JAERI-M
93-128

並列計算機を利用した遮蔽安全評価用モンテカルロコードMCACEの高速化

(3) 共有メモリ型並列計算機による並列処理および複数ワークステーションを使用した並列処理

１９９３年７月

高野　誠・小野寺えみ[*1]・増川　史洋
内藤　俶孝・今若　恒幸[*2]・依田　佳久[*3]

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

JAERI-M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，日本原子力研究所技術情報部情報資料課（〒319-11茨城県那珂郡東海村）あて，お申しこしください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 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。

JAERI-M reports are issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　いばらき印刷㈱

並列計算機を利用した遮蔽安全評価用モンテカルロコードMCACEの高速化 (3)共有メモリ型並列計算機による並列処理および複数ワークステーションを使用した並列処理

日本原子力研究所東海研究所燃料サイクル安全工学部
高野　誠・小野寺えみ*1・増川　史洋
内藤　俶孝・今若　恒幸*2・依田　佳久*3

(1993年6月1日受理)

遮蔽安全評価用モンテカルロコードMCACEの並列化を，共有メモリ型のベクトル並列計算機Monte-4を使用した場合および複数ワークステーションを使用した場合の2種類の並列処理について検討した。Monte-4では，MCACEコードのコピーを4CPU上で実行し，並列処理後の結果を同一ファイルへ書き込む方法を用いた。複数ワークステーションによるネットワークパラレル処理では，ホスト・ノード型のモデルによる並列化について検討した後，先のMonte-4で使用したモデルを特別な並列処理用ソフトを使用せずFORTRAN言語のみで実現することを試みた。

検討の結果，Monte-4では4CPUを使用した時，1CPUで処理した時のほぼ3倍の高速化が達成された。複数ワークステーションの場合には，4台のワークステーションを用いて，スカラー大型計算機M-780以上の処理速度を達成可能であることがわかった。

東海研究所：〒319-11　茨城県那珂郡東海村白方字白根2-4
＊1　㈱アイ・エス・エル
＊2　新日本製鉄㈱
＊3　ニチメンデータシステム㈱

JAERI-M 93-128

Speed up of MCACE, a Monte Carlo Code for Evaluation of Shielding Safety, by Parallel Computer (3) Parallel Computing by Shared Memory Type Parallel Computer and by Networked Several Workstations

Makoto TAKANO, Emi ONODERA[*1], Fumihiro MASUKAWA
Yoshitaka NAITO, Tsuneyuki IMAWAKA[*2] and Yoshihisa YODA[*3]

Department of Fuel Cycle Safety Research
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received June 1, 1993)

The parallel computing of the MCACE code has been studied on two platforms; 1) Shared Memory Type Vector-Parallel Computer Monte-4 and 2) Networked Several Workstations. On the Monte-4, a disk-file has been allocated to collect all results computed by 4 CPUs in parallel, executing the copy of the MCACE code on each CPU. On the workstations under network environment, two parallel models have been evaluated; 1) a host-node model and 2) the model used on the Monte-4 where no software for parallelization has been employed but only standard FORTRAN language.

The measurement of computing times has showed that speed up of about 3 times has been achieved by using 4 CPUs of the Monte-4. Further, connecting 4 workstations by network, the computing speed by parallelization has achieved faster than our scalar main frame computer, FACOM M-780.

Keywords: Monte Carlo, MCACE, Parallel Computing, Speed Up, Monte-4, Workstation, Network, Shared Memory

---

*1 I.S.L. Inc.

*2 Nippon Steel

*3 Nichimen Data Systems Corporation

目　　次

Contents

1. 緒　　言

遮蔽安全評価用モンテカルロコードMCACE の並列処理による高速化について，これまでに並列計算機シミュレーターを使った検討，および実際の分散メモリ型並列計算機（AP－1000）を使った検討を行った。[(1)～(4)] これらの検討では，分散メモリ型の並列計算機を対象としたが，本報では共有メモリ型のベクトル・並列計算機 Monte－4 [(5) (6)] を使った並列処理および複数のワークステーションを使った並列処理の2種類の並列処理について検討した。

共有メモリ型並列計算機，分散メモリ型計算機および複数ワークステーションによる並列構成のそれぞれに対し，メモリ，CPU およびディスクの関係をFig. 1. 1 に模式的に示した。共有メモリ型並列計算機では，メモリだけでなくディスクも全CPU から直接参照可能であるが，分散メモリ型並列計算機では通信を行って他の CPUユニットのメモリ上データにアクセスすることになり，また，ディスクは通常，各 CPUユニットに接続されていないため，ディスクを有する CPUユニットへ，必要なデータを通信により送受信してやる必要がある。一方，ネットワーク接続された複数のワークステーションによる並列処理の場合には，各ワークステーションの CPUは独立したディスクを有するが，通常，NFS(Network File System)等により，各ワークステーションから同一のディスクをアクセス可能な構成としている場合が多い。したがって，この場合ディスクについては共有メモリ型並列計算機と類似しており，メモリについては分散メモリ型並列計算機と類似した構成となっていると考えられる。

本報では，上述の様な点を考慮に入れて並列化アルゴリズムを検討した。つまり，共有メモリ型並列計算機では，基本的にディスクの共有性を利用するものとし，MCACE コード[(8)] のコピーを各CPU で実行し，各々の中間結果を次々と共有のディスクへ書き込んでゆき，最終的に必要となるファイルを作成する方法を用いた(Copy Model)。その際，各CPU へのロードバランスが均一となるような，配慮をした。一方，複数ワークステーションによる並列処理では，上述の共有メモリ型計算機で使用したアルゴリズムおよび，文献(1)～(4)で示した分散メモリ型並列計算機で用いたアルゴリズム(Host-Node Model) の両方に対し検討した。

原研に導入された共有メモリ型並列計算機 Monte－4 での並列化では，MCACE コードを実際に並列化するために必要となったプログラムの変更や改良方法について具体的に示すとともに，他のプログラムでも発生し得る問題点と解決方法を示し，これから Monte－4 を使用しようという人の参考となるようにした。また，本報で示した実行時間の測定は，平成5年3月から4月にかけて行ったものであり，今後さらにオペレーティングシステムや FORTRANコンパイラ等のバージョンアップにより，処理効率の改善が予想される。ここで，本報では，MCACE コードの並列化についてのみ検討しており，ベクトルについては取扱っておらず，今後の検討課題とした。

