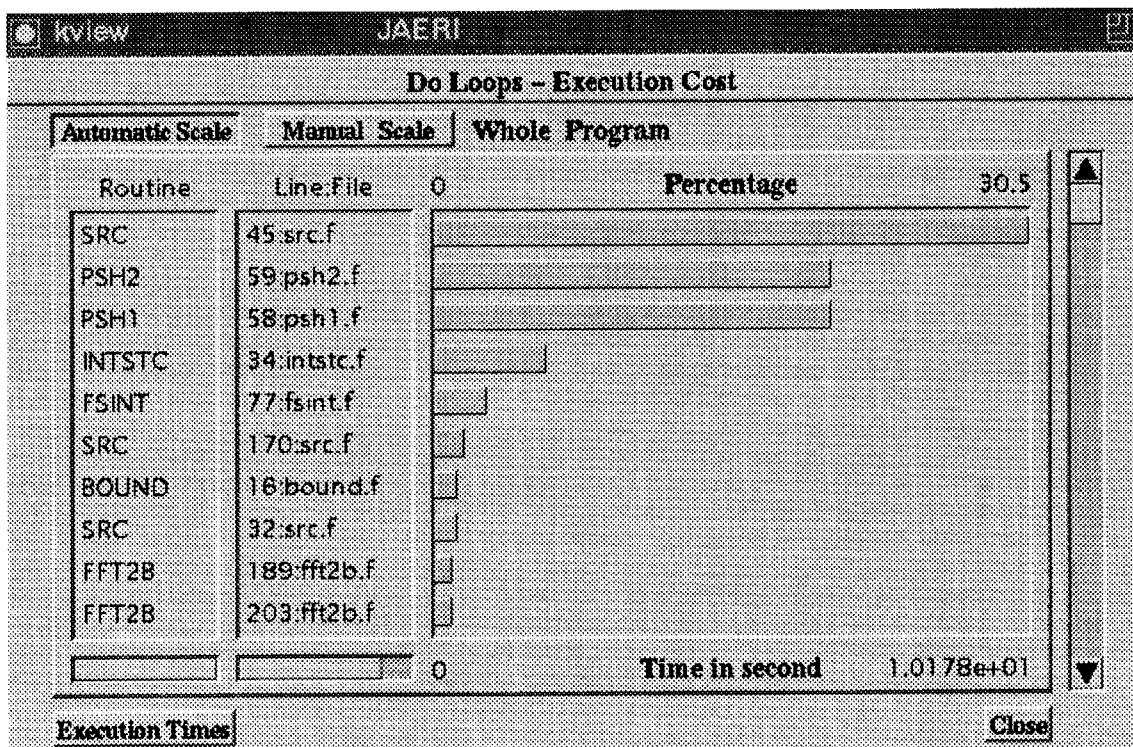


図 9 : do ループ実行時間画面

5 mtool

5.1 要求仕様

mtool は MPI(Message Passing Interface)による並列プログラムの並列化通信時間及び通信回数を解析するためのツールである。その要求仕様は以下の通りである。

- プログラムの MPI サブルーチンの実行時間(経過時間)を計測する機能を有する。
- プログラムの MPI サブルーチンの実行回数を計測する機能を有する。
- 計測のためのオーバヘッドをできるだけ少なくする。
- ユーザが指定した部分を計測する機能を有する。
- ツールのソースコードは fortran77 で作成し、fortran77 コンパイラを有する全ての並列計算機で動作する。
- make コマンドに対応できる。

5.2 設計及び製造

5.2.1 mtool のポリシー

上記要求仕様を見ても分かる通り、mtool は ktool とほぼ同様に設計できる。しかも以下の理由により、ktool より作成しやすいといえる。

- MPI における通信は全て、MPI ライブラリルーチンをコールする形でなされる。従って、ユーザプログラム中の”call mpi_xxxx”という部分の実行時間及び実行回数を計測すればよい。
- 本来通信は必要最小限行うべきものであり、通信の実行回数が甚だしく多く行われることは少ない。そこで全ての通信を計測することにしてもそのオーバヘッドは許容範囲内と考えられる。
- MPI には時刻取得用ルーチンとして mpi_wtime が標準でサポートされている。

mtool はユーザプログラムの通信部分(”call mpi_xxxx”という部分)の前後に実行時間計測ルーチン及び実行回数計測ルーチンを挿入する事により、並列化通信時間及び通信回数を計測する。(図 10 参照)

```
program main
```

```
    → ここと
```

```
    call mpi_allreduce(.....)
```

```
    → ここに計測ルーチンコールを挿入
```

