JP9950162

JAERI-Data/Code
99-014

並列／非並列共用プログラム性能解析ツールの開発

1999年3月

渡部　弘・長尾佐市*・滝川好夫*・熊倉利昌*

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。
入手の問合わせは，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）で複写による実費頒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.

編集兼発行　　日本原子力研究所

JAERI-Data/Code 99-014

並列／非並列共用プログラム性能解析ツールの開発

日本原子力研究所計算科学技術推進センター
渡部　弘・長尾　佐市*・滝川　好夫*・熊倉　利昌*

（1999年２月９日受理）

日本原子力研究所計算科学技術推進センターでは科学技術計算プログラムの並列化を推進するため、並列／非並列共用プログラム性能解析ツールＫＭｔｏｏｌを開発し、評価を行っている。ＫＭｔｏｏｌはＦＯＲＴＲＡＮ77及びＭＰＩで記述されたプログラムの性能を解析するためのツールであり、並列プログラムを作成する際の負担軽減を目的に開発されている。本稿ではＫＭｔｏｏｌ開発目的、設計、利用方法及び試用結果について説明する。

日本原子力研究所：〒153－0061　東京都目黒区中目黒２－２－５４
＊ウッドランド株式会社

JAERI-Data/Code 99-014

Development of Parallel/Serial Program Analyzing Tool

Hiroshi WATANABE, Saichi NAGAO,*
Yoshio TAKIGAWA* and Toshimasa KUMAKURA*

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

(Received February 9, 1999)

Japan Atomic Energy Research Institute has been developing "KMtool", a parallel/serial program analyzing tool, in order to promote the parallelization of the science and engineering computation program. KMtool analyzes the performance of program written by FORTRAN77 and MPI, and it reduces the effort for parallelization. This paper describes development purpose, design, utilization and evaluation of KMtool.

Keywords: FORTRAN77, MPI, Analyzing Tool, Parallel Program, Serial Program

* WOODLAND Corporation

目　　次

Contents

1 はじめに

日本原子力研究所計算科学技術推進センターでは科学技術計算プログラムの並列化を推進するため、平成7年度と8年度にはプログラム並列化支援解析ツール kpx[1] [2]を、平成9年度には並列効率予測ツール pep [3]を開発してきた(図1参照)。さらに平成10年度には、これらの機能を統合した新しいツール KMtool を開発し、評価を行っている。KMtool は FORTRAN77 及び MPI で記述されたプログラムの性能を解析するためのツールであり、並列プログラムを作成する際の負担軽減を目的に開発されている。本稿では KMtool の開発目的、設計、利用方法及び試用結果について説明する。

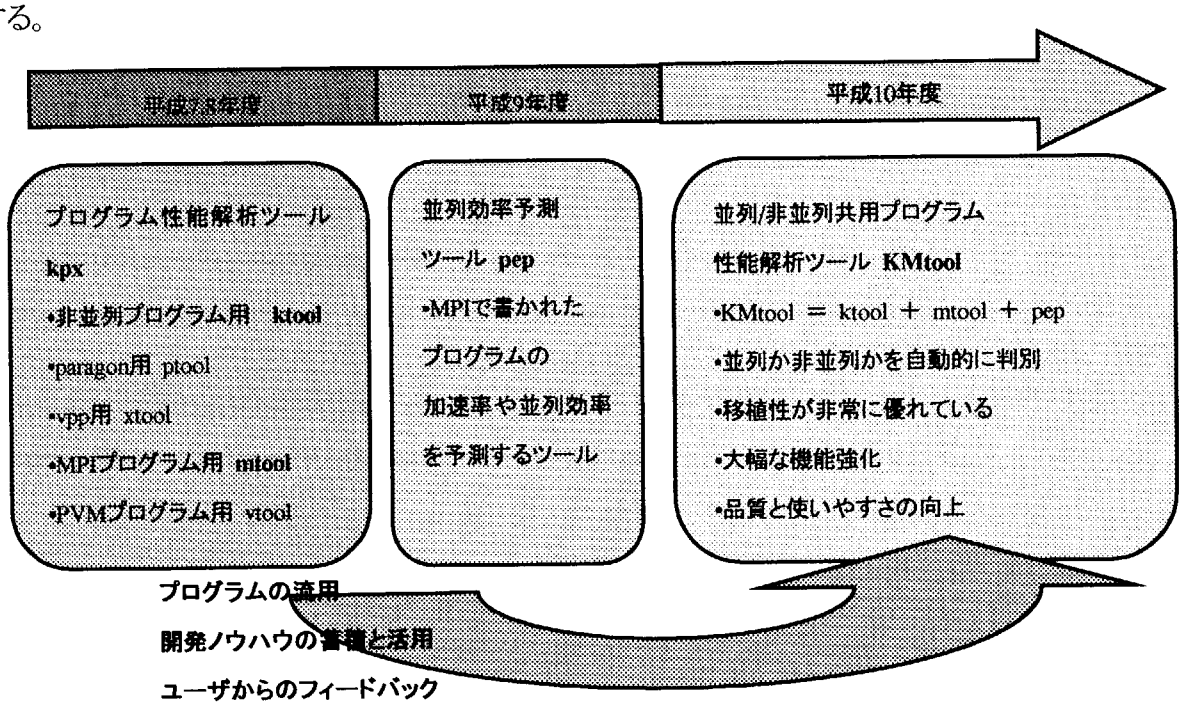


図 1 性能解析ツールの開発経緯

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るが、高速性やメモリ容量、さらには経済性の観点から、分散メモリ型の方が有利であることは議論の余地がない。しかしプログラミングは分散メモリ型の方がはるかに困難であり、そのことがプログラム開発者の悩みの種となっている。

分散メモリ型の並列計算機のプログラミングには、MPI(Message Passing Interface)を用いることが多い。MPI とは有力なコンピュータメーカを含む 40 以上の団体が参加した MPI フォーラムによって国際的に標準化された、計算機種に依存しない並列化通信ライブラリである。MPI は通信性能が良好であり、しかも非常に移植性に優れていたので、短期間の内に多くの並列計算機に実装された。そのため MPI で記述されたプログラムは、ほとんど全ての分散メモリ型並列計算機上で、何の変更もなく動作するようになり、並列計算の普及に大きく貢献している。

ところが MPI によるプログラム開発には重大な問題点がある。それはプログラムのライン数が膨大になり、多大な開発工数が必要となることである。その原因は MPI によるプログラミングでは、並列化通信インタフェースをプログラム中に記述することに加えて、各並列化プロセスの動作手順も1つの

プログラム中に全て記述しなければならないからである。このためプログラムが非常に複雑になり、デバッグや最適化作業に重大な支障をきたしている。この問題点を改善するための研究も研究各機関で行われているが[5]、残念ながら実用化されている研究成果は多くない。

このため MPI によるプログラミングでは、プログラマの労力を軽減するためのツールが必要不可欠である。例えばプログラムのどの部分を並列化すれば最も効果的か、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、プログラムを並列化し最適化する際の有効な指針となりうる。そのために利用されるツールがプログラム性能解析ツールである。

プログラム性能解析ツールはプログラムの各部分の計算時間、通信時間及び実行回数を計測する。これによりプログラマはどの部分を集中的に調査し並列化すればよいか、性能向上のネックになっている部分はどこかなどを知ることができる。

もちろんこのようなツールは並列計算機メーカ自体が開発し、提供している場合が多い。しかしメーカ提供のツールは、その並列計算機上でしか利用できない場合がほとんどである。すなわち可搬性がない。従ってプログラマが他の並列計算機を利用して性能評価を行う場合、ツールの利用方法を新たに修得しなければならない。これは開発者に大きな負担を強いるものであり、プログラムの並列化を妨げる原因ともなっている。

一方、機種によらないプログラム性能解析ツールとしては、UNIX が提供する prof コマンドやサードパーティが提供するツールがある。このうち prof は UNIX マシンならばどの機種でも利用でき非常に使いやすいが、ルーチン単位の正味の実行時間しか計測できず、機能的に不十分である。サードパーティが提供するツールはサポートされていない機種もあり、またツールがどのようにしてプログラムを計測しているかが分かりにくい場合が多い。プログラム性能解析ツール利用時には、ツール自体の副作用によりプログラムの性能が大きく変わることが起こりうる。例えばツールが計測用のシステムルーチンをプログラムに挿入したため、ベクトル化やインライン展開が阻害され、プログラムの速度が大幅に低下する場合等がそれにあたる。このような状況を回避するためにも、ツール動作の分かりやすさは重要な要素である。

以上のことから、今回我々が開発したプログラム性能解析ツールである KMtool では次のことに重点を置いた。

(1) いかなる並列計算機にも容易に移植できること
(2) プログラムの計測範囲の指定が自由かつ容易にできること
(3) ツールの計測方法がユーザに分かり易いこと
(4) ツールによるオーバヘッド時間を最小限にすること

本稿では KMtool がどのようにして上記の要求を実現しているかを中心に述べる。第2章で KMtool の機能及び構成を概観した後、第3章と第4章で KMtool の設計の詳細について記述する。第5章では KMtool の利用方法を、第6章では実際に科学技術計算プログラムに適用した結果について報告する。

2 KMtool の機能と構成

KMtool の計測対象は、拡張 FORTRAN77 及び MPI により記述されたプログラムであり、プログラムが並列か非並列かは KMtool が自動的に識別する。従って逐次型プログラムにおけるコスト分析作業と、並列化プログラムの最適化に必要となる時間計測及びロードバランス調査が、同一のツールでシームレスに行える(図2参照)。

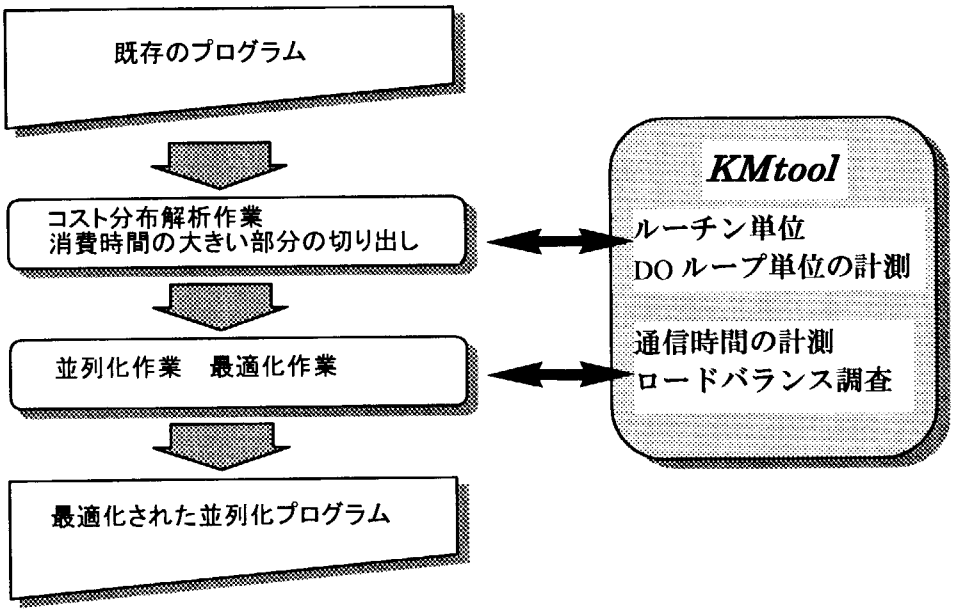


図 2 並列化作業の流れと KMtool

KMtool は全ての UNIX 系並列計算機上で動作するように設計されており、日本原子力研究所計算科学技術推進センター設置の5機種の並列計算機、SP2、SR2201、VPP300、SX-4、T94 において動作が確認されている。

KMtool は以下の機能を有する。

(1) メインルーチン、サブルーチン、外部関数の全経過時間(elapsed time)と正味の経過時間(net time)及び実行回数(counts)を計測する。以下ではメインルーチン、サブルーチン、外部関数をあわせてルーチンと総称する

(2) ルーチン中のdoループ、call文、入出力文、MPIルーチンコールの経過時間(elapsed time)及び実行回数(counts)を計測する。

(3) (2)に加えソースファイルにKMtool指示行を挿入することにより、任意の部分の経過時間及び実行回数を計測する。

(4) 並列プログラムの場合には、並列実行結果から加速率や並列効率を予測する。

(5) 専用のポストプロセッサKMviewにより、計測結果をWEBブラウザ上でグラフ表示することが可能である。

ここでルーチンの全経過時間(elapsed time)と正味の経過時間(net time)の違いについて説明しておこう。ルーチンの全経過時間には、そのルーチンが call しているサブルーチンの経過時間が含ま

れる。一方、正味の経過時間には call しているサブルーチンの経過時間は含まれない。例えば、サブルーチン sub1 がサブルーチン sub2 を call しており、図3のような経過をたどるものとすると、sub1 の全経過時間は T3 - T0、正味の経過時間は (T3-T2) + (T1-T0)で計算される。UNIX で提供される prof コマンドもそうであるが、既存のプログラム性能解析ツールには正味の経過時間のみを計測し、全経過時間は計測しないものも多い。しかしプログラム並列化作業では上位レベルでの並列化、すなわちサブルーチンを call する側での並列化が性能向上のために重要なので、全経過時間を計測する機能は必須である。

```
Time      T0              T1                  T2              T3
          +==============>+-------------------+==============>+------->
Event     start sub1      call sub2           ^               return
                          |                   |
                          v                   |
                          +================>+
                          start sub2          return
```