また，複数ワークステーションを用いた並列化では，分散メモリ型並列計算機で用いたアルゴリズムによる並列処理を行うため，並列処理用ソフトウェア「パラレルウェア」を使用した。一方，共有メモリ型並列計算機で用いたアルゴリズムによる並列処理を，特別なソフトウェアを使用せず，FORTRAN言語だけを使って実現する事を試みた。

ここで，本報で使用した用語についての説明を以下に示す。

① 「並列化効率」

並列処理を行った時の効率を0～100％で示すもので，10台のCPUを使って並列処理した時に1台のCPU の場合の1／10の処理時間で終了したとすれば，「並列化効率」は 100％であり，1／5の場合であれば50％となる。

つまり，

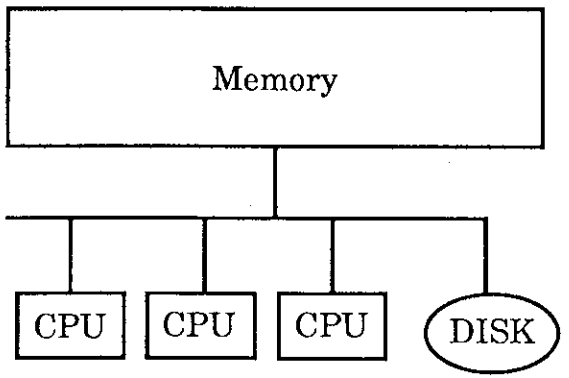
$$(\text{並列化効率}) = \frac{1\text{台のCPUによる処理時間}}{(\ n\text{台のCPUによる並列処理時間})\times n} \times 100(\%)$$

として示される値である。

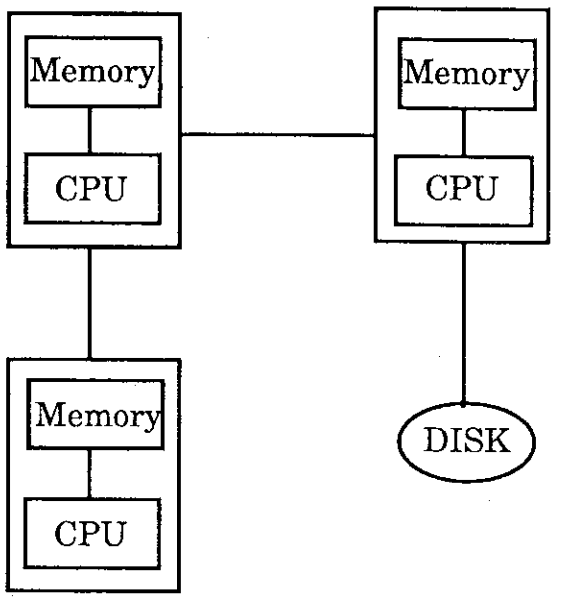
② 「並列処理時間」

スカラー計算機での処理時間は，CPU 時間を用いて容易に行えるが，並列計算機の場合には通信や同期のためにCPUが常に100％作動している訳ではないため，CPU 時間そのものを並列処理時間とすることはできない。つまり，並列化処理では，CPU の待機状態すなわちアイドル時間も含めたものが全体の処理時間に相当する。したがって，本報では並列計算機による処理時間（「並列処理時間」）として，ジョブ実行開始から終了までに経過した実際の時間（Elapse Time）を用いるものとした。したがって，並列処理時間の正確な測定を行うには，他のジョブが実行されていない状態である必要がある。

A. Shared Memory Type (Monte-4)



B. Distributed Memory Type (AP -1000)



C. Network Parallel
(Multiple Workstations)

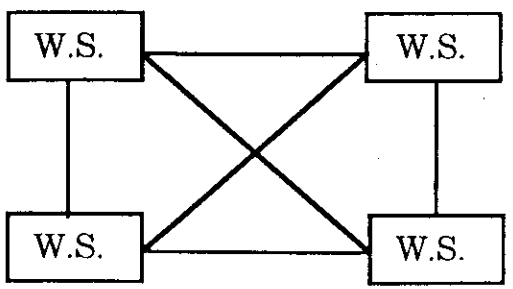


Fig. 1.1 Schematics of Three Types of Parallel Computers

2. 共有メモリ型並列計算機Monte－4による並列処理

共有メモリ型並列計算機 Monte－4 [5] [6] は，4台のベクトルプロセッサを使ったベクトル・並列計算機である。本章では，MCACEコードをMonte－4で実行するための並列化アルゴリズム，プログラムの書き換え，および並列化効率の測定結果について示す。これは，これから Monte－4で並列処理を試みるユーザーの参考となろう。

2.1 並列化方針

Fig. 2. 1にMCACE コードのプログラムフローを示した。図に示した様に，先ず入力データおよびデータプール形式の断面積データを読み込み，解析形状や各領域の物質指定等を行い，問題のセットアップを行う。次に,粒子のランダムウォークシミュレーションを，粒子毎に実行する。ここで，MCACE コードでは，数百～数千程度の粒子を単位とするバッチという概念を使用している。モンテカルロ遮蔽計算では，先に計算されたランダムウォークシミューレーションによる結果を次の粒子の計算に反映させる必要がなく，基本的に個々の粒子のランダムウォークシミューレーションは独立して行える。この点は，前回のバッチ（サイクル）で得られたソース分布を，次のバッチで使用するモンテカルロ臨界計算と大きく異なる点である。つまり，モンテカルロ臨界計算ではバッチ（サイクル）毎に同期を取り，新たなソース分布を求めてから次のバッチの計算を行うため，バッチ単位の遂次処理となっており，高い並列化効率が得にくくなっている [4] 。したがって，MCACE コードで採用しているバッチループは便宜的なものと言え，このループはバッチ毎の中間結果を得るためのみに使用されている。ここで，バッチ毎の中間結果は，作業用ファイルに次々と書き込まれて行き，全バッチ終了後，最終結果を得るために，このファイル内容が読み込まれる。

この様なプログラムフローとなっているため，バッチループを並列化すれば良い事がわかる。各プロセッサからのファイルアクセスは，Monte－4ではスカラー機と同様に利用可能であり，入力データ等の読み込みおよび処理も各プロセッサで独立に行える。一方，分散メモリ型のAP－1000では，ホストが入力データ等の読み込みと処理を行い，その処理結果を各セルへ放送してデータを送信する必要があり，ホストが入力データ処理を行っている時間，全セルはアイドル状態となる。さらに，データ転送という並列処理のためのオーバーヘッドが生じることとなる。このデータ転送に起因するオーバーヘッドを最小化して並列化効率を上げるためには，コモンデータの中から実際にホスト・ノード間の通信を必要とするデータを選び出す必要がある。転送すべきデータ量は，通常問題依存であり，必要最小限のデータを転送するのは，かなりやっかいな仕事となる。この点，Monte－4のように，各プロセッサからファイルを自由にアクセスできるのは，並列化のために行わねばならない作業量を軽減する点から非常に好ましいと言える。同様に，バッチループ終了後の，最終結果処理も，各プロセッサで行うことができる。