図 10 : 通信部分の計測手法

この方法によるメリットは以下の通りである。

- 通信時間及び通信回数の計測値が正確である。
- ユーザの fortran ソースファイルに通信計測ルーチン及び通信回数計測ルーチンを埋め

込めば良いので、コンパイラやOSに関する詳細な知識が必要ない。

- ツールが output する計測用ソースコードがユーザにとって理解しやすい。すなわちツールの透明性が高い。

さらにできるだけ高速に計測結果を得るために、ktool と同様に中間計測情報は全てメモリ上に記憶し、途中経過のファイル出力は一切行わないようにした。一方 ktool で高速化のために採用した実行回数と実行時間を別々計測するオプションは、上で述べたように通信の場合には必要性が小さいと考え、mtool では採用しなかった。

以上から mtool の処理の流れは図 11 のようになる。

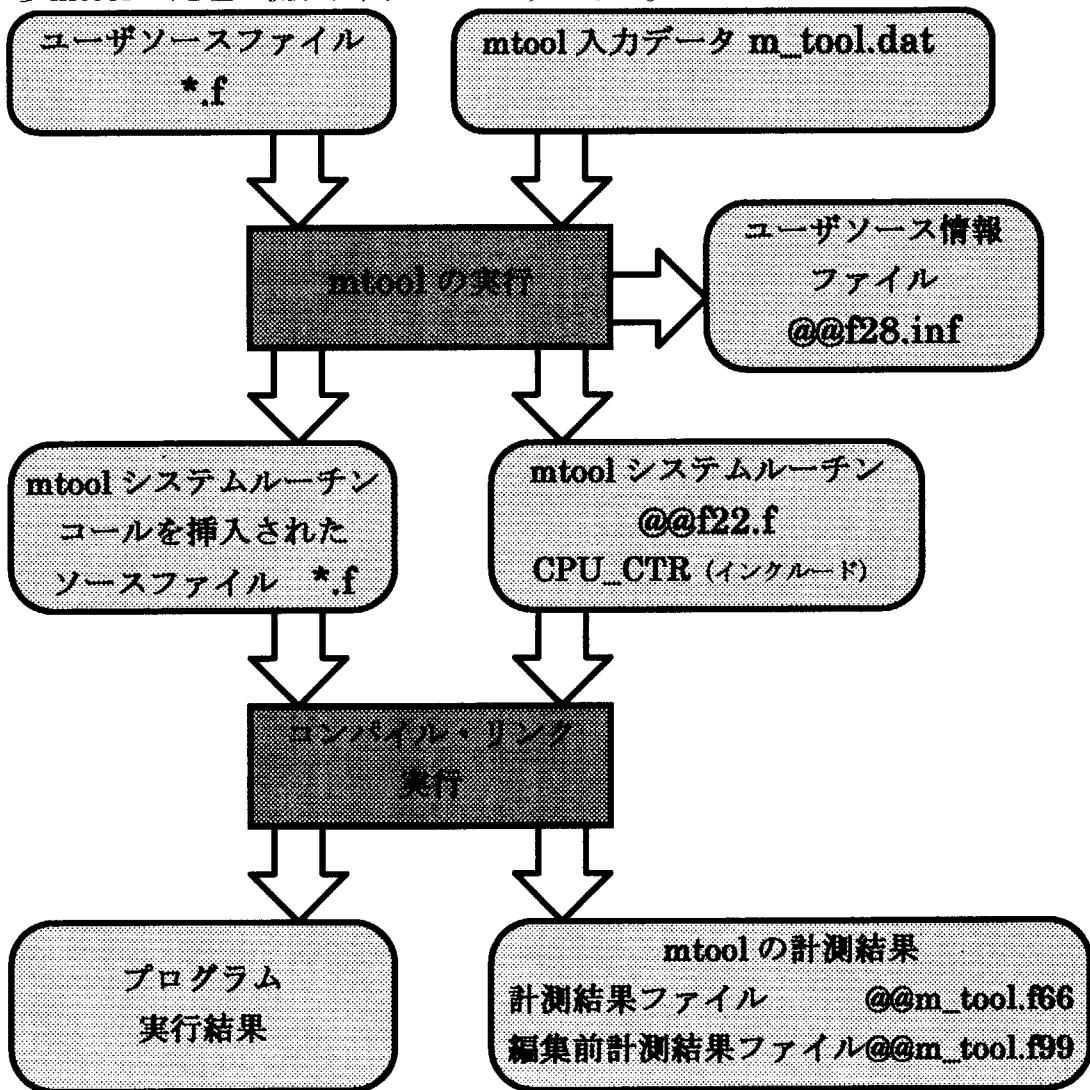


図 11 : mtool 処理の流れ

まずユーザが用意した fortran77 プログラムに mtool を適用することにより、mtool 用の時間計測ルーチン及び回数計測ルーチンのコール文を挿入し、オリジナルのユーザソースファイル名と同じ名前の mtool 計測用ソースファイルを作成する。同じ名前のファイルにす