図 3 全経過時間と正味の経過時間

次に KMtool の構成について述べる。KMtool は以下のモジュールから成り立っている。

(1) kmtool :KMtool 本体
(2) kmrun :KMtool 実行用シェルスクリプト
(3) KM_@system.f :KMtool システムルーチンファイル
(4) KM_control.dat :KMtool 制御データファイル
(5) KMview :KMtool の計測結果をグラフ表示するツール

kmtool は本ツールの本体であり、制御データファイル KM_control.dat に従って計測用ソースファイルの生成と、計測用の KMtool システムルーチンファイル KM_@system.f の生成を行う。KMtool 実行用シェルスクリプト kmrun は kmtool 実行のためにファイル環境を整え、kmtool を起動する。従ってユーザが直接 kmtool を起動することはない。

kmrun を実行にすると、計測用ソースファイルと計測用の KMtool システムルーチンファイル KM_@system.f が、新たに生成される。計測用ソースファイルにはオリジナルソースファイルの計測部分に、経過時間と実行回数を計測するためのシステムルーチンコールが挿入されている。

ユーザは計測用ソースファイルと計測用 KMtool システムルーチンファイル KM_@system.f をコンパイル・リンクすることにより、計測用のロードモジュール a.out を作成する。さらにこの計測用ロードモジュールを通常通り実行すると、計測結果ファイル KM_output.xxx が出力される。xxx は 000 から 999 までの番号であり、自動的に付加される。ユーザはこの計測結果ファイルの内容をエディタ等で確認することができる。また WEB ブラウザ上から KMview を起動することにより、計測結果をグラフ表示することも可能である。

図4に KMtool の計測方法の流れを示す。この図では例として、計測対象ソースファイル名を main.f としている。

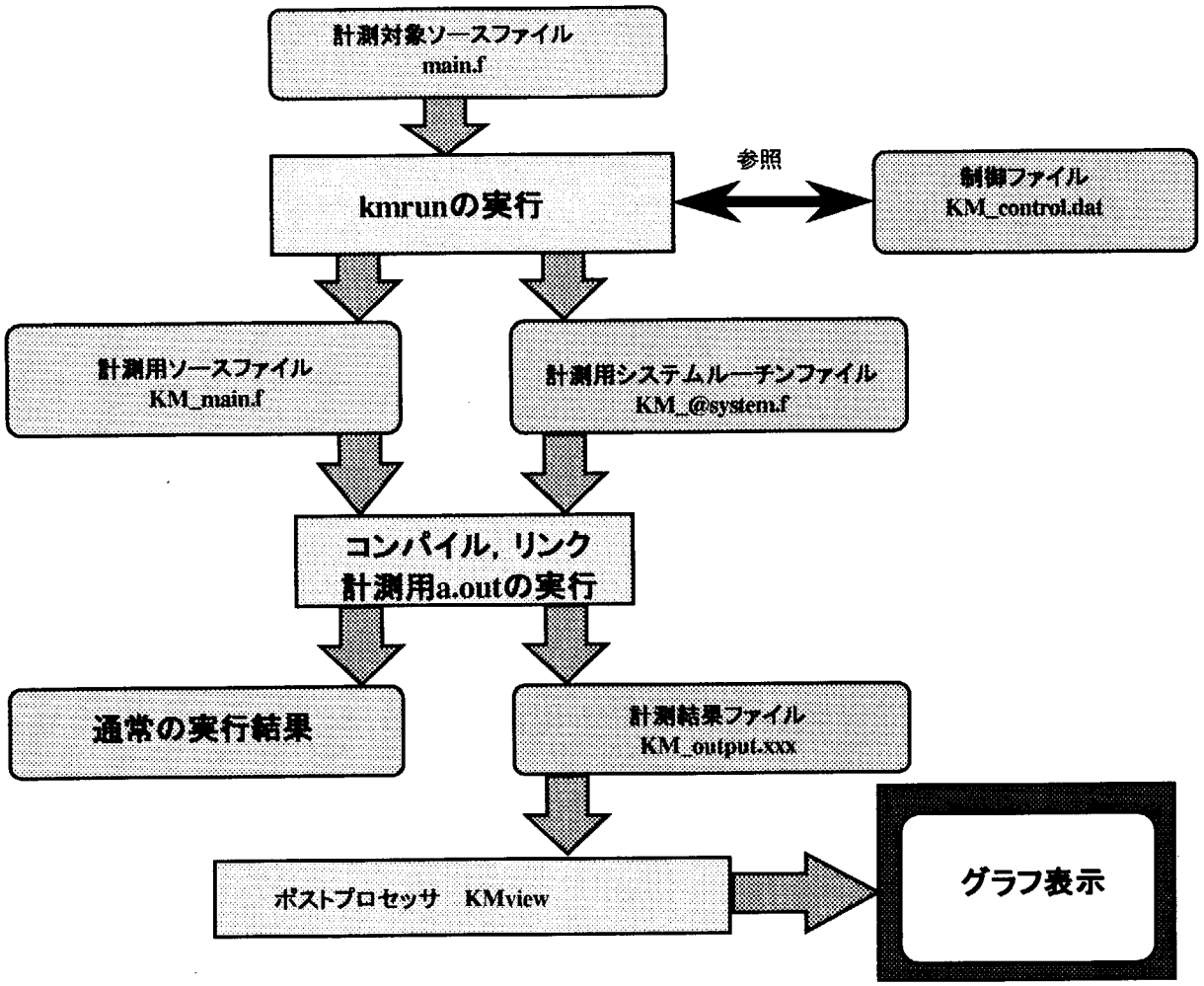


図 4 KMtool による計測の流れ

3 KMtoolの設計

3.1 設計目標

KMtoolは以下の機能及び特徴を有するように設計されている。

(1) 並列／非並列プログラム各部分の経過時間及び実行回数を計測する。計測対象言語はFORTRAN77とMPIとするが、広く普及しているFORTRAN77の拡張文法は積極的にサポートする。

(2) 並列／非並列はKMtoolが自動的に識別し、並列プログラムの場合には通信時間及び通信回数も計測する。

(3) 並列化プログラムの場合には、並列実行結果から加速率やPEの並列効率を予測する。

(4) いかなる並列計算機にも容易に移植できるようにする。

(5) プログラムの計測範囲の指定が自由かつ容易にできるようにする。

(6) ツールがどのような方法で計測しているかをユーザが見られるようにする。

(7) 計測のオーバヘッド時間を最小限にする。

第3章ではKMtoolがどのような設計により、これらの機能及び特徴を実現しているかを記述する。

3.2 移植性の確保

KMtoolの開発にあたり、移植性の確保は最重要課題である。なぜならコンピュータメーカ製性能解析ツールとKMtoolを差別化する唯一の点が、この移植性にあるからである。コンピュータメーカは自社製のハードウェアやOS等を熟知しており、その知識に基づいて性能解析ツールを開発できる。当然精度面や使い易さに関しては、KMtoolはコンピュータメーカ製性能解析ツールに太刀打ちできない。そのような中、KMtoolの重要な存在意義としてはどんな計算機上でも利用可能とする優れた移植性があげられる。いかに高精度で使いやすい性能解析ツールでも、異なる計算機を利用するたびに新しく利用方法を修得するのでは、ユーザの負担が大きい。一方、KMtoolは一旦利用方法を修得すれば、あらゆる計算機上で全く同じ方法で利用できる。

移植性を確保するためKMtoolの開発にはFORTRAN77言語を用いた。この種のツールの開発には開発効率の面から、C言語もしくはC＋＋言語を用いるのが一般的である。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。

- KMtool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版があるように機種によって多少の違いがある。

以上のような理由から開発効率は落ちるものの、開発言語としてFORTRAN77を採用した。しかしそれでも機種による時刻取得ルーチンの違いだけは如何ともしがたい。そこで我々はKM_gettという名称の抽象的なKMtool用時刻取得ルーチンを定義し、このKM_gett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。KM_gett は KMtool システムルーチンファイル KM_@system.f で定義されており、新しい計算機に KMtool をインストールする場合には、この部分だけを修正すればよい。

3.3 経過時間及び実行回数の計測手法

KMtool は経過時間及び実行回数を計測するために、以下のような手順で計測用ソースファイル及び計測用システムルーチンファイルを生成する。(図5参照)

(1) 計測対象ルーチンに経過時間や実行回数を格納するための計測データ配列の宣言文を挿入する。

(2) 計測対象ルーチンの実行文の直前に、ルーチン計測開始用のシステムルーチン KM_init の call 文を挿入する。このシステムルーチンは計測対象ルーチンがスタートした時刻の取得と実行回数のカウントアップを行う。

(3) 計測対象ルーチンの RETURN 文の直前に、ルーチン計測終了用のシステムルーチン KM_term の call 文を挿入する。このシステムルーチンは計測対象ルーチンからリターンする時刻を取得し、ルーチンの経過時間を計算して(1)の経過時間用計測データ配列に加算する。

(4) do ループや call 文等の計測対象部分直前に、計測開始用のシステムルーチン KM_msri の call 文を挿入する。このシステムルーチンは計測対象部分がスタートした時刻の取得と実行回数のカウントアップを行う。

(5) do ループや call 文等の計測対象部分直後に、計測終了用のシステムルーチン KM_msro の call 文を挿入する。このシステムルーチンは計測対象部分の実行終了時刻を取得し、その部分の経過時間を計算して、(1)の経過時間用計測データ配列に加算する。

(6) 計測対象ルーチン中に stop 文等がある場合には、その直前に計測結果を集計して出力するためのシステムルーチン KM_writ の call 文を挿入する。

(7) KM_writ は(1)で各計測対象ルーチンに挿入された計測データ配列に格納された計測結果を、common 文で参照する。この common 文は計測対象ルーチンごとに KM_writ に追加される。

このように計測各部分の全経過時間は、その部分の前後で時刻を取得し、その差を計測データ配列に加算することで計算している。

一方、ルーチンの正味の経過時間は全経過時間から次のようにして計算する。図3で示した sub1 が sub2 を call する場合についての、sub1 の正味経過時間を例に説明しよう。sub2 を呼び出すための call 文が計測対象になっている場合は簡単である。計測結果を出力する段階で、sub1 の全経過時間から sub2 の call 部分の経過時間を差し引いた値を正味経過時間として出力すれば良い。一見、これで問題は解決されたように思うかもしれないが、そうではない。なぜなら、これだけでは sub1 中にある外部関数の経過時間が考慮されないからである。サブルーチンの call 文に相当するシンボルが、外部関数には存在しないことが問題を難しくしている。さらに後述する理由により、KMtool は既定値では do ループ内の call 文を計測しない。従って sub2 の call 文が do ループ内にある場合には計測対象とならず、正味の経過時間を計算できない。

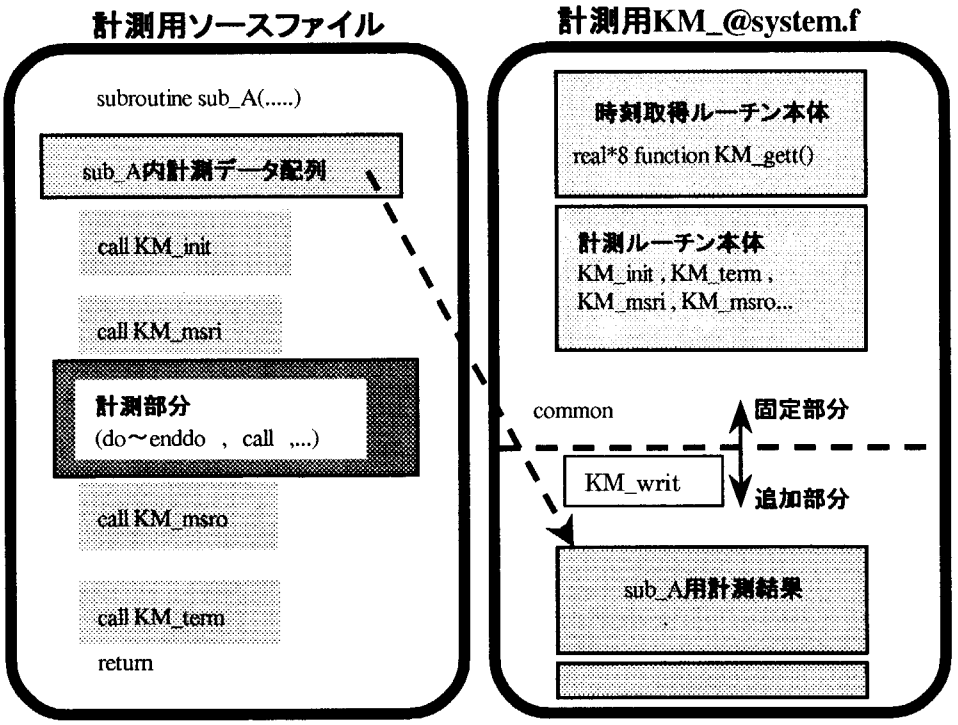


図 5 計測用ソースファイルと計測用システムルーチンファイル

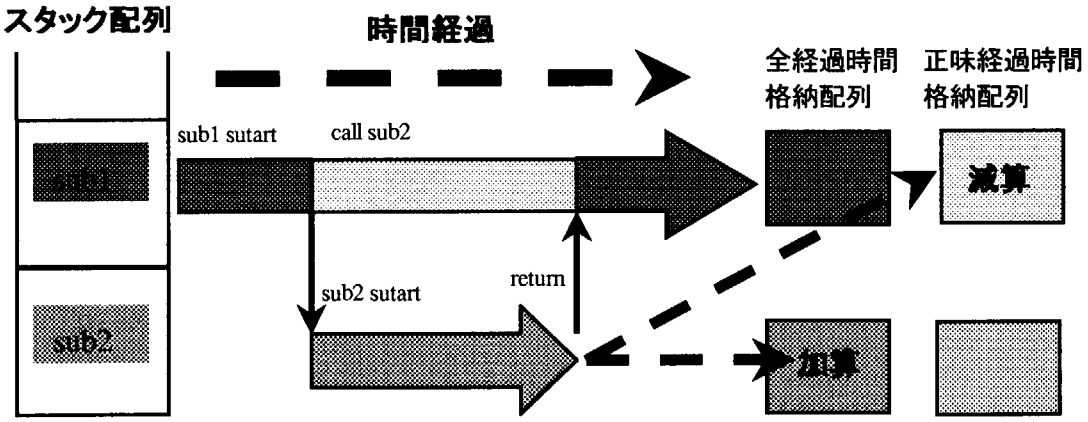


図 6正味経過時間の計算手法

この問題を我々は、ルーチンの親子関係をスタック配列で動的に管理することで解決している(図6参照)。すなわち sub1 がスタートした段階でスタック配列に sub1 実行中という情報を格納する。さらに sub2 が call され、スタートした段階でスタック配列に sub2 実行中という情報を追加する。sub2 からの return 時には sub2 の経過時間を sub2 の全経過時間用配列に加算すると同時に、スタック配列により認識される親ルーチン sub1 の正味経過時間用配列から差し引く。その後スタック配列から sub2 実行中の情報を削除する。ここで、ルーチン sub1 の中で call sub2 が計測対象になっている場