したがって，Monte－4での並列化方針は次の様になる。

① 並列化はバッチループに対して行う。

② 入力データ処理は，各プロセッサで同様の処理を同時並行的に行う（重複処理となるが，並列処理時間を増大させる性格のものではない）。

③ 最終結果の処理も，②と同様に各プロセッサで同時並行的に行う。

ここで，分散メモリ型並列計算機で配慮する必要のある，メモリ競合については，次節に示す様に全CPU が読み書きすべきデータの数を1つだけとすることができるため，大きな問題とはなっていない。

2.2 並列化プログラミング

2.2.1 アルゴリズム

前節の並列化方針に基づき，MCACE コードの並列化プログラミングについて示す。主要部分である，バッチループを並列化する方法として，Fig. 2.2に示す様に，ホスト・ノード型を用いる事も考えられるが，バッチの割り当てを制御するためのルーチンが必要となったり，オリジナル MCACE コードからのプログラムの切り出し等が必要となり，多少複雑な作業となりうるため，次に示す単純な方法を用いた。

実際に用いた方法は，バッチの割り当てを陽に制御するものではなく，全CPUにシリアル版MCACE コードをロードして独立に実行させ，各CPUでは処理したバッチ数を共有メモリ上に他のCPUから参照できる型で示しておくだけである。この時，共有メモリは各CPU に対して掲示板の様な機能を有することになる。したがって，Fig. 2.3に示す様に，あるCPU は，この掲示板に示された既に他のCPU が処理したバッチ数と，入力データで与えられた総バッチ数を比較して，次のバッチ計算を行うか否かを判断すれば良い。したがって，MCACE コードのコピーを各CPU で実行すれば良いことになる(Copy Model)。

この様な方法が，実際にMonte－4で可能かどうかを，Table 2.1に示すサンプルプログラムで確認することとした。ここで，メインプログラムから呼ばれるサブルーチンsubが，MCACEコード全体に対応するものである。また，サブルーチンsub の9行目（同表中では29行目）が，Fig. 2.3に示した掲示板を見ることに相当し，13行目（同表中では33行目）が単位バッチ処理結果を作業用ファイルへ書き込む事に相当する。また，同表中27行目，30行目および35行目のpllockやplunlockは掲示板を見たり，書き込んだりする作業や，ファイルへの書き込みを，他のCPU に対し排他的に行えるようにするためのものである。このサンプルプログラムは問題なく実行されたため，MCACE コードの並列化作業に着手した。

2.2.2 並列化作業

MCACE コードのコピーを1つのメインプログラムの下で複数並列実行させるためには，ptforkサブルーチンにより MCACEコードを子プロセスとして生成し実行する。しかしながら，4つの子プロ

セスを同時に走らせるため，オリジナル MCACEコードのコモン文を処理しないと共有メモリ上では独立した実行結果が得られない。そのために， Monte－４では，local common文が用意されており，これを使えば見かけ上１種類の子プロセス用 MCACEコードでも，ptforkにより生成した複数の子プロセスはそれぞれ独立したメモリ領域を持てる様になる。このlocal commonの機能がないと，コモンブロック名およびサブルーチン名にすべて重複のない，コピープログラムを４つも作成しなければならぬ事となる。

実際の作業手順を以下にまとめて示す。また，主要ルーチンであるサブルーチンMORSE の変更後プログラムフローをFig. 2. 4 に示す。

(1) プログラム全体の変更

- 名前付COMMONブロックを総てLOCAL COMMONブロックに変更
- 無名COMMONブロックを名前付LOCAL COMMON／emi／ブロックに変更
- サブルーチン中のデータ文は総て無名共通ブロックに割合てられてしまうので，各サブルーチン中のデータ文で定義される変数は，他と重複しない名称を有するGLOBAL COMMON ブロックで再定義

(2) プログラム単位での変更

サブルーチン名	DTLIST
変更箇所	NPID（引数追加） 入力データファイル番号を19から50＋NPIDに変更 出力データファイル番号を６から60＋NPIDに変更
変更理由	それぞれの子プロセスで入出力処理をさせる為

サブルーチン名	ELM000（コピーしたMCACEコードのエントリルーチン）
変更箇所	NPID（引数追加） NC(1)をNC(500000)に変更（サイズは，ここで定義される） 入力データファイル番号を19から50＋NPIDに変更 出力データファイル番号を６から60＋NPIDに変更 GLOBAL COMMON／DT009／NOUTGM
変更理由	

サブルーチン名	INPUT
変更箇所	MTFLAG, NPID（引数追加） GLOBAL COMMOM／DT013／JUNK, IHOLB STOP文を“MTFLAG＝１”にしてRETURN文に変更
変更理由	STOP文は子プロセス中では実行できないため

サブルーチン名	JOMIN
変更箇所	NPID（引数追加） 出力データファイル番号を16から20＋NPIDに変更
変更理由	

サブルーチン名	MAIN
変更箇所	新規作成
変更理由	

サブルーチン名	MORSE
変更箇所	NPID（引数追加） GLOBAL COMMON／LOCKDT／LOCKVA GLOBAL COMMON／ITCONT／ITCN(4) サブルーチンINPUT から戻ってきたとき“MTFLAG＝1”ならばRETURN文にそれ以外なら処理を続ける ４つのバッチに対し１つづつロックして値を書き出す。
変更理由	正常終了させる為RETURN文に変更

サブルーチン名	POPEN
変更箇所	４つのデータから現在読み込まれているデータ値と合ったデータファイルを選びオープンする
変更理由	

(3) 並列化に伴うバグの対応

これまでに示した(1), (2)の変更で並列化MCACEはMonte－４上で動作するはずであるが，実行時にエラーを起してアボートしてしまう。調査の結果，これは Monte－４の並列化ではSAVE文の使用が不可能であるという事が原因であった。しかしながら,MCACEコード自体には，SAVE文はどこにも使用されていない。この一見矛盾した現象は，特殊な状況でのみ発生し得るのではなく，むしろ大部分のコードで発生し得るものである。したがって，これから Monte－４で並列化を試みるにあたって，十分注意する必要があろう。　この現象は，並列化部分に含まれるサブルーチンがエントリー文を有している場合に発生する可能性がある。つまり，以下の様な ENTRYは通常問題なく動作する。

```
s u b r o u t i n e   a a （ x n ）
a ＝ x n
e n t r y   b b （ x n ）
x n ＝ a ＋ x n
r e t u r n
e n d
```