ることにより、make コマンドを利用しているユーザはメイクファイルを大幅に変更する必要がない。あわせて mtool 用の時間計測ルーチン及び回数計測ルーチンの本体(mtool 用システムルーチンファイル@@f22.f)を作成する。mtool の動作制御は mtool 入力データファイル m_tool.dat を編集することによりなされる。

次に mtool 用に修正された fortran ファイル及び mtool 用システムルーチンファイル @@f22.f をコンパイルリンクし、実行モジュールを作成する。その後、この実行モジュールを通常通り実行させると、プログラム実行終了時に mtool 計測結果ファイル@@m_tool.f66 が出力される。

ユーザはこの mtool 計測結果ファイル@@m_tool.f66 を見て、プログラムプロファイル情報を得る。残念ながら kview に相当するポストプロセッサは、mtool では開発されていない。

5.2.2 メインルーチンの計測

メインルーチンに対して mtool は以下のことを行う必要がある。なお mtool はメインルーチンを program 文の存在により自動的に検知する。

- メインルーチンの経過時間の計測する。
- メインルーチンの一番最初で mtool で用いられる各種変数の初期化や計測時間と計測回数を格納するための配列の初期化を行う。初期化ルーチンは実行文の先頭に挿入される。
- メインルーチンを終了する直前に計測結果を集計し、ユーザに分かりやすい形式でファイル出力する。そのため mtool 終了ルーチンはメインルーチンの最後と全ての STOP 文の前に挿入される。その際、mtool 終了ルーチンでは全プロセスの計測結果をマスタプロセスに通信し、マスタプロセスの終了ルーチンのみが結果をファイル出力するようにならなければならない。

なお mtool ではルーチン単位、do ループ単位、入出力単位の計測は仕様にないので、それらの設計は必要ない。

5.3 利用方法

5.3.1 インストール方法

mtool は kpx をインストールすることにより利用できる。kpx のインストール方法は 3.3.1 を参照されたい。

5.3.2 利用の仕方

mtool の実行のためには利用者の便を考え、mtool 実行用シェルスクリプト mrun が用意されている。ここでは mrun を利用した mtool 実行方法について述べる。

- mrun を用いて mtool 用のソースファイル及び@@f22.f を作成する。

host> mrun *.f

- 書き換えられたソースファイル及び新たに生成された@@f22.f をコンパイルし、実行モジュールを作成する。

host> f77 *.f ... *.f の中には@@f22.f も含まれる

- 作成された実行モジュールを実行する。

host> a.out

- 実行モジュールが終了すると同時に、時間コスト及び実行回数の計測結果を記した計測結果ファイル @@m_tool.f66 が出力される。

最後に計測結果ファイル @@m_tool.f66 をエディタ等で見る事により結果を確認する。計測結果ファイルは5のつ部分で構成されている。

- 計測個所一覧
- オリジナルソースファイル名一覧
- 各プロセッサ毎の計測結果
- オーバーヘッドの合計情報
- プロセッサ数による通信時間の平均値

mtool 計測結果ファイルの内容を説明して、本章を終える

(1) 計測個所一覧

プログラムのどの部分を計測したかについての情報である。

<<< MPI:A Message-Passing Interface library Cost List >>>					
Number of nodes	:	4	(実行時のプロセッサ数)		
MPI call lines	:	88	(計測個所の数)		
eq. Subprogram name loc.			MPI call name (ルーチンの種類)	(ファイル名と行数)	
-----*	-----1-----*	-----	-----	-----	-----
1 CALUVW	1		MPI_BARRIER	Subroutine	28 caluvw.f
2 CALUVW	2		MPI_WTIME	Function	29 caluvw.f
(途中省略)					
4 MSOLVE	85		MPI_BARRIER	Function	54 msolve.f
5 MSOLVE	86		MPI_WTIME	Subroutine	47 para_range.f
1 PARA_RANGE_SET	87		MPI_ABORT	Subroutine	53 msolve.f
1 SPLIT	88		MPI_ABORT	Subroutine	57 split.f

(2) オリジナルソースのファイル名一覧

ユーザソースファイル名に通し番号を与える。

```
File# filename
 1 calcul.f
 2 calcv1.f
 3 calcw1.f
      (途中省略)
18 setbnd.f
19 split.f
```