2 実行中の情報を削除する。ここで、ルーチン sub1 の中で call sub2 が計測対象になっている場合には、重複して sub2 の経過時間を差し引くことがないように注意する必要がある。それには計測対象の call 文の間だけフラグを立てればよい。こうすることにより sub1 の正確な正味経過時間を計算することが可能となる。

3.4 計測方法の分かり易さ

KMtool は計測用のソースファイルを生成し、それから計測用のロードモジュールを作成して計測を行う。従って生成された計測用ソースファイルを見れば、どのようにして計測が行われているかが容易に理解できる。このことは計測用ロードモジュールの実行時間が、オリジナルの実行時間に比べ著しく伸びた場合の原因究明及び回避に役立つ。

3.5 並列／非並列の自動認識

MPI により並列化されているプログラムの認識と計測は容易である。なぜなら MPI の通信インタフェースは全て、call MPI_xxxx という形式を持っているからである。従って、この形式の call 文の経過時間及び実行回数を計測すれば、各 PE の通信時間及び通信回数を得ることができる。

3.6 加速率・並列効率予測機能

並列化プログラムにおいて、良好な性能が得られているか否かの判断は容易でない。プログラム経過時間がプロセッサ数分の1になれば、理想的に並列化されているといえるが、実際にそのようなことはない。なぜなら並列化プログラムには並列化による通信部分と、並列化されず全てのプロセッサが全く同じ作業を行う冗長実行部分が存在するからである。しかも並列化されたプログラムではデータサイズが非並列の場合に比べ桁違いに大きい場合が多く、1PE で実行することはメモリ的にも計算時間的にも不可能なことが少なくない。この場合には判断の基準となる 1PE での経過時間さえ、計測することは困難である。

これに対処するため、KMtool では並列化プログラムの実行から 1PE での実行時間を予測し、さらに加速率や並列効率を予測する機能をサポートしている。ここで加速率・並列効率はそれぞれ

加速率＝1PE での実行時間／並列実行時間

並列効率＝加速率／PE 数

という式で定義される。ユーザはこれらの指標値を、プログラム最適化作業の目安とすることができる。

本機能実現の鍵は 1PE での実行時間をどのようにして予測するかである。KMtool では以下のようにして、1PE での実行時間を予測している。

・並列プログラムの並列実行時間、並列通信時間、冗長実行時間を計測する。

・上記計測時間から 1PE で実行した場合の経過時間 T_{1pe} を次式で予測する。

$$T_{1pe} = \text{冗長実行時間の最小値} + \Sigma_{all\,pe}\,\text{並列実行時間}$$

次の図は 1PE での実行時間予測方法の概念を示したものである。

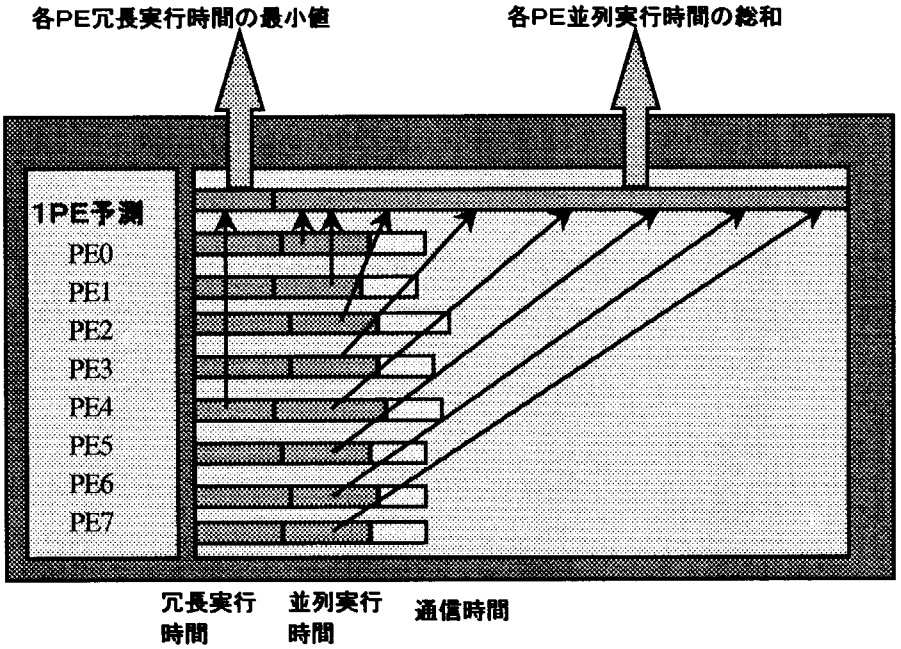


図 7 1PE 実行時間予測方法

3.7 計測オーバヘッドの最小化

KMtool の開発で最も力を注いだ点が、計測によるオーバヘッドを如何に小さくするかである。平成7、8年度に開発した kpx を実際の科学技術計算プログラムに適用した例では、計測用プログラムの実行時間がオリジナルのプログラム実行時間の2倍以上かかる場合がほとんどであり、ユーザの不評を買ったからである。kpx の開発時にも、計測処理は全てメモリ上で行いファイル入出力は一切行わない等の高速化の工夫をしたが、十分ではなかった。そこで KMtool では kpx 開発時に採用した工夫の他に、以下のような手法を用いて計測オーバヘッドの最小化を試みた。

3.7.1経過時間予測機能の導入

kpx による計測のオーバヘッドの原因を調査したところ、各マシンの時刻取得ルーチンを call する部分でオーバヘッド時間の大半を占めていることが判明した。そこで KMtool では、計測対象部分の実行回数がある回数(既定値では10万回)に達した時点で、1回当たりの平均経過時間を求め、以降の実行では実際の時間計測を行わずに平均経過時間を用いるようにした。これにより、時刻取得ルーチンの call 数が著しく減少する。

3.7.2段階的計測手法の利用

kpx では一回の計測で必要な全てのデータを採取することを目的に設計された。しかしこれでは、計測対象プログラムのあらゆる部分に計測用システムルーチンコールを挿入しなければならず、過大なオーバヘッドの原因となる。そこで KMtool では次の段階的計測手法を想定し、開発を行った。

(1) 最初は計測対象を粗くして計測する。

(2) (1)の計測結果を見て、より詳細な情報が必要な部分には、計測対象を細かくして再度計測を行う。

上記計測モデルに従った計測を行い易くするため、KMtool は次のように設計されている。

- 既定値の計測対象は、ルーチン、最外側 do ループ、do ループに含まれない call 文、及び MPI ルーチンコールである。これは(1)の最初の計測対象は粗くするという方針に対応している。
- KMtool はファイル単位で適用可能とする。これにより、最初粗く計測した後特定のルーチンのみを詳細に計測したいという場合、そのルーチンを含むソースファイルにのみ再度 KMtool を適用すればよい。他の部分は前回コンパイルしたオブジェクトファイルが流用できるので、コンパイル時間を短縮できる。
- ルーチンごとの計測対象レベルを容易に変更できるようにする。これには KMtool 制御ファイル KM_control.dat が用いられる。KM_control.dat については第5章で説明する。

4 KMviewの設計

4.1 設計目標

KMview は KMtool の計測結果を簡単にグラフ表示するためのポストプロセッサである。KMtool では計測結果をほとんど編集せず出力するので、計測結果ファイルからプログラムの性能特性を把握するのは必ずしも容易でない。特に並列化プログラムの場合には、PE 台数分のデータが計測結果ファイルとして出力されるので、プログラム性能解析はさらに困難となる。従って計測結果の解析にはポストプロセッサの利用が必須といえる。

KMview の開発にあたり、設計目標を以下のようにした。

- リモートの並列計算機上にある計測結果を、ローカルの端末でグラフ表示できること。
- 端末には PC や X 端末を採用し、なるべく多くの機種をサポートすること。
- インストールが容易であること。
- 初めての人でもマニュアルなしで利用できること。

4.2 KMviewの開発

上記の設計目標を実現するため、我々は KMview を JAVA 言語で開発することにした。JAVA 言語とは米国の SUN 社によって開発された、可汎性に優れたオブジェクト指向型言語である。JAVA 言語で開発することにより、以下の効果が期待できる。

(1) JAVA 言語は WWW 上で利用されることを前提に開発されており、リモートのファイルをローカルな環境で見るということが容易に実現できる。

(2) KMview を JAVA アプレットとして作成すれば WEB ブラウザ上で利用できる。従って WEB ブラウザが動作するマシンならどのようなマシンでも、KMview が利用可能となる。また使い慣れた WEB ブラウザ上で利用するので、操作法が容易に修得できる。

(3) HTML と共にクラスファイルを提供すれば、インストール作業が必要ない。もちろんリコンパイルの必要もない。

図 8 に日本原子力研究所計算科学技術推進センター中目黒地区における、KMview の利用形態と概観を示す。KMview の詳細については付録 2 を参照されたい。

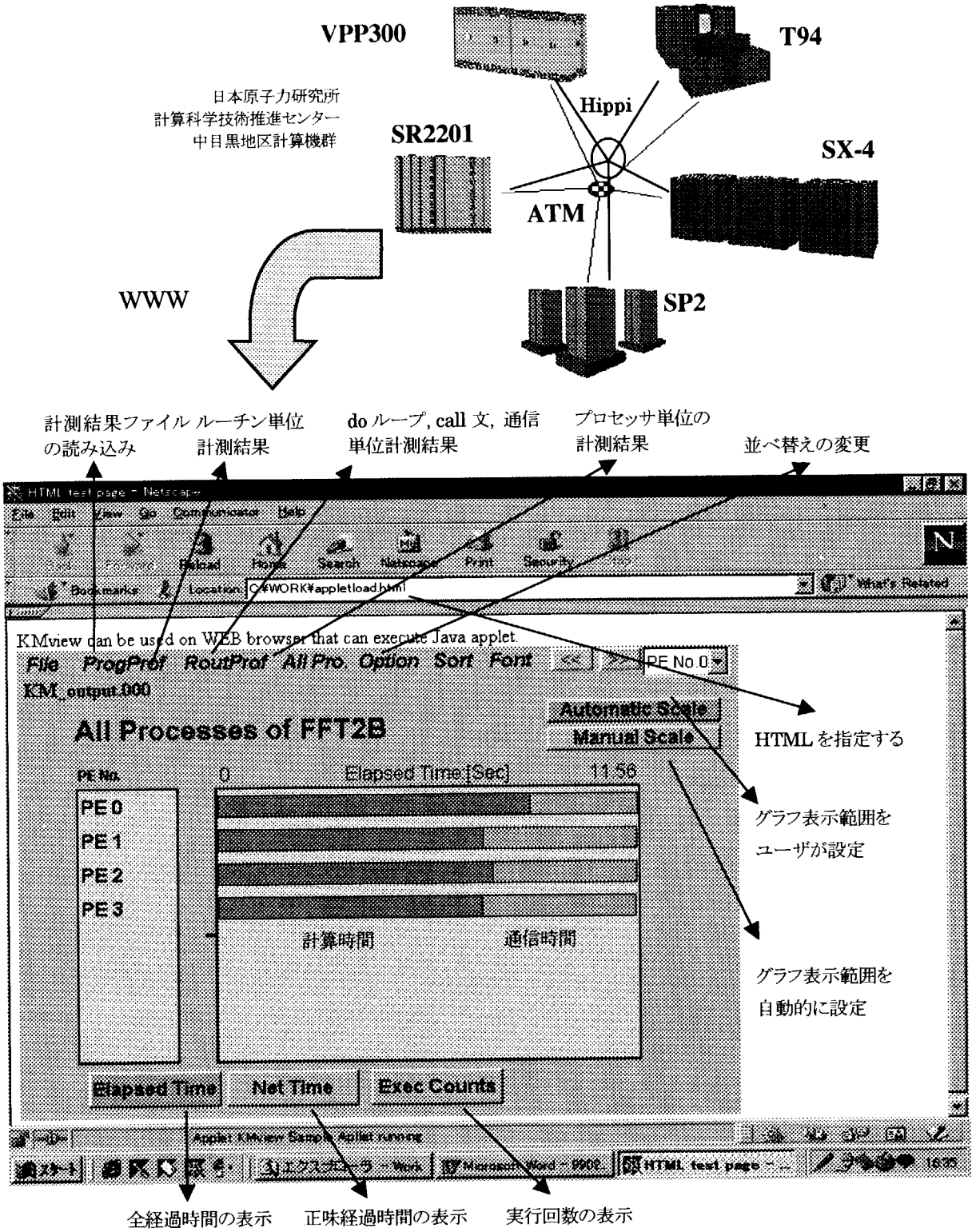


図 8 KMview の利用形態と概観

5 KMtoolの利用方法

5.1 KMtoolのインストール方法

KMtoolをインストールするためには、当センターのファイルサーバ等からtar形式のファイルを入手した後、インストール用シェルスクリプトkminstを起動する。以下にその方法を説明する。