（例１）

この時エントリー文の次に現れる変数ａは，その値がどこかにSAVEされていることになる。したがって，このプログラムは，以下のSAVE文を含むものと同様である。

```
s u b r o u t i n e   a a （ x n ）
s a v e   a
a ＝ x n
e n t r y   b b （ x n ）
x n ＝ a ＋ x n
r e t u r n
e n d
```

（例２）

つまり，（例１）のプログラムは，暗黙の内にSAVE文を使用していることに相当し，実際は(例２)の様に書くべき所だが，通常はSAVE文を入れなくとも支障なく動作する。一方,Monte－４ではSAVE文を並列化部分には使えない事になっており，(例２)はFORTRAN コンパイル時にエラーとなるが，（例１）ではコンパイル時ではなく実行時に変数ａの値には「ゴミ」が入り，エラーを発生することになる。対処法は，以下の（例３）の様にコモン変数として変数ａを定義してやれば良い。

```
s u b r o u t i n e   a a （ x n ）
l o c a l   c o m m o n   ／ a a c o m ／ a
a ＝ x n
e n t r y   b b （ x n ）
x n ＝ a ＋ x n
r e t u r n
e n d
```

（例３）

MCACEコードでは，サブルーチン GETNTとBANKR に，上述の該当部分が存在しないため，（例３）と同様な対処をした。

2.3　並列化効率の測定

Monte－4の運用形態は，複数ユーザーが同時にJOBを実行可能なものとなっている。したがって，１台のCPUのみを使用するシリアルジョブの場合には，unixのtimeコマンドやMonte－４ Fortran環境設定(F_PROGINF)により得られる積算 CPU時間を処理時間と考えることができるが，並列ジョブ

の場合には全4台のCPU をすべて専有した状態で測定を行う必要がある。これは，並列処理時間には各CPU間の同期待ちによるアイドル時間をも含める必要があるため，CPU時間ではなくジョブ開始から終了までの実時間で並列処理時間を評価するのが適当と考えられるからである。したがって，並列プログラム実行中に他のジョブが入り込む可能性のあるマルチユーザー環境では，実時間による測定値に他のジョブが実行による時間も含まれてしまうため，正確な並列処理時間を得ることが困難なものとなる。

このため，次に示す測定値は情報システムセンタに依頼して測定してもらったものであり，この時Monte－4はシングルユーザーモードで運用された。測定結果をTable 2.2に示す。この表から，CPU4台の時に，1台の時の2.88倍となっており，72%の並列化効率となっていることがわかる。

次に，MCACE コードが解析する問題による処理時間の変化について測定した。ここでは，①20粒子／バッチ，全8192バッチ，②80粒子／バッチ，全2048バッチ，③ 160粒子バッチ，全1024バッチおよび④1280粒子バッチ，全128 バッチの4種類の問題を実行して比較した。これら3題は，処理すべき全体の粒子数は 163,840個で同一であるが，排他処理をしてファイルへ書き込む操作が，それぞれ8192回，2048回，1024回および 128回であり，大きく差がある。これら4題をAP－1000で実行したとすれば，バッチ数に通信処理回数がほぼ比例するためた，並列処理時間は，④，③，②，①の順で大きく増大し，並列化効率もこの順番で悪化してゆくと予想される。シングルユーザーモードでの測定結果を，Table 2.3に示す。この表から，並列化処理時間（Elapse Time)は，バッチ数の変化に対して殆ど影響を受けていないことがわかる。4台のCPU時間の合計を見れば，バッチ数の増加に比例して，やや合計CPU時間が増大するのがわかるが，1.5%程度の増加であり，あまり有意なものではない。したがって，共有メモリ型の並列計算機では，データ通信に起因する並列処理のオーバーヘッドが非常に小さいものであると言える。つまり，共有メモリ型並列計算機では，各CPU 間のデータ通信を共有メモリ上の必要データにアクセスすることにより実現するため，分散メモリ型の場合より高速なデータ通信が可能になっていると言える。

Table 2.1 Sample Program for MCACE Parallelization on Monte-4

```
      external sub
      global common /lockdt/lockva
      call ptfork(tid1,tp1,sub,1)
      call ptfork(tid2,tp2,sub,2)
      call ptfork(tid3,tp3,sub,3)
      call ptfork(tid4,tp4,sub,4)
      call ptjoin(tid1)
      call ptjoin(tid2)
      call ptjoin(tid3)
      call ptjoin(tid4)
      rewind 7
      do 10 i=1,300
      read(7,700,end=900) idata
  700 format(2i5)
      write(6,700) i,idata
   10 continue
  900 stop
      end


      subroutine sub(id)
      local common /com1/ int(100)
      global common /g1/ ino(4)
      global common /lockdt/ lockva
      call subsub(id)
      do 10 i=1,100
      call pllock(lockva)
      ino(id)=i
      if(ino(1)+ino(2)+ino(3)+ino(4).gt.200) then
      call plunlock(lockva)
      return
      else
      write(7,700) int(i)
  700 format(i5)
      call plunlock(lockva)
   10 continue
      return
      end

      subroutine subsub(id)
      local common /com1/ int(100)
      do 10 i=1,100
      int(i)=id
   10 continue
      return
      end
```

Table 2.2 Result of Parallel Efficiency Measurement on Monte-4
(Single User Mode)

Run No.	No. of CPUs	Optimization*1	Measured Elapse Time (sec.)	Speedup Ratio (times)		Parallel Efficiency (%)
1	1	No	11209 *2	1.00	-	-
2	1	Yes	7457 *2	1.50	1.00	-
3	4	No	3898 *2	2.88	1.91	72 *4
4	4	Yes	(2589) *3	(4.33)	(2.88)	(72) *5

(Problem Size ... 1280 particles/batch, Total 512 batches)

*1 Fortran Compiler Option
*2 Measured by Information Systems Center
*3 Estimated by using 2.88 of speedup ratio
*4 Efficiency of Run No.3 to Run No.1
*5 Efficiency of Run No.4 to Run No.2

Table 2.3 Effects of Batches on Parallel Efficiency on Monte-4
(Single User Mode)