(3) 各プロセッサ毎の計測結果

各計測箇所の統計情報(通信時間の合計と通信回数)を出力する。

seq.	sub	loc.	MPI name	File#	Line.	Node	time(sec)	counts
-----	-----*	-----1-----	-----2-----	-----*	-----3-----	-----	-----	-----
1	CALUVW	1	MPI_BARRIER	4	28	1	0.002037 (Maximum)	
						3	0.000196 (Minimum)	
							0.001046 (Mean)	
						1	0.002037 (1)	
						2	0.001520 (1)	
						3	0.000196 (1)	
						4	0.000433 (1)	
2	CALUVW	2	MPI_WTIME	4	29	3	0.000003 (Maximum)	
						1	0.000002 (Minimum)	
							0.000002 (Mean)	
						1	0.000002 (1)	
						2	0.000002 (1)	
						3	0.000003 (1)	
						4	0.000002 (1)	

(4) オーバーヘッドの合計情報

各 MPI ルーチン毎の平均値や、全プロセッサでの処理時間の合計情報が出力される。

TOTAL	Elapsed Time	
	1.492285 second.	(プログラム全体の実行時間)
TOTAL	OVERHEAD	node total time(sec)
		---# ---

4	0.345916	(Maximum)
3	0.324624	(Minimum)
	0.336051	(Mean)
1	0.347007	
2	0.326657	
3	0.324624	
4	0.345916	

(5) プロセッサ数による通信時間の平均値

subprogram name	Line.	MPI call name	Avarage (sec)	ratio%
-----*-----1-----*-----2-----*-----3-----	-----	-----	-----	-----
CALUVW	28	MPI_BARRIER	0.001107	0.33
CALUVW	29	MPI_WTIME	0.000002	0.00
CALUVW	30	MPI_WTIME	0.000006	0.00
CALUVW	39	MPI_BARRIER	0.000190	0.06
(途中省略)				
BMT_FLOW	26	MPI_COMM_SIZE	0.000001	0.00
BMT_FLOW	27	MPI_COMM_RANK	0.000001	0.00
BMT_FLOW	32	MPI_OP_CREATE	0.000007	0.00
BMT_FLOW	33	MPI_OP_CREATE	0.000013	0.00
BMT_FLOW	34	MPI_OP_CREATE	0.000005	0.00
MSOLVE	43	MPI_BARRIER	0.000000	0.00
MSOLVE	44	MPI_WTIME	0.000000	0.00
MSOLVE	45	MPI_WTIME	0.000000	0.00
MSOLVE	53	MPI_BARRIER	0.000000	0.00
MSOLVE	54	MPI_WTIME	0.000000	0.00
PARA_RANGE_SET	47	MPI_ABORT	0.000000	0.00
SPLIT	57	MPI_ABORT	0.000000	0.00

6 kpx のバージョンアップ

6.1 kpx の評価

平成9年6月に kpx を日本原子力研究所内で公開して以来、約10名のユーザが kpx を利用している。その評価を確かめるためユーザにアンケート調査を行った結果、以下のようなことが分かった。

- インストールは全員が1時間以内に終了しており、作業は容易もしくは普通という評価であった。ただし、センターの機種に対しては運用者がインストールした kpx をみんなで利用する形態が望ましいとの意見があった。
- 実際に利用されたツールは ktool、mtool、kview であった。並列化オーバヘッド計測ツールで mtool のみが利用されているのは、MPI が並列化通信インターフェースのデファクトスタンダードとなっていることの反映と考えられる。
- 計測対象プログラムは fortran77 であるにも関わらず、計測不能なユーザプログラムはなかった。kpx が fortran77 の拡張仕様に積極的に対応してきたことと、ユーザプログラムに fortran90 の文法がまだあまり浸透していないことが理由と思われる。
- 全員が ktool の機能は十分であるとの評価。kview に関してもほとんどが十分という回答であった。
- 一方、mtool の機能は不十分という回答が多かった。その内容はロードバランスに関する情報が得られない、ktool のようなルーチン単位、do ループ単位の情報もほしい、送受信の対応がわからないというもの。
- 計測精度については全員が信頼できる、あるいはある程度信頼できるという回答であった。精度的にはおおむね良好な結果が得られていると考えられる。
- 計測結果の理解しやすさについては、ktool と mtool については理解しにくいという意見が多く、kview については理解しやすいという意見が多かった。
- 使い勝手についても ktool と mtool については使いにくいという意見が、kview については使いやすいという意見が多かった。ktool、mtool では計測に時間がかかりすぎる、全てのソースプログラムに毎回ツールを適用しコンパイルし直すのはたいへんとの指摘があった。
- ドキュメント類に関してはほとんどが理解しやすいという評価であった。
- 今後サポートしてほしい言語としては fortran90 と C が挙げられた。

ktool についての評価は以下のようにまとめられる。

- 機能や精度は十分であるが、やや使いにくい。
- 計測結果の見やすさにもやや難があるが、kview の利用により改善されている。

一方、mtool については機能、使いやすさ、計測結果の見やすさの全てについて改善の必要がある。特に機能の不足は大きな問題である。例えば ktool 並みの計測単位が実現され