(1) 当センターのファイルサーバ(ccsemfa, 202.241.61.101)よりtarファイル(/nhome001/g0188/j0000/KPX/KMtool.tar)を入手し、インストール対象計算機のホームディレクトリに置く。

```
host> cd
host> ftp 202.241.61.101
ftp> bin
ftp> cd ~j0000/KPX
                OR
     cd /nhome001/g0188/j0000/KPX
ftp> get KMtool.tar
ftp> bye
```

(2) インストール対象計算機のホームディレクトリにKMtool用ディレクトリを作成し、KMtool.tarをそのディレクトリ下で展開する。再インストールの場合はディレクトリKMtoolを削除した後、下記を実行する。

```
host> mkdir KMtool
host> cd KMtool
host> tar xvfo ~/KMtool.tar
```

(3) インストーラに計算機名の引数を与えて起動し、KMtoolのロードモジュールを作成する。現状で受付可能な計算機名の引数は、SP2、SR2201、VPP300、SX-4、T94のどれか一つである。インストール対象の計算機がこれ以外の場合でも、FORTRANコンパイラ名と時刻取得ルーチン名が同じ計算機があれば、その計算機名を指定できる。例えばインストール対象計算機がsx5の場合には、HOSTにはsx4と指定すればよい。

```
host> kminst HOST    (または csh kminst HOST)
       (HOSTにはsp2, sr2201, vpp300, sx4, t94, otherのどれか一つを指定)
sp2     : SP2(ＩＢＭ)用のKMtoolロードモジュールを作成
sr2201  : SR2201(日立)用のKMtoolロードモジュールを作成
vpp300  : VPP300(富士通)用のKMtoolロードモジュールを作成
sx4     : SX-4(ＮＥＣ)用のKMtoolロードモジュールを作成
t94     : T94(ＣＲＡＹ)用のKMtoolロードモジュールを作成
other   : 上記以外のマシン用のKMtoolロードモジュールを作成
```

(4) (3)のHOST引数にSP2、SR2201、VPP300、SX-4、T94が指定できない場合は、HOSTにotherを用いる。この場合、インストーラがFORTRAN77コンパイラ名の入力を要求するので、適切なコンパイラ名を指定する。

```
host> kminst other
========== Install KMtool for the OTHER ==========
           :
Please input FORTRAN77 compiler name. (RETURN key : f77)
fort77              ... コンパイラ名がfort77の場合
```

(5) さらにkminstの実行が終了した後、KMtoolシステムルーチンファイルKM_@system.f中にある時刻取得関数KM_gettを編集して、clockという時刻取得ルーチン名を適切なルーチン名に変更する。変更箇所は1箇所だけである。以下の作業は、インストール対象計算機がSP2、SR 2201、VPP300、SX-4、T94の場合には必要ない。

```
host> vi KM_@system.f

      function KM_gett                                          0
      implicit none
!KM_gett
      real*8   KM_gett
c
c     KM_gett is a double precision function of KMtool to get time
c     at the moment. This function must return time in seconds,
c     so user may need to modify this function.
c
!(CS)KM_cget
!KM_cget
      common /KM_cget                                           /
     -                KM_gcnt
      integer         KM_gcnt
!(CE)KM_cget
c
      real*8   elp_time
c     real*8   cpu_time
c
c     ------------------------                                  !STAND
      call clock  (elp_time)                                    !STAND
          ^^^^^ -> このclockを変更する
```

(6) KMtoolディレクトリにパスを設定する。具体的には.cshrcの最後に次の行を付け加えればよい。

```
set path = ( $path ~/KMtool )
```

(7) インストールが正常に終了すると、ディレクトリKMtool下に以下のファイルが生成される。

KMtool/	:KMtool directory
KM_@system.f	:KMtoolシステムルーチンファイル
KM_control.dat	:KMtool制御データファイル
README.euc	:KMtool簡易説明書 本ドキュメント eucコード
README.sjis	:KMtool簡易説明書 本ドキュメント シフトJISコード
README.txt	:KMtool簡易説明書 英語版
kminst	:インストーラ(インストール後は消去可)
kmrun	:KMtool実行用シェルスクリプト
kmtool	:KMtool本体

5.2 kmrun の利用方法

第2章で説明したように、KMtoolにおける計測にはkmrunを利用する。ここではkmrunの利用方法を説明する。

(1) FORTRANソースファイルのあるディレクトリに移動する。

(2) kmrunを起動し、計測用ソースファイルとKM_@system.fを生成する。

```
host> kmrun *.f
          ... 全てのFORTRANソースファイルを計測対象とする
              KM_*.fとKM_@system.fが生成される
```

あるいは次のように、一部のソースファイルにのみ適用することもできる。

```
host> kmrun main.f aaa.f
          ... main.fとaaa.fを計測対象とする
              KM_main.f KM_aaa.f KM_@system.fが生成される
```

一部のソースファイルに適用する場合、プログラムが終了する可能性のあるルーチン(メインルーチンや STOP 文のあるルーチン)を含むソースファイルは、必ず **kmrun** の引数で指定しなければならない。そうしないと、計測結果ファイル **KM_output.xxx** が出力されない可能性がある。

(3) 計測用ソースファイルの出力ディレクトリを変更したい場合は、kmrunに-tdオプションをつけて起動する。-tdは最初の引数でなければならない。-tdで存在しないディレクトリが指定された場合、新たに作成される。

```
host> kmrun -td test *.f
          ... test/の下に計測用ファイルが生成される
```

(4) 計測部分を変更したい場合、出力ディレクトリ上にKMtool制御データファイルKM_control.datを用意して編集するか、直接ユーザソースファイルにKMtool制御指示行を埋め込む。出力ディレクトリとはkmrunの-tdオプションで指定されたディレクトリであり、-tdオプションがなければカレントディレクトリである。計測部分変更の詳細は、次節を参照されたい。

(5) 同じディレクトリで再度kmrunを実行した場合、下記に示すメッセージが出力される場合がある。これは前回のKMtool制御データファイルKM_control.datが残っているためであり、計測部分を変更する等の理由でユーザがKM_control.datを編集した場合にはyを指定する。nを指定した場合は~/KMtoolにある既定値のKM_control.datにより上書きコピーされ、KMtoolの動作制御に利用される。

```
host> kmrun *.f
The control file ./KM_control.dat already exists.
Use this control file? (y/n)
y      ... KMtoolの制御のために./KM_control.datを使用する
n      ... ~/KMtool/KM_control.datをコピーして使用する
```

(6) kmrunが終了した時点で、もしKM_message.txtというテキストファイルが出力されていれば、その内容を確認する。そのファイルにはKMtoolのエラーメッセージや警告メッセージが記述されている。

(7) 生成された計測用ソースファイル及びシステムルーチンファイルKM_@system.fをコンパイル・リンクし、計測用ロードモジュールを作成する。その後、通常の方法で計測用ロードモジュールを実行する。

```
host> f77 KM_*.f        ... KM_*.fの中にはKM_@system.fも含まれる
host> a.out             ... 計測用ロードモジュールを実行
```

(8) 計測用ロードモジュールが終了すると同時に、計測結果ファイルKM_output.xxxが出力される。xxxは000から999まで自動的に番号が付けられる。

(9) KMviewを利用することにより、計測結果をWEBブラウザからグラフ表示することができる。

5.3 KMtool 制御ファイルの仕様

KMtool の計測範囲を制御するには、KMtool 制御ファイル KM_control.dat を利用する。その他に KMtool への指示行を計測対象ソースファイルに挿入することも可能だが、あくまでも補助的な手段である。ここでは KM_control.dat の仕様を説明する(図 9 参照)。

```
! "KM_control.dat" is the KMtool's control data file. Parts after
!  a exclamation mark(!) in a line are regarded as comments.

OUTPUT_FILE_NUMBER = 99
TRACE_FILE_NUMBER  = 0

TIME_PREDICTION_MODE_NUMBER = 100000
! KMPI_FOR_STA = 1

! TABNUM  = 04
! TABCODE = 09

! PRINT_OUT_ROUTINES = routine1,routine2
IGNORED_ROUTINES = mpi_wtime,mpi_wtick

ROUTINE_MEASURED_LEVEL = 2
DOLOOP_MEASURED_LEVEL = 1
CALL_MEASURED_LEVEL = 1
MPI_MEASURED_LEVEL = 2
IO_MEASURED_LEVEL = 0

! IF_ROUTINE = subroutine1,function2
!    CALL_MEASURED_LEVEL = 2
! END_IF

END_OF_DATA
Lines below "END_OF_DATA" are comments and are ignored by KMtool.
```

図 9 KMtool 計測制御データファイル

(1) OUTPUT_FILE_NUMBER

計測結果出力ファイル用装置番号。

(2) TRACE_FILE_NUMBER

トレースファイル用装置番号。1以上の値が指定された時はそのファイル番号でトレースファイルが作成される。

(3) TIME_PREDICTION_MODE_NUMBER

経過時間予測機能を用いるしきい値。

(4) KMPI_FOR_STA

1が指定された時はSTA用の計測用ソースファイルを生成。

(5) PRINT_OUT_ROUTINES

このキーワードで指定されたルーチンの return 時にも計測結果を出力する。メインルーチンがC言語で書かれている場合等に利用する。

(6) IGNORED_ROUTINES

このキーワードで指定されたルーチンは完全に計測対象外になる。すなわち call する側でもされる側でも計測されない。

(7) ROUTINE_MEASURED_LEVEL

ルーチン内の計測レベルを指定する。レベル指定は以下の通り。

0 : そのルーチン中ではいかなる計測も行なわない
1 : そのルーチンの経過時間と実行回数のみを計測する
2 : 1に加え ルーチン中のDOループやCALL文等も計測する(既定値)

(8) DOLOOP_MEASURED_LEVEL

do ループの計測レベルを指定する。レベル指定は以下の通り。

0 : DOループは計測対象としない
1 : 最外側DOループのみ計測対象とする(既定値)
2 : ２重DOループまでを計測対象とする
3 : ３重DOループまでを計測対象とする
:

(9) CALL_MEASURED_LEVEL

call 文の計測レベルを指定する。レベル指定は以下の通り。

0 : CALL文は計測対象としない
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれないCALL文を計測対象とする(既定値)
2 : 全てのCALL文を計測対象とする

(10)MPI_MEASURED_LEVEL

MPIシステムルーチンコール文の計測レベルを指定する。レベル指定は以下の通り。

0 : MPIルーチンは計測対象としない
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれないMPIルーチンを計測対象とする
2 : 全てのMPIルーチンを計測対象とする(既定値)

(11)IO_MEASURED_LEVEL

read文 write文 open文の計測レベルを指定する。レベル指定は以下の通り。

0 : 入出力文は計測対象としない(既定値)
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれない入出力文を計測対象とする
2 : 全ての入出力文を計測対象とする

(12)IF_ROUTINE

このキーワードで指定されたルーチン内では計測レベルを変更する。変更する計測対象のキーワードとその計測レベルをIF_ROUTINEとEND_IFの間に記述する。

(13)INNER_PREFIX

KMtoolが利用する変数の接頭語を指定する。既定値はKM_で、普通は変更する必要はない。ユーザソースプログラム中の変数とKMtoolが利用する変数が、万一かち合った場合に、このキーワードを指定して、かち合う可能性のない接頭語に変更する。

6 KMtoolの評価

KMtool を実際に利用されている科学技術計算プログラムに適用し、評価を行った。対象としたプログラムは、日本原子力研究所で開発されたプラズマ粒子運動のシミュレータ[6]である。本シミュレータは約 7000 行 49 ルーチンで構成されており、MPI により並列化されている。今回はこのシミュレータの 4 並列版を VPP300 で動作させ、KMtool による性能評価を行った。図 10 に計測結果出力ファイル KM_output.000 の概要を示す。なお図中のカッコ内は説明文であり、実際の出力ファイルには記述されていない。

```
0004     0        ! Processors and Rank              ...（プロセッサ数とプロセッサ番号）
  3.5661136e+02 ! Total time                          ...（プログラム全体の経過時間）
        2498592 ! Counts of routines
        1221138 ! Counts of time get function
  1.1275062e-06 ! Time get function cost(1 call)
  1.3768407e+00 ! Time get function cost (Total)
              1 ! Output counts by KM_writ
  5.7690265e+00 ! MPI time                            ...（MPI通信時間合計）
  0.0000000e+00 ! I/O time                            ...（ファイル入出力時間合計）
0048     0        ! Profiling Programs

     （ルーチン名）      （全経過時間）  （正味経過時間）  （実行回数）  （MPI通信時間）
!    Routine name        Elapsed (sec)  Net time(sec)    Counts        MPI time
   1 AMYUDS              2.3744031e-01  3.3349245e-02        2  5.0387500e-04
   2 ATZFDO              3.4756850e-03  2.6393499e-04        1  0.0000000e+00
   3 ATZFLD              6.9921018e+00  3.7667988e-01      199  0.0000000e+00
   4 BDWGHT              0.0000000e+00  0.0000000e+00        0  0.0000000e+00
   5 BOUND               5.7506618e-01  5.7506618e-01      600  0.0000000e+00
                        （中略）
  48 XCHNG               1.4382083e+02  1.4323479e+02      600  5.8604381e-01
!    Total（総和）                       3.4730256e+02  +2498592.  5.7690265e+00

   1 AMYUDS        0009 AMYUDS.f     ...（ルーチン名   計測個所の個数   ファイル名前）
     （計測個所）                （行番号）（経過時間）          （実行回数）
!  0  Elapsed                            2.3744031e-01
   1 'DO I=1,NDST              '    25  4.4949807e-06 0.        2          0
   2 'DO I=1,NO_ME             '    30  3.3222807e-02 0.        2          0
   3 'MPI_ALLREDUCE            '    38  5.0161998e-04 0.        2          0
   4 'DO I=1,NDST              '    43  6.8700279e-06 0.        2          0
   5 'CALL PLOT1               '    62  1.9800099e-01 0.        2          0
   6 'CALL PLOTN               '    63  5.2374498e-04 0.        2          0
   7 'CALL PLOT2               '    64  4.4980748e-04 0.        2          0
   8 'CALL PLOT3               '    65  4.3304950e-03 0.        2          0
   9 'CALL PLOT4               '    66  2.7087000e-04 0.        2          0

   2 ATZFDO        0004 ATZFDO.f
!  0  Elapsed                            3.4756850e-03
(後略　各ルーチンの計測結果の後 プロセッサ番号 1,2,3 の結果が引き続き記載される)
```