Run No.	Problem Size		Elapse Time (sec.)	CPU Time (sec.)				
	Particles/Batch	Total Batches		No.1	No.2	No.3	No.4	Total
1	20	8192	1057	1028	996	974	887	3885
2	80	2048	981	979	971	967	927	3844
3	160	1024	974	973	969	966	931	3839
4	1280	128	981	981	957	956	934	3828

(Fortran Compiler Option ... No Optimization, No Vectorization)
(Measured by Information Systems Center)

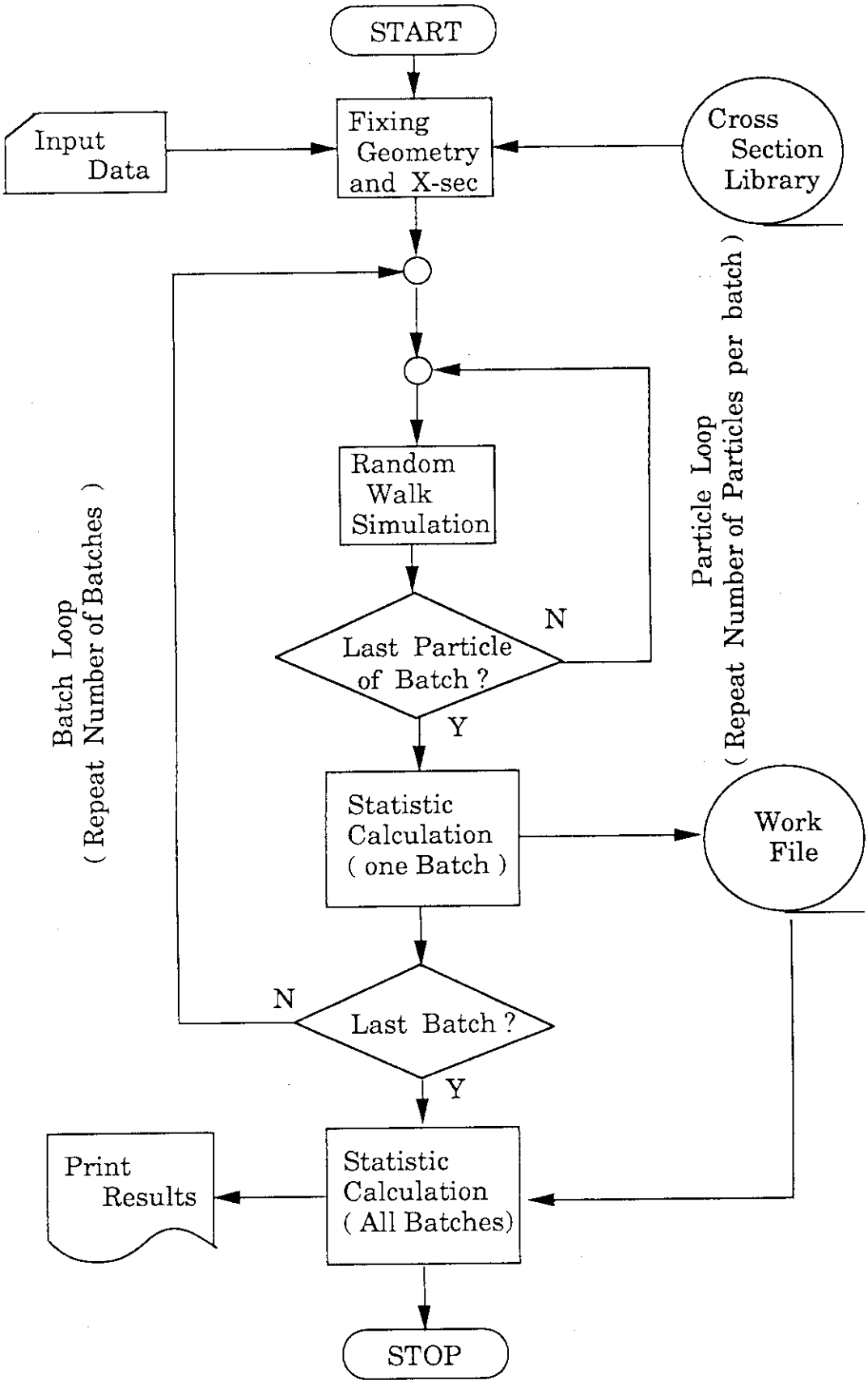


Fig. 2.1 Program Flow of MCACE

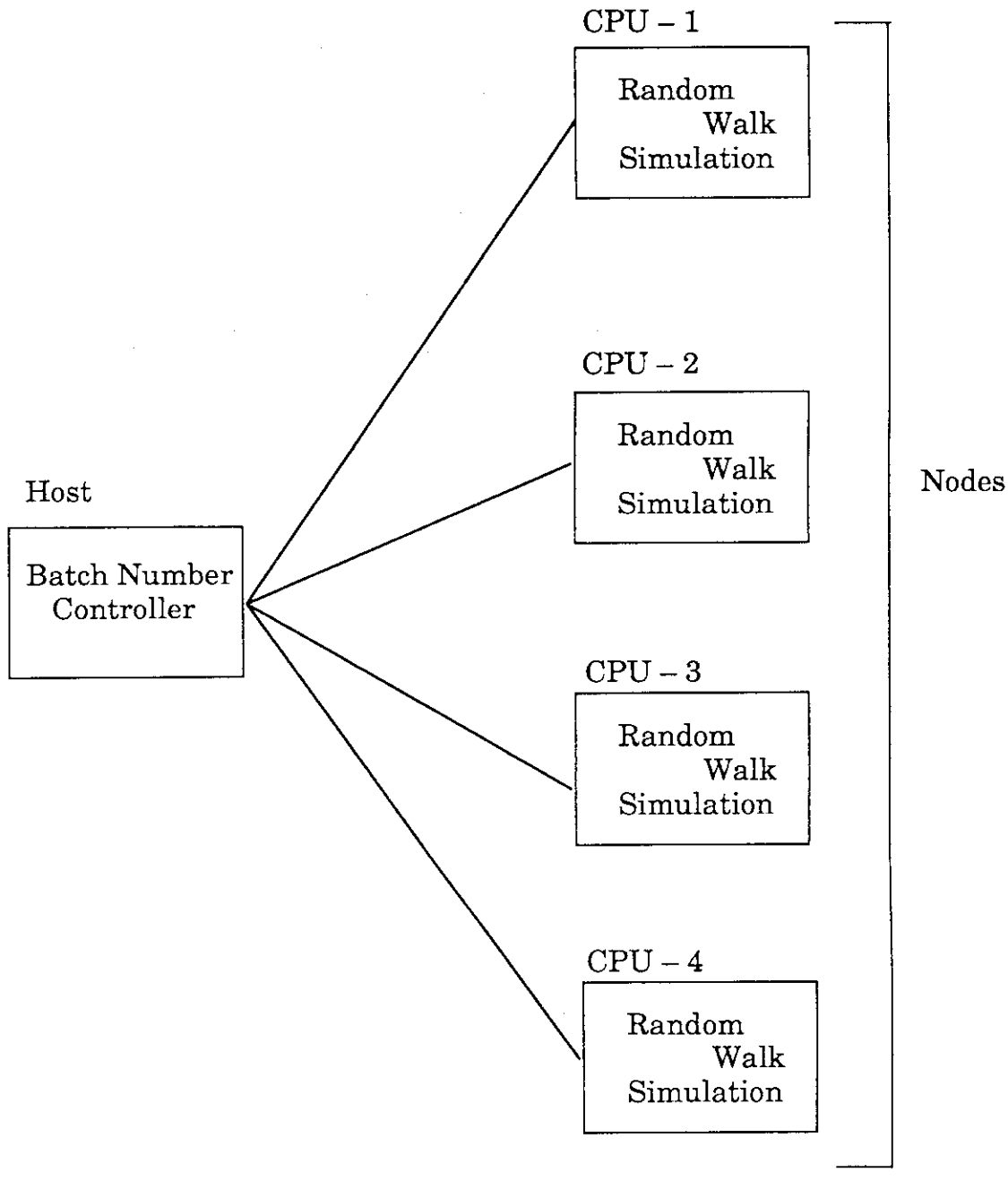


Fig. 2.2 Parallelization of Batch Loop(Host-Node Model)