ば、プロセス間のロードバランスを知ることも可能となり、その意義は大きいと考えられる。またG U Iによるポストプロセッサも不可欠であろう。

6.2 バージョンアップ仕様

上記の評価を踏まえ、我々は平成10年度に kpx とりわけ mtool のバージョンアップを予定している。その要求仕様は以下の通りである。

- mtool の機能強化を行い、通信時間と通信回数の他に、各プロセッサ毎のルーチン単位、do ループ単位、入出力単位の計測を可能にする。
- 計測の際のオーバヘッドを現状よりさらに小さくする。オリジナルのプログラム実行時間の10%程度のオーバヘッドで上記全データが計測できることを目標とする。
- 一部分のユーザプログラムのファイルにツールを適用しても、そのファイル内のルーチンに関しては計測可能にする。言い換えれば、全部のユーザプログラムファイルにツールを適用しなくても良いようにする。
- 各ルーチンの実行時間と実行回数の計測と共に、そのルーチン中の通信単位、サブルーチン call 単位、do ループ単位、入出力単位の計測を行う。すなわち、各ルーチンの実行時間の内分けを計測できるようにする。
- fortran90 によるユーザプログラムに対応する。
- バージョンアップされた mtool 用の G U I ポストプロセッサを開発する。

上記の中で特に一部分のユーザファイルにツールを適用可能とする仕様は、kpx の実用性や利便性を飛躍的に高めるものと期待される。しかしその実現には周到で注意深い設計が必要である。それは kpx の認識できる資源がプログラムの全資源とは限らないため、ツールの配列や変数をどこで確保し初期化するか、どの部分で計測結果を出力すればよいかが明らかでないからである。

またツールによるオーバヘッドを最小限にすることも使い勝手の面で重要である。基本的には計測範囲を小さくすれば、計測オーバヘッドも少なくなる。ユーザにとって必要十分な計測範囲の設定が重要である。

以上のような検討を中心に、現在我々は kpx バージョンアップ版の設計を行っている。今後の計画としては平成10年度上期に fortran77 対応版及びG U I ポストプロセッサの開発を行い、平成10年度下期に fortran90 対応を行う予定である。

7 おわりに

本論文では並列化支援解析ツール kpx の現状及び今後の計画について説明した。現在 kpx は日本原子力研究所内で利用されているが、平成 10 年度には WWW を通じて外部に提供することを予定している。科学技術分野における並列計算とはある意味で非常に狭い世界であり、kpx のようなツールはその必要性はあっても商業的には成立しにくい。従って当センターが率先して並列計算機の機種によらない並列化支援ツールを開発し、それを広く公開することは科学技術分野の並列計算にとって非常に有意義と考えている。

今後もより多くのユーザの意見を積極的に取り入れ、よりよい並列化支援ツールを開発していく所存である。本ツールが科学技術並列計算を志す研究開発者に利用され、その研究開発に少しでも役立つなら、筆者のこれに過ぎる喜びはない。

謝 辞

日本原子力研究所計算科学技術推進センターの方々にはモニタをお願いし、多数の貴重な意見を頂きました。また同センターの相川裕史次長には kpx 開発にあたり多大のご支援を終わりました。ここに感謝の意を表します。

参 考 文 献

- (1) 松山雄次 折居茂夫 大田敏郎 久米悦雄 相川裕史, “プログラム並列化支援解析ツール kpx”, JAERI-Tech 97-017
- (2) T94:Guide to Parallel Vector Applications SG-2182 2.0
- (3) SP2:IBM Parallel Environment for AIX:Operation and Use Version 2.1.0
- (4) SP2:AIX Version 4.1 Commands Reference
- (5) SP2:User's Guide Parallelware-FORTRAN-8X20-3-541(E)
- (6) SR2201:HI-UX/WE2 リアルタイムパフォーマンスマニタ for SR2201
- (7) SR2201:HI-UX/WE2 アプリケーションチューニングツール for SR2201
- (8) SR2201:The FORGE Explorer TOOLBOX The Distributed Memory Fortran Parallelizer FORGEX/DMP User's Guide Version2.0
- (9) SR2201:FORGE Explorer The complete Fortran Program Browser On UNIX/Motif Workstations
- (10) FORGEX Version2.0 User's Guide

(付録) ktool 計測結果ファイルの説明

<<< PROGRAM UNIT RESULTS >>> ... サブルーチン単位の計測結果

PROGRAM	TIMES	TIME:SEC	% LINE FILE/*.f
<hr/>			
SRC	600	9.983 33.0	2 src.f
PSH2	300	6.882 22.7	2 psh2.f
PSH1	297	6.786 22.4	2 psh1.f
INTSTC	4	1.701 5.6	3 intstc.f
REVERS	524288	1.280 4.2	4 revers.f
USINM	28744	0.697 2.3	39 usinm.f
VELSET	131072	0.589 1.9	3 velset.f
BOUND	600	0.377 1.2	3 bound.f
CFTN	3993	0.330 1.1	64 cftn.f
FSINT	7186	0.304 1.0	39 fsint.f
(中略)			
UNIFRM	0	0.000 0.0	2 unifrm.f
<hr/>			
30.285		... 合計	