図 10 KMtoolの計測結果出力ファイル

本シミュレータの実行時間は、性能解析を行わないオリジナルの状態で 342.2 秒である。一方、KMtoolによる生成された計測用ロードモジュールの実行時間は、図10にある通り356.6秒である。すなわちKMtoolによるオーバヘッドは、オリジナルの実行時間の4.2パーセントに抑えられている。従って、当初の目標であったオーバヘッドの少ない性能解析ツールが実現できたといえる。

7 おわりに

本稿では並列／非並列共用プログラム性能解析ツール KMtool の設計、利用方法及び試用結果について説明した。現在 KMtool は日本原子力研究所計算科学技術推進センター内で評価作業中であり、近い将来WWWを通じて外部に公開される予定である。

さらに平成 11 年度には、KMtool の FORTRAN90 対応版である KMtool90 の開発を予定している。既に科学技術計算プログラムの多くが FORTRAN90 で作成されており、KMtool90 の開発は急務といえる。現在 KMtool90 の設計に着手しているが、以下のことを考えている。

(1) FORTRAN 文法の解析ルールを、制御データファイル KM_control.dat 中で記述できるようにする。これは FORTRAN95 や FORTRAN2000 によるプログラムが流布するようになっても、最小限の変更で KMtool90 が利用できるようにするためである。
(2) 日本原子力研究所計算科学技術推進センターが開発している、並列分散科学技術計算環境 STA[5] から利用可能とする。
(3) 計測結果ファイルは現在の形式の他に、csv 形式等もサポートする。
(4) 利用方法や各種ファイル形式は現在の KMtool と互換性を持たせる。これにより KMtool ユーザの KMtool90 への移行が容易になる。
(5) モジュール化の徹底とオブジェクト指向技術の導入により保守性の向上を図り、さらに機能追加も容易に行えるようにする。

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、KMtool のようなツールは、その必要性にも関わらず、商業的には成立しにくい。従って当センターが率先して汎用性のある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。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研究員各位には KMtool ベータ版を試用して頂き、多数の貴重な意見を給わりました。また同センター相川裕史次長には、KMtool 開発にあたり多大のご支援を給わりました。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] 渡部弘 折居茂夫 熊倉利昌 滝川好夫, "プログラム並列化支援解析ツール kpx(その2)", JAERI-Data/Code 98-016

[3] 渡部弘 折居茂夫 滝川好夫 熊倉利昌, "並列用性能評価ツールとその応用", 計算工学講演会論文集 Vol3,No.1,p47 1998 年 5 月

[4] PALLAS "VAMPIR-Visualization and Analy-sis of MPI Resources" http://www.pallas.de

[5] 武宮博 今村俊幸 太田浩史 川崎琢治 小出洋 樋口健二 笠原博徳 相川裕史, "並列分散科学技術計算環境 STA(1)-(4)", 計算工学講演会論文集 Vol3,No.1,p73 1998 年 5 月

[6] 徳田ほか, "トカマク・プラズマにおけるハイブリッド・シミュレーション", 日本物理学会年会第 53 年会, 1pP5(1998)

(付録１　KMtool 簡易説明書)

並列／非並列共用FORTRANプログラム性能解析ツール"KMtool"簡易説明書

日本原子力研究所 計算科学技術推進センター　平成１０年１２月１６日

１．はじめに

KMtoolは日本原子力研究所計算科学技術推進センター(以下当センターと略す)が、並列処理技術推進のための共通基盤として開発した、FORTRANプログラム性能解析ツールです。計測対象プログラムは、逐次型またはMPIにより並列化されたFORTRANソースプログラムです。

本ツールはFORTRAN77仕様及び、多くの計算機で採用されている拡張FORTRAN77仕様に基づくプログラムを想定して、作成されています。またツール自体もFORTRAN77で記述されており、移植性にたいへん優れています。現在までに当センター所有のSP2, SR2201, VPP300, SX-4, T94において、その動作が確認されています。さらに計測結果は専用ポストプロセッサKMviewにより、WEBブラウザ上でグラフ表示することが可能です。

KMtoolには以下の機能があります。

(1) メインルーチン、サブルーチン、外部関数の、全経過時間(elapsed time)と正味の経過時間(net time)及び実行回数(counts)を計測する。ルーチンの全経過時間(elapsed time)には、そのルーチンがcallしているサブルーチンの経過時間が含まれる。一方、正味の経過時間(net time)にはcallしているサブルーチンの経過時間は含まれない。例えば、サブルーチンsub1がサブルーチンsub2をcallしており、次のような経過をたどるものとすると、

```
Time      T0            T1                T2              T3
          +============>+-----------------+==============>+------>
Event     start sub1     call sub2         ^               return
                         |                 |
                         v                 |
                         +================>+
                         start sub2        return
```

sub1の全経過時間(elapsed time)はT3 - T0、正味の経過時間(net time)は(T3-T2) + (T1-T0)となる。

(2) DOループ、CALL文、入出力文、MPIルーチンコールの経過時間(elapsed time)及び実行回数(counts)を計測する。

(3) (2)に加えソースファイルにKMtool指示行を挿入することにより、プログラム中任意の部分の経過時間及び実行回数を計測する。

(4) 並列化プログラムの場合には、並列実行結果から加速率や並列効率を予測する。

本説明書ではメインルーチン、サブルーチン、外部関数をあわせて、ルーチン(Routine)と総称することにします。

本説明書の構成は以下のようになっています。

本ツールは日本原子力研究所計算科学技術推進センター並列処理支援技術開発グループが作成したフリーウェアであり、著作権は全て日本原子力研究所に帰属します。また第三者への再配布はご遠慮下さい。本ツールの使用により何らかの損害や不利益を被っても、日本原子力研究所は一切責任を負いません。

２．インストール方法

KMtoolをインストールするためには、当センターのファイルサーバ等からtar形式のファイルを入手し、それを展開した後、インストール用シェルスクリプトkminstを用います。以下にその方法を記述します。

(1)当センターのファイルサーバ(ccsemfa, 202.241.61.101)よりtarファイル(/nhome001/g0188/j0000/KPX/KMtool.tar)を、インストール対象計算機のホームディレクトリに置く。

```
host> cd

host> ftp 202.241.61.101

ftp> bin

ftp> cd ~j0000/KPX
                OR
     cd /nhome001/g0188/j0000/KPX

ftp> get KMtool.tar

ftp> bye
```

(2)インストール対象計算機のホームディレクトリにKMtool用ディレクトリを作成し、KMtool.tarをそのディレクトリ下で展開する。再インストールの場合はディレクトリKMtoolを削除した後、下記を実行する。

```
host> mkdir KMtool

host> cd KMtool

host> tar xvfo ~/KMtool.tar
```

(3)インストーラに計算機名の引数を与えて起動し、KMtoolのロードモジュールを作成する。現状で受付可能な計算機名の引数は、以下のどれか一つである。

```
host> kminst HOST    (または csh kminst HOST)
      (HOSTにはsp2, sr2201, vpp300, sx4, t94, otherのどれか一つを指定)

sp2     : SP2(ＩＢＭ)用のKMtoolロードモジュールを作成
```

sr2201 ：SR2201(日立)用のKMtoolロードモジュールを作成
vpp300 ：VPP300(富士通)用のKMtoolロードモジュールを作成
sx4 ：SX-4(ＮＥＣ)用のKMtoolロードモジュールを作成
t94 ：T94(ＣＲＡＹ)用のKMtoolロードモジュールを作成
other ：上記以外のマシン用のKMtoolロードモジュールを作成

インストール対象の計算機がSP2、SR2201、VPP300、SX-4、T94以外の場合でも、FORTRANコンパイラ名と時刻取得ルーチン名が同じ計算機があれば、その計算機名をHOSTに指定できる。例えばインストール対象計算機がsx5の場合には、HOSTにはsx4と指定すればよい。

(4) (3)のHOST引数にSP2、SR2201、VPP300、SX-4、T94が指定できない場合は、HOSTにotherを用いる。この場合、インストーラがFORTRAN77コンパイラ名の入力を要求するので、適切なコンパイラ名を指定する。

```
host> kminst other
========= Install KMtool for the OTHER =========

Please input FORTRAN77 compiler name. (RETURN key : f77)

fort77              ... コンパイラ名がfort77の場合
```

またkminstの実行が終了した後、KMtoolシステムルーチンファイルKM_@system.f中にある時刻取得関数KM_gettを編集して、clockという時刻取得ルーチン名を適切なルーチン名に変更する。変更箇所は１箇所だけである。

```
host> vi KM_@system.f

      function KM_gett                                          ()
      implicit none
!KM_gett
      real*8   KM_gett
c
c     KM_gett is a double precision function of KMtool to get time
c     at the moment. This function must return time in seconds,
c     so user may need to modify this function.
c
!(CS)KM_cget
!KM_cget
      common /KM_cget                                           /
     -              KM_gcnt
      integer        KM_gcnt
!(CE)KM_cget
c
      real*8   elp_time
c     real*8   cpu_time
c
c     ------------------------                                  !STAND
      call clock  (elp_time)                                    !STAND
          ^^^^^ -> このclockを変更する
```

以上の作業は、インストール対象計算機がSP2、SR2201、VPP300、SX-4、T94の場合には必要ない。

(5) KMtoolディレクトリにパスを設定する。具体的には.cshrcの最後に次の行を付け加えればよい。

```
set path = ( $path ~/KMtool )
```

3. ファイル構成

インストールが正常に終了すると、ディレクトリKMtool下に以下のようなファイルが生成されます。

```
KMtool/                          :KMtool directory
        KM_@system.f             :KMtoolシステムルーチンファイル
        KM_control.dat           :KMtool制御データファイル
        README.euc :KMtool簡易説明書 本ドキュメント eucコード
        README.sjis              :KMtool簡易説明書 本ドキュメント シフトJISコード
        README.txt :KMtool簡易説明書 英語版
        kminst                   :インストーラ(インストール後は消去可)
        kmrun                    :KMtool実行用シェルスクリプト
        kmtool                   :KMtool本体
```

4. KMtoolの利用方法

KMtoolを利用するには、附属のシェルスクリプトkmrunを用います。本節では、kmrunによる計測方法及び計測部分の変更方法について説明します。

4. 1 計測処理の流れ

(1) kmrunはユーザのFORTRANソースプログラムに、KMtool用の経過時間及び実行回数を計測するためのルーチンコール等を挿入して、新しく計測用FORTRANプログラムを生成する。計測用ソースファイル名はユーザソースファイル名にKM_という接頭語がついた名前になる。例えばユーザソースファイルがmain.fの場合には、KM_main.fという計測用ソースファイルが生成される。

(2) 同時に、main.f計測用のKMtoolシステムルーチンファイルKM_@system.fが生成される。

(3) kmrunが終了した時点で、もしKM_message.txtというテキストファイルが出力されていれば、その内容を確認する。そのファイルにはKMtoolのエラーメッセージや警告メッセージが記述されている。

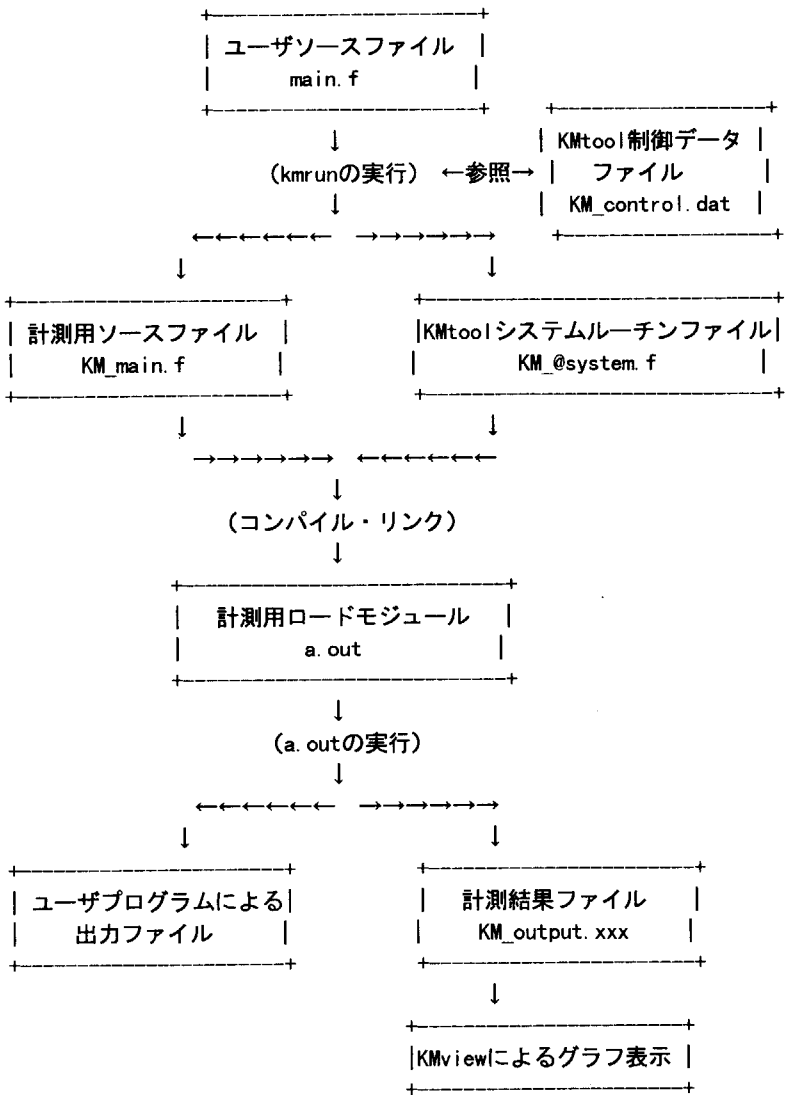
(4) 問題がなければ、生成された計測用ソースファイルKM_main.fとKMtoolシステムルーチンファイルKM_@system.fをコンパイル・リンクし、計測用ロードモジュールを作成する。