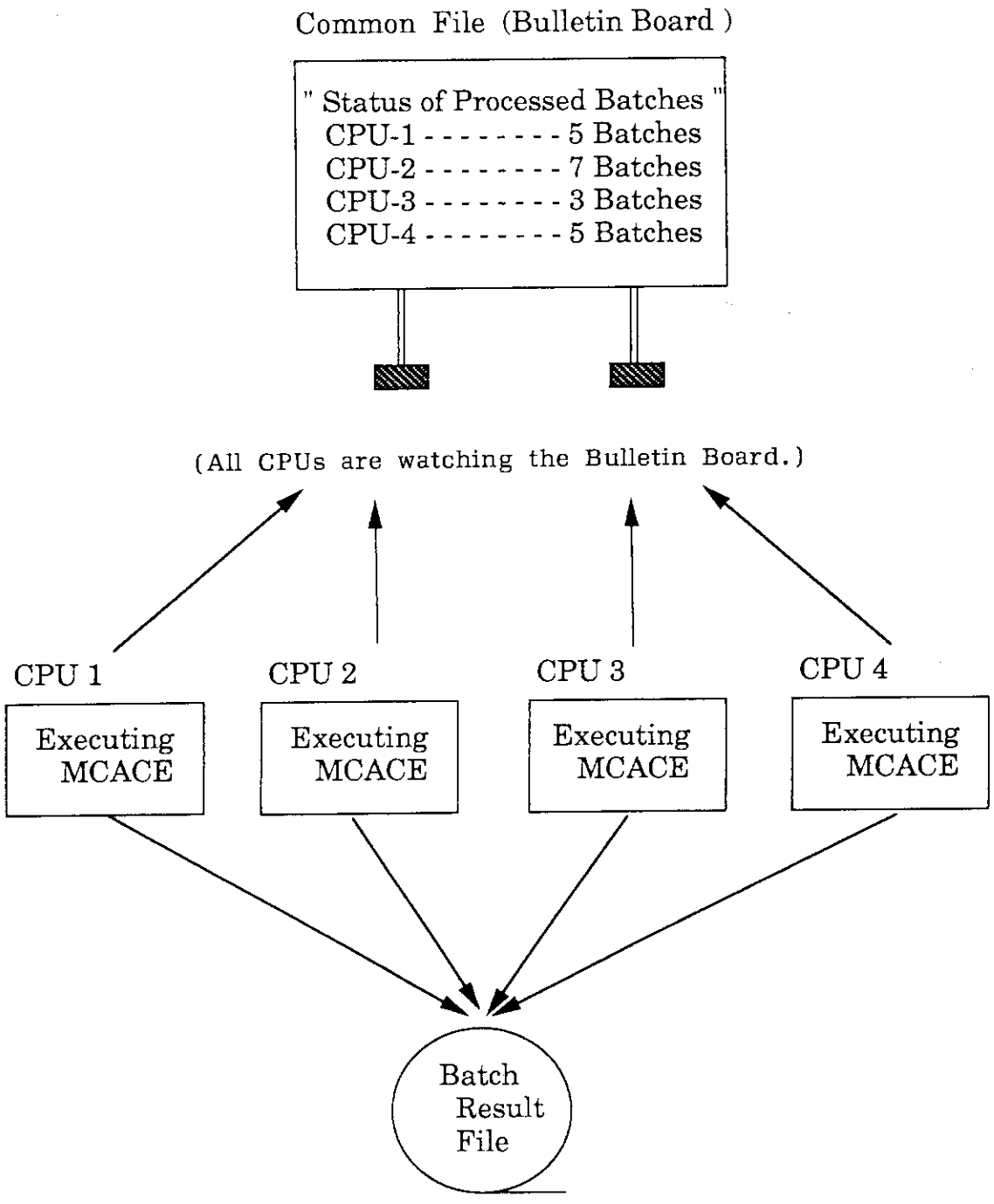


Fig. 2.3 Employed Parallelization Scheme

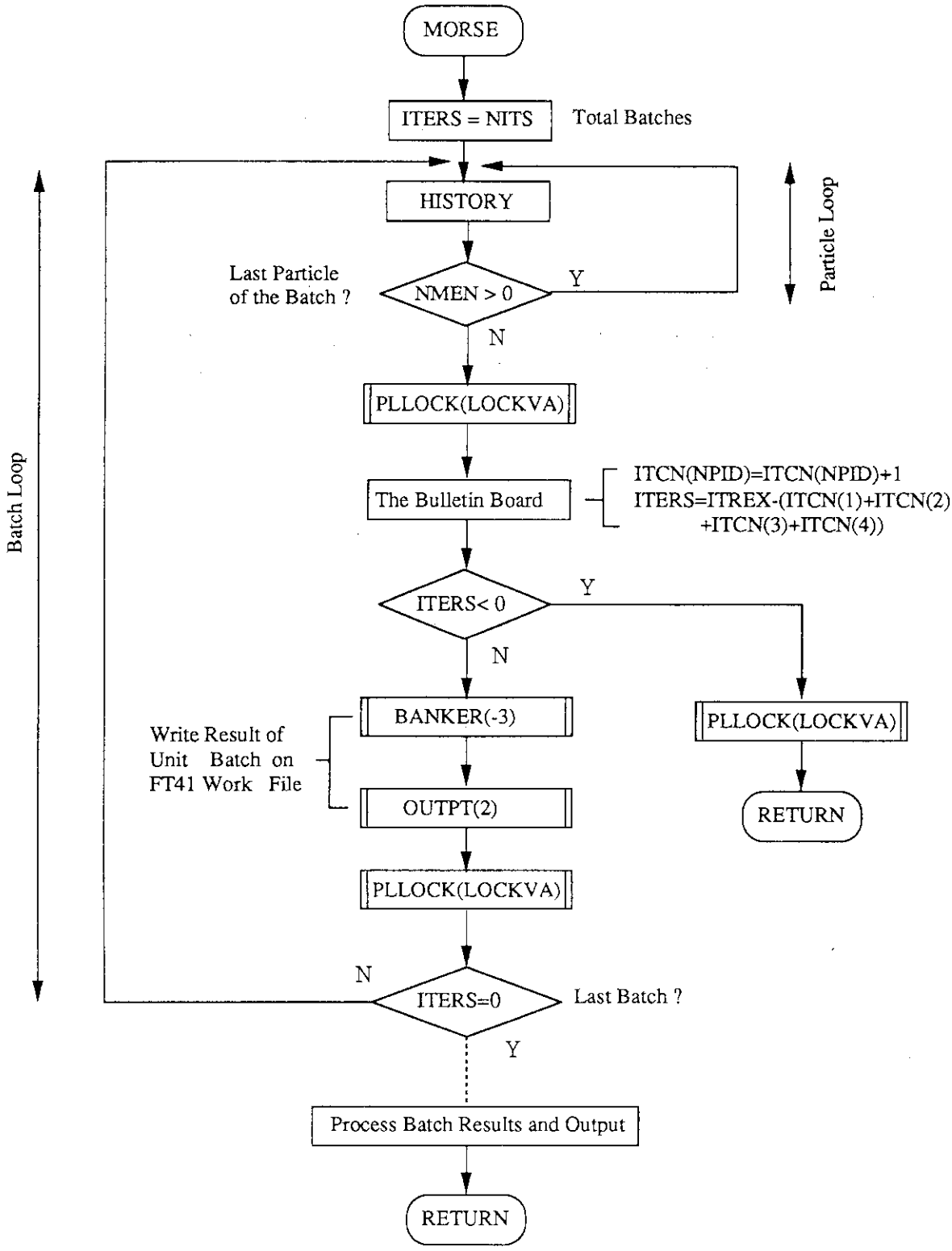


Fig. 2.4 Modified Subroutine MORSE for Parallel Execution on Monte-4

3. 複数ワークステーションによる並列処理

最近のワークステーションの処理速度向上は目覚ましく，数年前のスカラー大型計算機と同等以上の性能を有するものも出現している。さらに通常，ワークステーションは，イーサネット等によりネットワークに接続され，電子メールの交換やNFS(Network File System)によるディスクの共有が可能となっている。したがって，ネットワークを介して各ワークステーション相互間でデータ通信を行いながら，それぞれのワークステーション上で JOBを実行することが基本的に可能である。つまり，ネットワークを介した複数のワークステーションにより並列処理が可能であり，これは，分散メモリ型の並列計算機と同様の構成になっていると言える。しかしながら，この様なハードウェア環境のみでは効率的なネットワークパラレル処理を行うことが出来ず，データ通信や同期を行うための基本的なソフトが必要となる。このソフトは，並列処理に伴い必要となる各ワークステーション上のプログラム相互間のデータ通信を可能とし，さらにプログラムの流れを制御するための同期を可能とする様なものである。

ネットワークパラレルを実現するソフトは種々存在するし，これから新たなものも開発されてゆくと思うが，本報では,米国パラソフト社の「Network Parallel Ware 以下,パラレルウェアと称す」を使用するものとした。この他にも，「Linda」や「PVM」というソフトも存在する。これらのソフトについては，現在の所あまり情報がなく不明な点が多い。