(各項目の説明)

PROGRAM	: サブルーチン単位名(プログラム名 サブルーチン名 外部関数名)
TIMES	: 実行回数
TIME:SEC	: 実行時間(単位 秒)
%	: 全計測時間に対する比率
LINE	: 行番号
FILE/*.f	: 記述されているファイル名

<<< I/O PROCESS RESULTS >>> ... 入出力単位の計測結果

PROGRAM SEQ.	UNIT NO.	KIND	TIMES	TIME:SEC	% LINE	FILE/*.f
<hr/>						
POTOTC	6	UNIT NO. write	192	0.017	8.9	86 pototc.f

POTOTE	6	UNIT NO.	write	192	0.017	8.8	88	potote.f
POTOUT	6	UNIT NO.	write	192	0.017	8.6	89	potout.f
HISTORY	1	UNIT NO.	write	20	0.016	8.5	46	histry.f
ELDAT	3	UNIT NO.	write	2	0.013	6.6	55	eldat.f
ELDAT	4	UNIT NO.	write	2	0.013	6.6	56	eldat.f
PLOT1	12	UNIT NO.	write	63	0.009	4.9	55	plot1.f
ENERGY	1	UNIT NO.	write	10	0.009	4.9	196	energy.f
PLOT1	10	UNIT NO.	write	63	0.009	4.8	51	plot1.f
		(中略)						
VELSET	1	UNIT NO.	write	0	0.000	0.0	33	velset.f
				0.193	...	合計		

(各項目の説明)

PROGRAM	:	サブルーチン単位名(プログラム名 サブルーチン名 外部関数名)
SEQ.	:	出現順位
UNIT NO.	:	FORTRAN 論理機器番号
KIND	:	入出力の種別
TIME:SEC	:	実行時間(単位 秒)
%	:	全計測時間に対する比率
LINE	:	行番号
FILE/*.f	:	記述されているファイル名

<<< DO LOOP RESULTS >>>

... DO LOOP 単位の計測結果

PROGRAM	SEQ.	LABEL	T	ROTATION	MEAN:ROT	TIMES	TIME:SEC	%
LINE								
FILE/*.f								
PSH2	2	20	1	13107200	512.0	25600	6.852	22.4
PSH1	2	20	1	12976128	512.0	25344	6.757	22.1
SRC	3	30	1	26214400	512.0	51200	4.854	15.9
SRC	6	50	1	209715200	128.0	1638400	3.748	12.3
SRC	4	90	1	26214400	512.0	51200	0.986	3.2

INTSTC	1	100 1	131072	65536.0	2	0.738	2.4	34	intstc.f
FFT2B	9	210 1	26048880	16.0	1628055	0.716	2.3	89	fft2b.f
SRC	8	70 1	189337600	29584.0	6400	0.556	1.8	170	src.f
FFT2B	30	520 1	43525152	16.0	2720322	0.553	1.8	205	fft2b.f
FFT2B	27	510 1	43525152	16.0	2720322	0.447	1.5	191	fft2b.f
SRC	1	10 1	189337600	473344.0	400	0.397	1.3	32	src.f
BOUND	1	10 1	26214400	65536.0	400	0.375	1.2	16	bound.f
		(中略)							
USINM	26	310 1		0	0.0	0	0.000	0.0	389 usinm.f
USINM	27	320 1		0	0.0	0	0.000	0.0	398 usinm.f
<hr/>									
					30.573	...	合計		

(各項目の説明)

PROGRAM : サブルーチン単位名(プログラム名 サブルーチン名 外部関数名)
 SEQ. : 出現順位
 LABEL : DO LOOP のラベル
 T : DO LOOP の回転数種類(1:確定型 0:未確定型 下記注意(1)参照)
 ROTATION : 総回転数(下記注意(2)参照)
 MEAN:ROT : 平均回転数
 TIMES : 実行回数
 TIME:SEC : 実行時間(単位 秒)
 % : 全計測時間に対する比率
 LINE : 行番号
 FILE/*.f : 記述されているファイル名

<<< SOURCE FILE NAME(S) >>> ... ソースファイル情報

FILE	SOURCE FILE NAME
-----*-----1-----*-----2-----*-----3-----*-----4-----*-----5.	
1 main.f	
2 src.f	
3 psh0.f	
4 psh1.f	