(5) 作成された計測用ロードモジュールを通常の方法で実行する。

(6) 計測用ロードモジュールの終了と同時に、KMtool計測結果ファイルKM_output.xxxが生成される。xxxは000から999まで自動的に番号が付けられる。ユーザはこのファイルを見ることにより、プログラム各部の経過時間及び実行回数を知ることができる。

(7) KMtool用ポストプロセッサKMviewを利用することにより、計測結果をWEBブラウザからグラフ表示することができる。現時点で動作が確認されているWEBブラウザはNetscape Communicator 4.5である。

（処理の流れ図）



4．2 kmrunの実行方法

(1)FORTRANソースファイルのあるディレクトリに移動する。

(2)kmrunを起動し、計測用ソースファイルとKM_@system.fを生成する。

```
host> kmrun *.f
          ... 全てのFORTRANソースファイルを計測対象とする
              KM_*.fとKM_@system.fが生成される
```

あるいは次のように、一部のソースファイルにのみ適用することもできる。

```
host> kmrun main.f aaa.f
          ... main.fとaaa.fを計測対象とする
              KM_main.f KM_aaa.f KM_@system.fが生成される
```

一部のソースファイルに適用する場合、プログラムが終了する可能性のあるルーチン(メインルーチンやSTOP文のあるルーチン)を含むソースファイルは、必ずkmrunの引数で指定しなければならない。そうしないと、計測結果ファイルKM_output.xxxが

出力されない可能性がある。

＊＊＊＊＊＊＊＊＊＊＊＊＊　注意！！　＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
KMtoolはカレントディレクトリ下に、KM_接頭語付きのファイルを一時的に
数多く作成します。従ってもしKM_接頭語付きのファイルが既に存在する場合、
上書きされる可能性があります。そのため、KM_接頭語付きのファイルはあらか
じめリネームするか、別の場所にコピーしておくことを強くお勧めします。
＊＊＊＊＊＊＊＊＊＊＊＊＊　注意！！　＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

(3)計測用ソースファイルの出力ディレクトリを変更したい場合は、kmrunに
-tdオプションをつけて起動する。-tdは最初の引数でなければならない。
-tdで存在しないディレクトリが指定された場合、新たに作成される。

```
host> kmrun -td test *.f
          ... test/の下に計測用ファイルが生成される
```

(4)計測部分を変更したい場合、出力ディレクトリ上にKMtool制御データファイル
KM_control.datを用意して編集するか、直接ユーザソースファイルにKMtool制御
指示行を埋め込む。出力ディレクトリとはkmrunの-tdオプションで指定された
ディレクトリであり、-tdオプションがなければカレントディレクトリである。
計測部分変更の詳細は、次の４．３を参照されたい。

(5)同じディレクトリで再度kmrunを実行した場合、下記に示すメッセージが出力される
場合がある。これは前回のKMtool制御データファイルKM_control.datが残っている
ためであり、計測部分を変更する等の理由でユーザがKM_control.datを編集した
場合にはyを指定する。nを指定した場合は~/KMtoolにある既定値のKM_control.dat
により上書きコピーされ、KMtoolの動作制御に利用される。

```
host> kmrun *.f

The control file ./KM_control.dat already exists.
Use this control file? (y/n)

y      ... KMtoolの制御のために./KM_control.datを使用する
n      ... ~/KMtool/KM_control.datをコピーして使用する
```

(6)kmrunが終了した時点で、もしKM_message.txtというテキストファイルが出力されて
いれば、その内容を確認する。そのファイルにはKMtoolのエラーメッセージや警告
メッセージが記述されている。

(7)生成された計測用ソースファイル及びシステムルーチンファイルKM_@system.fを
コンパイル・リンクし、計測用ロードモジュールを作成する。その後、通常の方法
で計測用ロードモジュールを実行する。

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

host> a.out             ... 計測用ロードモジュールを実行
```

(8)計測用ロードモジュールが終了すると同時に、計測結果ファイルKM_output.xxxが
出力される。xxxは000から999まで自動的に番号が付けられる。

(9)KMviewを利用することにより、計測結果をWEBブラウザからグラフ表示することが
できる。KMviewの利用方法は次の通りである。

(9-a)当センターのファイルサーバ(ccsemfa, 202.241.61.101)から、HTMLファイル
"appletload.html"とKMview本体のファイル"KMview.jar"を手元のパソコン等

の端末にコピーする。

```
host> ftp 202.241.61.101

ftp> bin

ftp> cd ~j0000/KPX
                OR
     cd /nhome001/g0188/j0000/KPX

ftp> get appletload.html

ftp> get KMview.jar

ftp> bye
```

(9-b) KMtoolによる計測結果ファイル"KM_output.xxx"を手元の端末にコピーする。

(9-c) 手元の端末のWEBブラウザを起動し、URLにappletload.htmlを指定する。なお動作が確認されているWEBブラウザは、Netscape Communicator 4.5である。

4．3 計測部分の変更方法

KMtoolの既定値の計測部分は以下の通りです。

(1) kmrunの引数として与えられたFORTRANソースファイル中のメインルーチン、サブルーチン、外部関数の、全経過時間(elapsed time)と正味の経過時間(net time)及び実行回数。

(2) (1)のルーチン中の、最外側DOループの経過時間(elapsed time)及び実行回数。

(3) (1)のルーチン中の、DOループに含まれていないCALL文の経過時間(elapsed time)及び実行回数。

(4) (1)のルーチン中の、MPI_WTIMEとMPI_WTICK以外の全MPIルーチンの経過時間(elapsed time)及び実行回数。

これらの計測部分を変更したい場合は、KMtool制御ファイルKM_control.datを編集するか、直接ユーザのFORTRANソースファイルにKMtool制御指示行を埋め込みます。以下ではそれらの方法を説明します。

4．3．1 特定のサブルーチンや外部関数を計測対象外にしたい場合

KM_control.datのIGNORED_ROUTINESという項目に、計測対象外にしたいルーチン名を指定する。ルーチン名はコンマ(,)により複数個指定できる。ここで指定されたルーチンは、そのルーチン内はもちろん、そのルーチンをCALLする側でも計測対象外になる。

```
IGNORED_ROUTINES = MPI_WTIME,MPI_WTICK,sub1,func2
... MPI_WTIME、MPI_WTICK、サブルーチンsub1、外部関数func2が計測対象外
```

4．3．2 全てのDOループ,CALL文,MPIルーチン,入出力文を計測対象外にしたい場合

計測のオーバヘッド時間を減らすために、全てのDOループ、CALL文、MPIルーチン、

入出力文を計測対象外にしたい場合がある。その場合はKM_control.dat中のROUTINE_MEASURED_LEVELという項目を1に指定する。これにより、各ルーチンの経過時間と実行回数のみが計測される。

```
ROUTINE_MEASURED_LEVEL = 1
```

ROUTINE_MEASURED_LEVELは0,1,2のどれかの値を取り、各々の意味は次の通り。

0 : そのルーチン中ではいかなる計測も行なわない
1 : そのルーチンの経過時間と実行回数のみを計測する
2 : 1に加え、ルーチン中のDOループやCALL文等も計測する(既定値)

4．3．3 計測するDOループを変更したい場合

KMtoolは計測対象DOループの既定値を最外側DOループとしている。すなわち次のコード例では、do 100のみが計測され、do 200は計測されない。

```
        subroutine sub1(a,b,c)
        do 100 i = 1,100      ------------------------+
           do 200 j = 1,200   -------+                |
                 ...                 |                |
                 ...            計測されない      計測される
                 ...                 |                |
 200       continue            -------+                |
 100    continue               ------------------------+
      return
        end
```

do 200も計測したい場合は、KM_control.dat中のDOLOOP_MEASURED_LEVELという項目を2に指定する。

```
DOLOOP_MEASURED_LEVEL = 2
```

DOLOOP_MEASURED_LEVELは0以上の値を取り、意味は次の通り。

0 : DOループは計測対象としない
1 : 最外側DOループのみ計測対象とする(既定値)
2 : ２重DOループまでを計測対象とする
3 : ３重DOループまでを計測対象とする
:
:

4．3．4 計測するCALL文を変更したい場合

KMtoolは既定値では、DOループに含まれないCALL文のみを計測対象としている。正確にいうと、4．3．3で説明したDOLOOP_MEASURED_LEVELで計測指定された最も内側のDOループに含まれないCALL文を計測対象とする。例えば、DOLOOP_MEASURED_LEVELが2と指定された場合は、次のようになる。

```
subroutine sub2(a,b,c)
call sub3              ---> 計測される
do 100 i = 1,100
   call sub3           ---> 計測される
   do 200 j = 1,200
```

```
                ...
            call sub3        ---> 計測されない
                ...
200       continue
100    continue
  return
     end
```

DOLOOP_MEASURED_LEVELに関わらず、全てのCALL文を計測対象としたい場合は
CALL_MEASURED_LEVELという項目を2に指定する。

```
CALL_MEASURED_LEVEL = 2
```

CALL_MEASURED_LEVELは0,1,2のどれかの値を取り、意味は次の通り。

0 : CALL文は計測対象としない
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれないCALL文を計測対象とする(既定値)
2 : 全てのCALL文を計測対象とする

４．３．５ 計測するMPIルーチンを変更したい場合

KMtoolはMPI_WTIMEとMPI_WTICK以外の全MPIルーチンを計測対象の既定値としている。計測するMPIルーチンを変更したい場合は、MPI_MEASURED_LEVELという項目を変更する。MPI_MEASURED_LEVELの詳細は４．３．４のCALL_MEASURED_LEVELと同様である。

0 : MPIルーチンは計測対象としない
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれないMPIルーチンを計測対象とする
2 : 全てのMPIルーチンを計測対象とする(既定値)

４．３．６ 計測する入出力文を変更したい場合

KMtoolは既定値では、全ての入出力文(read文、write文、open文等)を計測対象外としている。計測する入出力文を変更したい場合は、IO_MEASURED_LEVELという項目を変更する。IO_MEASURED_LEVELの詳細は４．３．４のCALL_MEASURED_LEVELと同様である。

0 : 入出力文は計測対象としない(既定値)
1 : DOLOOP_MEASURED_LEVELで指定されたDOループに含まれない入出力文を計測対象とする
2 : 全ての入出力文を計測対象とする

４．３．７ ある特定のルーチン内でのみ計測部分を変更したい場合

４．３．２から４．３．６で、全ルーチンにおける計測部分を変更する方法を説明したが、ある特定のルーチン内でのみ計測部分を変更したい場合には、IF_ROUTINEでルーチン名を指定する必要がある。ルーチン名はコンマ(,)により複数個指定できる。計測対象の変更方法は４．３．２から４．３．６までで説明した通りだが、変更する項目をIF_ROUTINEとEND_IFで囲まなければならない。

例として、サブルーチンsub1とsub2中では２重DOループまでと全てのCALL文を計測対象としたい場合、KM_control.datに次を加える。

```
IF_ROUTINE = sub1, sub2
    DOLOOP_MEASURED_LEVEL = 2
    CALL_MEASURED_LEVEL   = 2
END_IF
```

KM_control.dat中にIF_ROUTINE～END_IFブロックは何個あってもよい。

4．3．8 ある特定箇所のみ計測したい場合

制御データファイルKM_control.datに関係なく、ある特定箇所を計測したい場合は、ユーザのFORTRANソースファイルにKMtool計測指示行を直接埋め込む。計測開始箇所には1カラム目から、次の指示行を記述する。

```
!KM_measure start
```

計測終了箇所には1カラム目から、次の指示行を記述する。

```
!KM_measure end
```

5．困った時は

(1)計測用ソースファイルが生成されない時

kmrun実行後に計測用ソースファイルKM_xxxx.fが生成されない場合は、出力ディレクトリに作成されるKMtoolメッセージファイルKM_message.txtを参照してください。計測用ソースファイルが生成されない理由が記述されています。

KMtoolに入力されるFORTRANソースファイルはFORTRAN77の文法に従っている必要があります。もし計測用ソースファイルが生成されない場合は、マシン固有の拡張FORTRAN仕様を用いていないか確認して下さい。ただし、DO～ENDDOのような広く一般的に受け入れられている拡張仕様に対しては、正常に動作するようになっています。

(2)計測用ソースファイルがコンパイルできない時

KMtoolにより生成される計測用ソースファイルには、KM_xxxxという形式の変数名、配列名、common名、サブルーチン名が多数挿入されています。従ってもし、オリジナルのソースプログラムにもKM_xxxxという形式の変数やルーチン等が使用されている場合、これらの名称がかちあっている可能性があります。この場合は制御データファイルKM_control.dat中のINNER_PREFIXを、かちあう可能性のない名称に変更して下さい。