ただし，「PVM」については Pubic Domain（無償ソフト）にあり，今後，入手してみようと考えている。これは，無償ソフトであるため，今後の展開によってはネットワークパラレルの標準ソフトとになる可能性があるためである。（本稿執筆中に「PVM-3，を米国ORNLよりネットワークを介して入手することができた。「PVM-3」については後日，改めて報告したい。）

3.1 ワークステーションパラレル用ソフト

本報で使用したソフト「パラレルウェア」は，LAN 上の複数のワークステーション(WS)等（大型機も可）を並列実行単位（ノード）として使用し，並列処理を可能とするものである。パラレルウェアでは，LAN 上の複数のWSにFig. 3.1 に示す様なホストとノードの関係を持たせ，各ノードに並列化したプログラムをロードし並列処理を行う。ホストプログラムおよびノードプログラムは，ホストWSから各ノードWSへネットワークを介してロードされるため，各ノードWSのディスクにノードプログラムを持つ必要はない。したがって，ホストWSは，任意のノードWSへ任意のプログラムをロードして実行させることが可能である。パラレルウェアでは，並列処理に必要となる初期化，通信および同期を行うためのルーチンをライブラリとして有し,FORTRANのサブルーチンとして呼び出すことができる。

本報の環境では，ホストWSとノードWS相互間の通信，同期は，Fig. 3.2 に示す様にイーサネット

を介して行われている。また，パラレルウェアを使用するためには，使用する全WS上に，ホストWSからアクセス（rlogin）可能なユーザーアカウントが必要であり，さらに各WS上のユーザーは，他のWSに自由にrlogin可能な環境となっている必要がある。例えば，ユーザー名「emi」で使用可能なWS上にアカウントが有すとすれば，各WS上の/etc/hostsファイル中に自分以外のWSのホスト名およびIPアドレスを定義し，さらに各WS中のユーザー「emi」のホームディレクトリ中ファイル・rhostsへ自分以外の全WSホスト名を登録する必要がある。