5 psh2.f

(後略)

(注意)

- (1) DO LOOP にはループの実行直前に回転数(反復数)がわかる場合(確定型)と、
 ループ内に GOTO 文、 RETURN 文、 STOP 文があるためわからない場合(未確定型)
 がある。ktool は回転数を計測するためのカウンタを、確定型ではループの
 直前に、未確定型ではループ内に挿入する。

- (2) 総回転数とは DO LOOP 内ブロックが実行された回数である。例えば

```
do i = 1,100
    sum = sum + a(i)
enddo
```

という DO LOOP が 10 回実行されれば、総回転数は 1000 (= 100*10) である。
 また、平均回転数 = 総回転数 / 実行回数 である。

- (3) 特定のサブルーチンや DO LOOP で時間計測が抑制された場合、
 上位のサブルーチンや DO LOOP にその消費時間が加算される。

(ディレクトリ及び各種ファイル)

インストールが正常に終了すると、ディレクトリ kpx 下に以下のような
 ファイルが生成される。mtool,vtool はそれぞれ MPI,PVM が利用できるマ
 シンの場合に作成される。

kpx/	(kpx directory)
DESIGN	;kpx 各種ツールの内部構造
K_TREE	;ktool のツリー構造図
M_TREE	;mtool のツリー構造図

V_TREE	;vtool のツリー構造図
README.english	;最初に読むべきテキストファイル(英語)
README.euc	;最初に読むべきテキストファイル(日本語 euc コード)
README.jis	;最初に読むべきテキストファイル(日本語 jis コード)
README.sjis	;最初に読むべきテキストファイル(日本語シフト jis コード)
kdo/	;ktool テスト用ディレクトリ
kpx.tar	;kpx 用 tar ファイル(インストール後は不要)
kpx_man.euc	;kpx オンラインマニュアル(日本語 euc コード)
kpxinst*,kpx	用インストールシェル(インストール後は不要)
ktool/	;ktool システムディレクトリ
mdo/	;mtool テスト用ディレクトリ
mtool/	;mtool システムディレクトリ
vdo/	;vtool テスト用ディレクトリ
vtool/	;vtool システムディレクトリ

kpx/ktool/ (ktool system directory)

k_tool.hlp	;ヘルプファイル
k_tool0.dat	;ktool 制御データ(実行回数と時間コスト同時計測)
k_tool1.dat	;ktool 制御データ(実行回数計測)
k_tool2.dat	;ktool 制御データ(時間コスト計測)
krun*	;ktool 実行用シェルスクリプト
ktool*	;ktool 実行モジュール
myclock.dat	;ktool システム用データ(CPU 時間計測)
mysystem.dat	;ktool システム用データ(経過時間計測)

kpx/mtool/ (mtool system directory)

m_clock.dat	;mtool システム用データ(各機種固有の時計で計測)
m_system.dat	;mtool システム用データ(MPI_WTIME で計測)
m_tool.dat	;mtool 制御データ
m_tool.hlp	;ヘルプファイル
mrun*	;mtool 実行用シェルスクリプト
mtool*	;mtool 実行モジュール

kpx/vtool/ (vtool system directory)

v_system.dat	;vtool システム用データ(各機種固有の時計で計測)
v_tool.dat	;vtool 制御データ

v_tool.hlp	;ヘルプファイル
vrun*	;vtool 実行用シェルスクリプト
vtool*	;vtool 実行モジュール

<<< For the Paragon. >>>

kpx/ptool/	(ptool system directory)
p_system.dat	;ptool システム用データ
p_tool.dat	;ptool 制御データ
p_tool.hlp	;ヘルプファイル
prun*	;ptool 実行用シェルスクリプト
ptool*	;ptool 実行モジュール

<<< For the VPP. >>>

kpx/xtool/	(xtool system directory)
x_system.dat	;xtool システム用データ
x_tool.dat	;xtool 制御データ
x_tool.hlp	;ヘルプファイル
xrun*	;xtool 実行用シェルスクリプト
xtool*	;xtool 実行モジュール

以上

国際単位系(SI)と換算表

表1 SI基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表3 固有の名称をもつSI組立単位

量	名称	記号	他のSI単位による表現
周波数	ヘルツ	Hz	s ⁻¹
力	ニュートン	N	m ² kg/s ²
圧力, 応力	パスカル	Pa	N/m ²
エネルギー, 仕事, 熱量	ジュール	J	N·m
工率, 放射束	ワット	W	J/s
電気量, 電荷	クーロン	C	A·s
電位, 電圧, 起電力	ボルト	V	W/A
静電容量	ファラード	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m ²
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	°C	
光束度	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m ²
放射能	ベクレル	Bq	s ⁻¹
吸収線量	グレイ	Gy	J/kg
線量等量	シーベルト	Sv	J/kg