```
INNER_PREFIX = JAERI_    ... JAERI_xxxxという形式の変数名になる
```

(3)計測用ロードモジュールが作成できない時

計測用ロードモジュールを作成するには、計測対象外のソースファイルのコンパイルが必要な場合もあります。例えば、main.fとaaa.fからa.outができるプログラムがあって、main.f中のルーチンのみを計測したい場合は、次のような手順となります。

```
> kmrun main.f    ... KM_main.fとKM_@system.fが生成される

> f77 KM_main.f aaa.f KM_@system.f    ... 計測用のa.outが生成される
```

(4) 計測用ロードモジュール実行中に異常終了する時

KMtoolは計測結果ファイル用のファイル番号として、99番を使用しています。そのため、もしユーザプログラム中で既に99番を使用している場合は、異常終了する可能性があります。この場合は制御データファイルKM_control.dat中のOUTPUT_FILE_NUMBERを99から別の番号に変更して下さい。

OUTPUT_FILE_NUMBER = 98 ...ファイル番号を98にする

その他にも、KMtoolのバグが原因で異常終了する場合もありえます。その場合は下記連絡先に御連絡下さい。

E-mail: watanabe@koma.jaeri.go.jp

(5) 計測用ロードモジュール実行後に計測結果ファイルが出力されない時

KMtoolはプログラムの終了直前に、計測結果出力用のKMtoolシステムルーチンをcallし、KM_output.xxxというファイルに出力します。従って、もし計測対象外のルーチン(つまりkmrunが実行されていないルーチン)でプログラムが終了した場合には、計測結果が出力されないことになります。これを回避するには、STOP文やCALL EXIT文を含む、全てのプログラム終了する可能性のあるルーチンを計測対象にします。すなわち、STOP文やCALL EXIT文を含む全てのソースファイルにkmrunを実行する必要があります。

(6) 計測用ロードモジュールの実行がなかなか終了しない時

4．3で説明した方法で計測部分を絞って再計測して下さい。その際、どのルーチンで実行が停滞しているかを特定したい場合は、制御データファイルKM_control.dat中のTRACE_FILE_NUMBERを1以上の値に変更して下さい。プログラム実行中にその番号のトレースファイルが生成され、どのルーチンが実行されているか特定できます。

TRACE_FILE_NUMBER = 20 ...fort.20やft.20等のトレースファイルが出力される

(7) 経過時間をより正確に計測したい時

KMtoolはツールによるオーバヘッド時間を少なくするため、10万回以上計測される箇所では、総経過時間を10万回までの経過時間の平均値から推測しています。従ってその箇所の経過時間にはある程度の誤差があります。もしより精度のよい計測を行ないたい場合は、KMtool制御データファイルKM_control.dat中のTIME_PREDICTION_MODE_NUMBERを10万以上の適切な値に変更して下さい。例えば次のようにすると100万回まで直接計測されるので、その実行回数以下の計測箇所では正確な経過時間が得られます。

TIME_PREDICTION_MODE_NUMBER = 1000000

なおTIME_PREDICTION_MODE_NUMBERで指定された回数以上の計測がなされた箇所では、計測出力結果ファイル中の実行回数(Counts)が＋記号つきで表示されます。

(8) ある特定のルーチンのリターン時に計測結果を出力したい時

KM_control.datのPRINT_OUT_ROUTINESという項目を指定することにより、特定のルーチンからリターンするたびに、その時点までの計測結果が出力されます。

PRINT_OUT_ROUTINES = sub1, sub2 ... sub1とsub2のリターン時に
計測結果ファイルが出力される

これは、C言語で書かれたメインルーチンから呼ばれるFORTRANサブルーチンの計測を行なう場合などに利用します。

(9) 時刻取得ルーチンを変更したい時

KMtoolの既定時刻取得ルーチンには各マシンの経過時間(Wall Clock)取得ルーチンを採用しています。これを例えばCPU時間に変更したい場合は、KM_@system.fのKM_gettをCPU時間で計測するように編集します。詳細は2. (4)を参照してください。

(連絡先)
日本原子力研究所 計算科学技術推進センター 並列処理支援技術開発グループ
担当 渡部弘(わたなべひろし) watanabe@koma.jaeri.go.jp

(付録2　KMviewの利用方法)

5　KMviewの利用方法

並列／非並列プログラム共用性能解析ツールは、次の三つの部分からなる。

1. ユーザプログラム計測機能
2. ユーザプログラム性能予測機能
3. GUIによる出力結果表示機能

本マニュアルは、3.の出力結果を表示する機能（以下このシステムをKMviewと呼ぶ。）についてシステム構成、使用説明及び、プログラム構造について記述してある。このシステムは、Java言語のApletを継承して作成しており、Netscape[1]などのWebヴューア上で実行することができる。

5.1 概要

ユーザプログラムの計測及び、予測機能プログラム（KMtool）を実行することで作成される出力結果ファイルを入力し数値を棒グラフで表示する。表示できる計測結果は

- 各並列プロセスでの並列実行時間及び、通信時間
- プログラムのルーチン単位での経過時間、正味の経過時間及び、実行回数
- ルーチン内のDOループ、CALL文、入出力、通信単位のそれぞれ、経過時間、正味の経過時間及び、実行回数
- 上記の2.及び3.の項目の全並列プロセスでの値

である。各数値は、棒グラフの長さに変換され表示される。表示順は、解析結果の順番、値の小さい順及び、値の大きい順を選択することができる。また、表示したグラフを、ヒストリとして保存する機能があり、ボタンにより任意の時点のグラフを再現できる。また、マウスポインタを棒グラフの中に移動すると、棒グラフの数値と、その数値の全体での割合がパーセントで表示される。

5.2 システムの構成について

システムを構成するファイルは、以下のファイルである。

- KMviewの実行に必要なすべてのJavaクラスファイルを含んだアーカイブファイル。アーカイブ形式は、jar形式のファイルである。
- 上記、jarファイルをアプレットとして起動するためのHTMLファイル。アプレットを起動するために必要最低限の記述のみである。
- 計測プログラムによって出力された解析結果ファイル（KMviewが入力するために専用のフォーマットのテキストファイルである。フォーマット形式については付録5.6を参照のこと）

上記の、HTMLファイルの内容を示すと、

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<TITLE>
HTML テスト ページ
</TITLE>
</HEAD>
<BODY>
KMview can be used on WEB browser that can execute Java applet.<BR>
<APPLET
  CODEBASE = "."
  CODE     = "KMview.KMview.class"
  NAME     = "KMview Sample Apllet"
  WIDTH    = 600
  HEIGHT   = 400
  HSPACE   = 0
  VSPACE   = 0
  ALIGN    = middle
  ARCHIVE  ="KMview.jar"
>
<PARAM NAME = "Anl_filenm" VALUE = "http://[host 名]/kmview/KM_output.000">
<PARAM NAME = "Min_FontSize" VALUE = "14">
</APPLET>
</BODY>
</HTML>
```

上記のようにjarファイルは、HTMLファイルのアプレットタグ内の"ARCHIVE"によって参照される。また、Kmviewアプレットが起動時に入力する解析ファイル名を"PARAM NAME"タグに与える。WebサーバがUNIXシステムの時には、ファイル名は英大文字と小文字は区別されるので注意する必要がある。

注意 1 *Java*のアプレットの制限上、ファイルを入力する時には、アプレットの存在する*URL*と入力ファイルの*URL*が同じでなければならない。したがって、上記の*3*種類のファイルは、同じディレクトリに置かなければならない。ネットワークを介さずにローカルに起動する場合もこの制限が当てはまる。この場合には、*"http://[host 名]"*のところを、*"file://"*に変更する必要がある。

5.3 使用方法について

NetscapeなどのWebブラウザを起動し、前の章で記述したHTMLファイル(12ページ)のURLを指定する。あるいは、そのHTMLファイルへのリンクが存在するHTMLファイルを開ける。しばらくして、Javaが起動され以下のような画面が現れる。

表示されているグラフの値は、同じディレクトリに存在する、解析ファイルKM_output.000の内容である。もし、起動時に、"PARAM NAME" タグに与えられたファイルが存在しない場合には、KMviewプログラムの内部で作成されたサンプルデータが表示される。この場合には、[file] メニュー－＞[Open] をクリックし存在するファイルのファイル名を入力する。

5.3.1 アプレットの操作方法

アプレットの操作は、アプレットの上部に表示されるメニューとアプレット上のボタンによりグラフの種類、表示計測量、X軸のスケールなどを変更できる。

メニューの操作方法 アプレットのメニュー上のラベル文字をクリックすると選択できるメニューアイテムを含んだポップアップメニューが表示される。ポップアップメニュー上のメニューアイテムをクリックすると、処理を実行する。

File 新規ファイルの入力及び、アプレットの終了。

file: 新しい解析ファイルを読込む。このメニューアイテムをクリックするとファイル名を入力するテキストボックスが表示されるので、ファイルのURLを入力してEnterキーを押下する。

Quit: Webブラウザで表示している時には動作しない。Aplletviewerで表示している時には、アプレットを終了する。

ProgP. プログラムプロファイルの選択と全実行時間の選択。

Program Profile: プログラム内のルーチン毎の計測結果を表示する。

RoutP. プログラムのルーチン内の計測対象の計測結果を表示する。このラベルをクリックすると、ルーチン名を含んだポップアップメニューが表示され、表示するルーチンを選択する。

AllPro. 全プロセスでの計測値が、PE毎に表示される。表示可能な計測値は、ルーチン毎でもルーチン内のDOループやCALL文の単位でも可能である。

Option このラベルをクリックして現れるポップアップメニューは、トグルになっており、一度にはどちらかのメニューがアクティブになっている。アクティブなメニューには"*" の印が付いている。

Total Time 通信時間を含んだ全体の計測時間を表示する。

Comp & Comm 全体の計測時間から、通信時間を色の違った棒グラフとして表示する。デフォールトでは、全体の計測時間から通信時間を差し引いた値をオレンジで、通信時間を水色で表示する。

Comp & Comm 上記の逆で、通信時間を先に表示する。

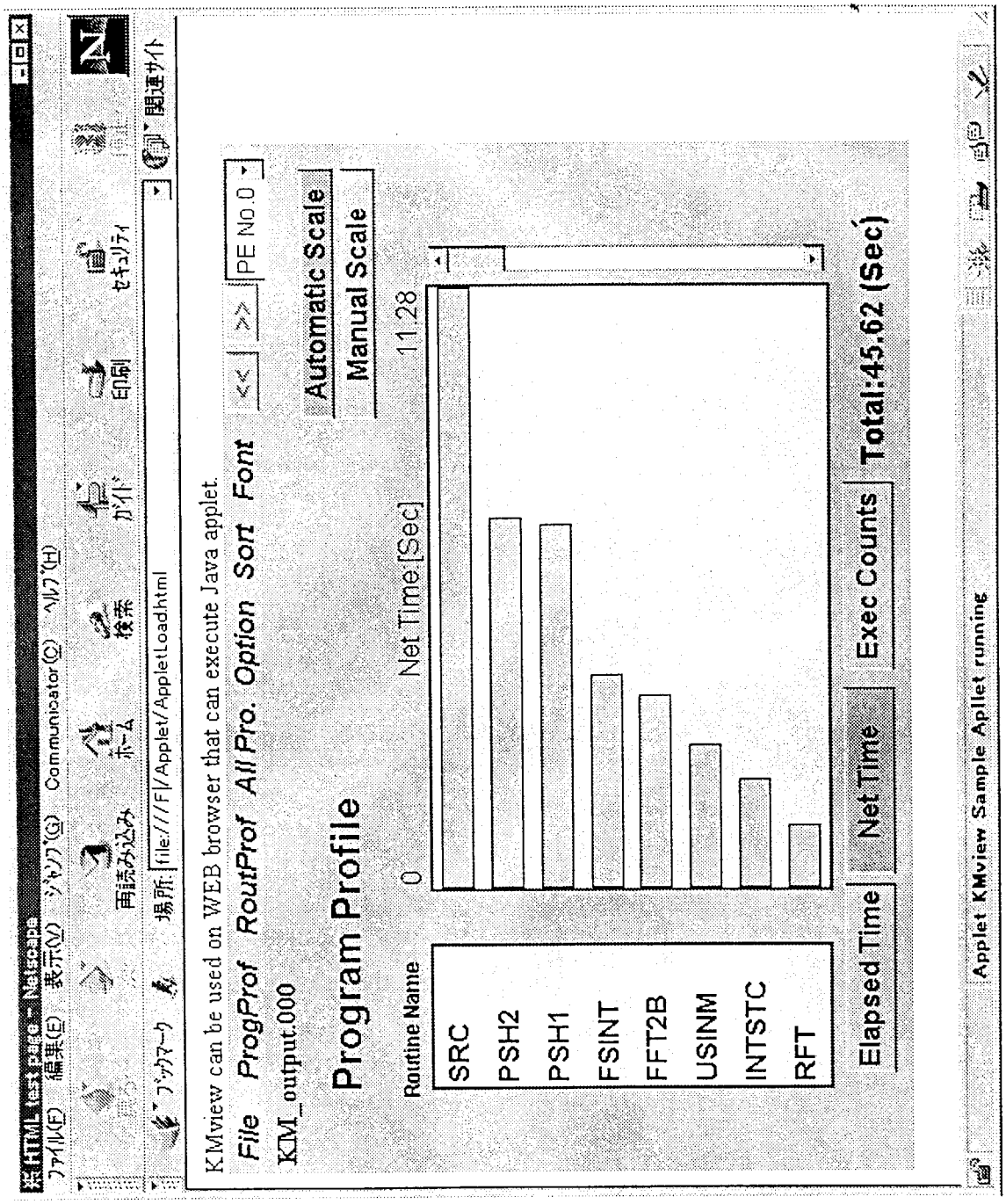


図 1: KMviewの起動画面

Sort このラベルをクリックして現れるポップアップメニューは、トグルになっており、一度にはどれか一つのメニューがアクティブになっている。アクティブなメニューには"*"の印が付いている。