表2 SIと併用される単位

名 称	記 号
分, 時, 日	min, h, d
度, 分, 秒	°, ', "
リットル	L
トン	t
電子ボルト	eV
原子質量単位	u

$$1 \text{ eV} = 1.60218 \times 10^{-19} \text{ J}$$

$$1 \text{ u} = 1.66054 \times 10^{-27} \text{ kg}$$

表5 SI接頭語

倍数	接頭語	記号
10 ¹⁸	エクサ	E
10 ¹⁵	ペタ	P
10 ¹²	テラ	T
10 ⁹	ギガ	G
10 ⁶	メガ	M
10 ³	キロ	k
10 ²	ヘクト	h
10 ¹	デカ	da
10 ⁻¹	デシ	d
10 ⁻²	センチ	c
10 ⁻³	ミリ	m
10 ⁻⁶	マイクロ	μ
10 ⁻⁹	ナノ	n
10 ⁻¹²	ピコ	p
10 ⁻¹⁵	フェムト	f
10 ⁻¹⁸	アト	a

(注)

1. 表1~5は「国際単位系」第5版、国際度量衡局1985年刊行による。ただし、1eVおよび1uの値はCODATAの1986年推奨値によった。

2. 表4には海里、ノット、アール、ヘクタールも含まれているが日常の単位なのでここでは省略した。

3. barは、JISでは流体の圧力を表わす場合に限り表2のカテゴリーに分類されている。

4. EC閣僚理事会指令ではbar, barnおよび「血圧の単位」mmHgを表2のカテゴリーに入れている。

換算表

力	N(=10 ⁵ dyn)	kgf	lbf
	1	0.101972	0.224809
9.80665		1	2.20462
4.44822		0.453592	1

$$\text{粘度 } 1 \text{ Pa}\cdot\text{s}(N\cdot\text{s}/\text{m}^2) = 10 \text{ P(ボアズ)}(\text{g}/(\text{cm}\cdot\text{s}))$$

$$\text{動粘度 } 1 \text{ m}^2/\text{s} = 10^4 \text{ St(ストークス)}(\text{cm}^2/\text{s})$$

圧力	MPa(=10bar)	kgf/cm ²	atm	mmHg(Torr)	lbf/in ² (psi)
力	1	10.1972	9.86923	7.50062 × 10 ³	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10 ⁻⁴	1.35951 × 10 ⁻³	1.31579 × 10 ⁻³	1	1.93368 × 10 ⁻²
	6.89476 × 10 ⁻³	7.03070 × 10 ⁻²	6.80460 × 10 ⁻²	51.7149	1

エネルギー・仕事・熱量	J(=10 ⁷ erg)	kgf·m		kW·h		cal(計量法)	Btu	ft·lbf	eV	1 cal = 4.18605J (計量法)	
		1	0.101972	2.77778 × 10 ⁻⁷	0.238889					= 4.184J (熱化学)	
	9.80665		1	2.72407 × 10 ⁻⁶	2.34270	9.29487 × 10 ⁻³	7.23301	6.12082 × 10 ¹⁹		= 4.1855J (15°C)	
	3.6 × 10 ⁶	3.67098 × 10 ⁵	1	8.59999 × 10 ³	3412.13	2.65522 × 10 ⁶	2.24694 × 10 ²⁵			= 4.1868J (国際蒸気表)	
	4.18605	0.426858	1.16279 × 10 ⁻⁶	1	3.96759 × 10 ⁻³	3.08747	2.61272 × 10 ¹⁹			仕事率 1 PS(仮馬力)	
	1055.06	107.586	2.93072 × 10 ⁻⁴	252.042	1	778.172	6.58515 × 10 ²¹			= 75 kgf·m/s	
	1.35582	0.138255	3.76616 × 10 ⁻⁷	0.323890	1.28506 × 10 ⁻³	1	8.46233 × 10 ¹⁸			= 735.499W	
	1.60218 × 10 ¹⁹	1.63377 × 10 ²⁰	4.45050 × 10 ⁻²⁶	3.82743 × 10 ⁻²⁰	1.51857 × 10 ⁻²²	1.18171 × 10 ⁻¹⁹	1				

放射能	Bq		Ci		吸収線量	Gy		昭射線量	C/kg	R		線量当量
	1	2.70270 × 10 ⁻¹⁰	1	100		1	0.01			1	100	
	3.7 × 10 ¹⁰	1							2.58 × 10 ⁻⁴	1		0.01

昭射線量	C/kg		R
	1	3876	
	2.58 × 10 ⁻⁴	1	

線量当量	Sv		rem
	1	100	
	0.01	1	

(86年12月26日現在)

プログラム並列化支援解析ツール kpx (その2)