Original Order 解析ファイルに出力されている順番で表示する。

Ascending Order 計測時間の小さい順に表示する。

Descending Order 計測時間の大きい順に表示する。

Font 表示フォントの最小の値を選択する。KMviewでは、選択された数値にしたがってフォントを拡大、縮小表示を行う。起動時のフォントサイズは、アプレットをロードするHTMLファイル内の"Min_FontSize"パラメタで与える。

<<及び>>ボタン グラフ表示ののヒストリを再表示するのに使用する。"<<"ボタンは、以前に表示されていたグラフを再現する。">>"ボタンは、一つ後のグラフ履歴を表示する。表示しようとするグラフの履歴が存在しない場合にはボタンが選択できなくなる。

コンボボックス PEの番号を指定する。全プロセスでの計測結果が表示されている時には、機能しない。それ以外の場合に、指定したPEでの計測結果が表示される。

以下に、メニューの図を示す。

図 2: KMviewのメニューについて

アプレット上のボタンについて　アプレット上には、X軸のスケールを設定するためのボタンと表示計測量を変更するためのボタンがある。X軸のスケールを設定するためのボタンは、グラフの上部にある[Automatic Scale]と[Manual Scale]ボタンで、表示計測量を変更するためのボタンは、グラフの下部にある[Elapsed Time]、[Net Time]及び、[Exec Counts]ボタンである。両方のボタンとも表示されている状態を示すボタンの背景色が黄緑色に設定されている。それぞれのボタンを以下に示す。

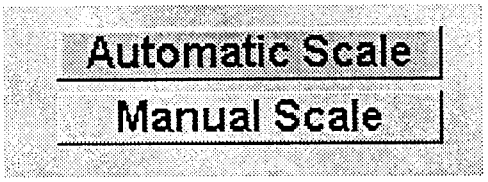


図 3: X軸のスケールを設定するボタン

図 4: 表示計測量を変更するためのボタン

マニュアルでX軸のスケールを設定するには、[Manual Scale]ボタンを押下し以下のダイアログボックス（図5）の最大値及び、最小値を設定して[OK]ボタンを押下する。ダイアログが開いた時には、現在表示されているグラフの最大値及び、最小値が表示されている。[Cancel]ボタンを押下した時にはX軸のスケールは変更されずに、[Automatic Scale]が選択される。

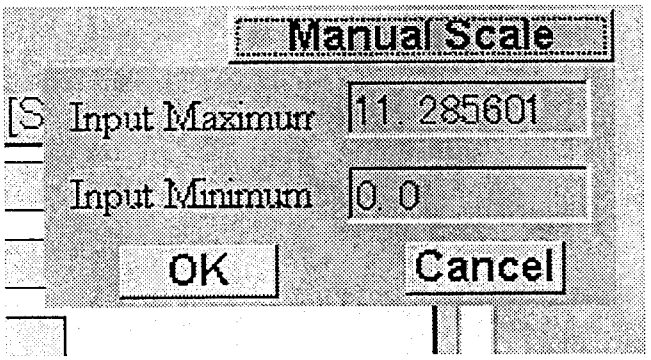


図 5: マニュアルスケールの入力ダイアログ

5.4 プログラムの構造について

この章では、KMviewのメンテナンスのために必要な情報を記述する。開発環境、クラス構造及び、イベント処理について述べる。

5.5 開発環境

Kmviewプログラムの開発は、INPRISE Japan（旧社名Borland Interbnational）社のJBuilder2 Professional版を使用して行った。したがって、KMviewプログラムのソースコードを編集しコンパイルするには、JBiulder2を使用するのがもっともよい方法である。KMviewは、INPRISE社が提供するJavaのクラスも使用しているのでJBuilder2を使用しない場合には、ソースコードをINPRISE社から入手する必要がある。

注意 2 ライセンスに対しては、*Borland*社の社名の入った*Java*ソースコードは、配布することはできません。コンパイル後のクラスファイルは再配布可能です。また、*WOODLAND*社の社名の入ったソースコードについては、ソースコードの著作権を放棄し、フリーウエアとします。ソースの修正、再配布は自由です。

5.5.1 プログラム構造について

Java言語は、単一継承を基本にしたオブジェクト指向言語である。その構造は、わかりやすくすっきりした構造になっている。KMviewプログラムもオブジェクト指向設計を基本にしてオブジェクトを設計し、実装した。以下にオブジェクト構造を示す。

```
Apllet
  +-- KMview

Object
  +-- XYlocation
  |     +-- myRectangle
  |           +-- Draw_rect
  |                 +-- Draw_string
  |                 +-- Bar_graph
  +-- AnalysisData
  |     +-- CreateAnalysisData
  +-- Draw_data
  |     +-- Draw_data1
  |           +-- Draw_data2
  |                 +-- SortedDraw_data2
  +-- ProgProfile
  +-- RoutineProfile
  +-- CreateDrawData
  |     +-- CreateDrawDataPopup
  +-- TypeProfile
  +-- ExspoFormat
  +-- GlobalInfoData
  +-- PopupInformation
  +-- Xscale
        +-- Xaxis
Panel
  +-- GrPanel
  |     +-- GrLocation
  |           +-- GraphPanel
  |           |     +-- GraphPanel2
  |           +-- CaptionPanel
  |                 +-- CaptionPanel2
  +-- hScrollLabel
  +-- PopupInformationPanel

Container
  +-- ScrollView

BeanPanel
  +-- Aplmenu
  +-- ScaleP
  +-- Selectlb
  +-- DrawPrevNext

EventListener
  +-- MenuSelectListener
  +-- ScalingListener
  +-- LblStatChangedListener
  +-- PrevNextListener
```

KMviewクラスは、アプレットから継承したこのプログラムのメインクラスである。また、Objectクラスから継承したクラスはKMviewプログラムの基本的な処理をするクラス群である。BeanPanelから継承したクラスは、それぞれイベントを処理しKMviewクラスに渡す独立したクラスで、イベント情報を渡すためにEventListenerクラスから継承したインターフェースを実装している。以下にオブジェクトの基本機能を示す。

KMview アプレットクラスから継承したKMviewのメインクラス。イベントの処理や描画のコントロールなどを行う。

XYlocation 画面上の位置の情報を保存するクラス。描画される文字及び、グラフの位置情報を持っている。

myRectangle 矩形情報を保存するクラス。文字やグラフが描画される矩形範囲を決定する。アブストラクトクラスである。

Draw_rect 矩形情報に加え、グラフィックオブジェクト、色及び、クリッピング情報を持っている。

Draw_string 描画できる文字列のクラス。文字列のディメンション（幅と高さ）は、文字列のフォントのメトリックから定義することができるが、描画する矩形とは独立に文字のディメンションを保存している。

Bar_graph 棒グラフのグラフィックオブジェクト。矩形情報に加えボーダーの情報を保存している。

AnalysisData KMtoolで作成された計測データを保存する。計測データクラスをベクトルに保存する。

CreateAnalysisData 計測ファイルから実際に読込むクラス。読込むために、StreamTokenizerクラスを作成し入力を行っている。ファイルが見つからない時、入力エラーの時には例外処理を行う。

Draw_data 描画するグラフの情報を保存する。グラフのキャプション、グラフの値、タイトルなどの情報を持っている。

Draw_data1 描画するためのポップアップ情報を保存する。

Draw_data2 グラフの内訳情報を持つ。具体的には、計測量の通信時間である。通信時間がなければ、ゼロの値を持っている。

SortDraw_data2 描画するグラフのソートの順を定義する。

CreateDrawData グラフに必要なデータを、与えられた条件にしたがって計測データから抽出する。

CreateDrawDataPopup グラフのポップアップ情報を作成し描画データクラス "Draw_data1" に保存する。

GrPanel グラフ及び、文字列を描画するためのキャンバスの情報を定義するクラス。キャンバスの幅と高さ、色、枠の幅などを決めている。

GrLocation キャンバス上のグラフ及び、文字列の位置情報を計算する。グラフの間隔も保存している。

GraphPanel 棒グラフを描画するための情報と実際に描画するためのメソッドを持つ。また、マウスの移動イベントで詳細情報を表示する機能を実装している。

GraphPanel2 グラフの内訳情報（通信時間）を棒グラフに追加描画する機能を持つ。

CaptionPanel グラフのキャプション情報を保存し、実際に描画するためのメソッドを持つ。キャプションの文字列、色、フォント情報を保存している。

CaptionPanel2 キャプションが描画矩形をはみ出している場合、すなわち文字列がクリッピングされて途切れている場合に、マウスの移動により文字列を描き直す機能を実装する。

hScrollLabel ラベルの文字が表示領域幅を超えている時に横スクロールを可能にし、ラベル文字の全体が表示できるようにしたクラス。キャプションの表示の時に使用している。

PopupInformationPanel ポップアップ情報を表示するクラス。"PopupInformation" に保存されたデータを画面に表示する。

ScrollView グラフのキャプション情報と棒グラフをスクロール制御する。スクロールバー及び、キャプションパネル、グラフパネルをもっており、スクロール制御メソッドを実装する。

ProgProfile 計測データのプログラムプロファイルの情報を保存するクラス。

RoutineProfile 計測データのルーチンプロファイルの情報を保存するクラス。

TypeProfile 計測データのルーチン内のデータ情報を保存するクラス。

ExspoFormat 数値データをフォートランのE形式あるいは、F形式に変換する。F形式の場合に指定精度以下の数値は、E形式に変換される。

GlobalInfoData 解析データの中の、PE毎の実行時間、通信時間などの情報を保存する。

PopupInformation ポップアップ情報を保存するためのクラス。

Xscale スケーリングの情報を保存するクラス。データ列の最大最小値を計算する。

Xaxis スケーリングの情報とグラフ軸情報（位置、長さ、タイトルなど）を基に実際にX軸を描画する。

Aplmenu メニューのイベント処理を実行する。マウスによるイベントを受け取り、Kmviewに必要なデータに変換し、MenuSelectListenerインターフェースを作成しKMviewアップレットにイベントを渡す。

ScaleP スケール方法を選択するイベント情報を処理する。マニュアルスケールの時には、最大値、最小値の入力を促し、最大最小値を取得する。ScalingListenerインタフェースを作成し、KMviewアプレットにイベントを渡す。

Selectlb 経過時間、実時間及び、実行回数を選択する。選択情報は、数値に変換し、LblStatChangedListenerインターフェースでKMviewアプレットに通知する。

DrawPrevNext 描画データのヒストリの保存とヒストリへのアクセスメソッドを実装する。"<<":前のヒストリ及び、">>":次のヒストリボタンにより、過去に描画された任意のグラフを再描画することが可能である。イベント情報は、描画データに変換されPrevNextListenerによってKMviewアプレットに通知する。

EventListenerインターフェースを継承した四つのインターフェースは、ビーンパネルから作成したクラスに実装されイベント情報をKMviewアプレットに通知する役割を果たしている。これらのインターフェースの実装は、JBuilder2によって自動化されており、追加修正は容易に可能である。

This is a blank page.

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，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV=1.60218×10^{-19}J

1 u=1.66054×10^{-27}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

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·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	℃	
光束	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4　SIと共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å=0.1nm=10^{-10}m

1 b=100fm^2=10^{-28}m^2

1 bar=0.1MPa=10^5Pa

1 Gal=1cm/s^2=10^{-2}m/s^2

1 Ci=3.7×10^{10}Bq

1 R=2.58×10^{-4}C/kg

1 rad=1cGy=10^{-2}Gy

1 rem=1cSv=10^{-2}Sv

（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，1 eVおよび1 u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(=10^5dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2)=10 P(ポアズ)(g/(cm·s))

動粘度　1m^2/s=10^4St(ストークス)(cm^2/s)

圧力	MPa(=10bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	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	0.101972	2.77778×10^{-7}	0.238889	9.47813×10^{-4}	0.737562	6.24150×10^{18}
	9.80665	1	2.72407×10^{-6}	2.34270	9.29487×10^{-3}	7.23301	6.12082×10^{19}
	3.6×10^6	3.67098×10^5	1	8.59999×10^5	3412.13	2.65522×10^6	2.24694×10^{25}
	4.18605	0.426858	1.16279×10^{-6}	1	3.96759×10^{-3}	3.08747	2.61272×10^{19}
	1055.06	107.586	2.93072×10^{-4}	252.042	1	778.172	6.58515×10^{21}
	1.35582	0.138255	3.76616×10^{-7}	0.323890	1.28506×10^{-3}	1	8.46233×10^{18}
	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

1 cal= 4.18605J（計量法）

= 4.184J（熱化学）

= 4.1855J（15℃）

= 4.1868J（国際蒸気表）

仕事率　1 PS(仏馬力)

= 75 kgf·m/s

= 735.499W

放射能	Bq	Ci
	1	2.70270×10^{-11}
	3.7×10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	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）

並列／非並列共用プログラム性能解析ツールの開発