3.2　並列化方針

並列化は，以前にAP-1000で行った並列化方針に基づくものとした[1]。これは，分散メモリ型並列計算機である AP-1000とネットワークパラレル処理の環境が非常に類似しており，容易に並列化を実現できると考えられるからである。つまり，分散メモリ型の並列計算機を想定した場合，「ホスト・ノード」型モデル用の並列用ライブラリは，並列計算機の機能を問わず，必然的に同様な機能を有するものとならざるを得ないからであり，基本的に必要な機能としては，ホストとノード（セル）間のデータ通信用機能および，データ通信を確実に行うために必要となる同期を実現するための機能の２種類となる。ただし，AP-1000ではハードウェアにより同期を実現しているが，「パラレルウェア」ではソフトにより実現している点が異なっている。

MCACEコードをホスト・ノード型に並列化するためには，サブルーチンMORSEをホスト側とノード側に分離する。ノード側のプログラム部分としては，粒子のランダムウォークを計算する部分が適当であり，バッチのループを並列化した。この時，ホストとノード間で必要なデータを並列用ライブラリを使って通信することになる。ホストとノードに分割されたプログラムは，それぞれFig. 3. 3に示す様な機能を有している。並列化したMCACEコードの実行環境や実行方法については，付録Ａ２に示した。

3.3　並列化効率の測定

ホスト・ノード型モデルで並列化した MCACEコードを，実際に複数台のWSで実行し処理速度の向上についての測定を行った。処理時間の測定は，MCACEコードのノード部プログラムを各WSへロードするためのロード時間，形状・断面積データをホストから各ノードへ送信するためのデータ転送時間，およびモンテカルロ計算を処理するためのバッチ処理時間とに分けて測定した。これらの時間測定は，プログラム内に組み込んだタイマールーチン(etime)により得られる経過時間（実時間)から求めた。ただし，後述の3. 3. 2節ではunixのtimeコマンドで得られる経過時間を測定値とした。また，測定時には，他のユーザーがWSにログインしない様配慮した。

3.3.1　８台のSUNワークステーションを使用した場合（Case１）

並列化効率の測定に使用したWSはSUNのSparc-2が５台，4/370，4/490およびSparc1＋が各１台の

計8台であり，処理速度の異なるWSが混在している。また，測定環境としては，一般的な運用状態で行い特別に測定環境を設定した訳ではない。したがって，測定中に各WS及びネットワークの負荷状態は一定ではない。また，WSはイーサネットによりネットワーク化されており，このネットワークには，測定に使用した8台のWS以外にも，約80台のWSが接続されている。

測定環境および測定結果をTable 3.1に示す。この表では，8台のWSそれぞれの処理速度の指標としてMIPS値を示した。さらにSparc-2のMIPS値を1.0としてWS台数の増加によるMIPS値の相対的増分をカッコ内に示した。これは，並列化が理想的であり 100%の効率である場合の処理速度倍率に対応するものと考えられる。また，ロード時間，データ転送時間およびバッチ処理時間の合計値より全処理時間が大きくなるのは，入力データ処理や計算結果処理に要する時間が加わるからである。ここで，ロード時間とデータ転送時間は，ホスト・ノード型並列化処理モデル特有のものと言えよう。得られた，全体処理時間およびバッチ処理時間に対する，WS台数の増加による処理速度の増加を，1台のWSでの値をベースにして示した（Fig. 3.5参照）。処理速度倍率は，全体処理時間に対して8台のとき，66%の効率となっているが，バッチ処理時間で見たものは79%となっており，両者の差が大きい。これは，8台の時にロード時間とデータ転送時間の和が全処理時間の18%にも達しているからである。したがって，WSの台数をさらに増やしてゆく場合には，この点を改善する必要がある。

3.3.2 4台のSUNワークステーションを使用した場合（Case 2）

先のCase 1では，比較的小規模(400粒子／バッチ，全64バッチ）な問題を対象としたが，ここでは当室のWSおよびネットワーク上に接続された情報システムセンタのWSを合計4台使用し，比較的大規模（1280粒子／バッチ，全512バッチ，総粒子数655,360個）な問題を実行し全処理時間を測定した。測定環境および測定結果をTable 3.2に示す。これから，4台の時の並列化効率が63%と先のCase 1の場合の86%と比べて，かなり悪いことがわかる。Case 1に比べ計算時間が長いため，相対的にロード時間やデータ転送時間による並列化効率悪化の効果は小さいはずであり，何か別の要因であると思われる。この要因について次に検討した。

先のCase 1とCase 2の測定環境で異なる点は，Case 1では，すべての測定がパラレルウェアを組み込んだ並列版MCACEコードを実行させて行ったのに対し，Case 2では，WS 1台の時にはパラレルウェアを組み込み以前のオリジナルスカラー版 MCACEコードを実行して測定している点である。従って，Case 1のプログラムは並列化のためにホストプログラムとノードプログラムに分割されており，実行時間には，これらプログラム間での通信のオーバーヘッドおよびパラレルウェアのオーバーヘッドが含まれていると考えられる。この点について，SUN Sparc-2とSparc-ipxを使って確かめてみた。サンプル問題としては1280粒子／バッチ，全24バッチの小規模なものを使用し，パラレルウェアの有無による効果を Sparc-2を1台使用して測定した。また，パラレルウェアを使った時のWS台数の増加と処理速度倍率の関係についてSparc-2とSparc-ipxを使って測定した。測定結果をTable 3.3に示す。同表中のCase 1とCase 2を比べれば，パラレルウェアによる並列化のオーバーヘッドにより，WS 1台の場合に，約33%の効率悪化が見られる。一方並列化版のみで比べれば，処

理速度向上はWSを1台から2台にした時，約2倍となり理想的な並列化となっていることを示している。このパラレルウェアによるオーバーヘッドを軽減するため，以下のことを行ない処理時間を測定した。

1　パラレルウェアシステムバッファの最適化

パラレルウェアのexcustomコマンドによりシステムバッファのサイズを変更

2　パラレルウェアのシステムバッファの最適化及びパラレルウェアバージョンアップ

1の最適化に加えこれまでのテストに使用していたパラレルウェア ver3.2.4に対し，最新バージョンであるパラレルウェア ver3.2.5を使用

測定結果をTable 3.4に示す。この結果，パラレルウェアのシステムバッファの最適化により26％のオーバーヘッド，バージョンアップにより10％のオーバーヘッドに押されられることが分かった。さらに，この10％のオーバーヘッドの詳細をパラレルウェアのコミュニケーションプロファイラにより測定したところプログラムを分割したことによる通信のオーバーヘッドが5％，またパラレルウェアのシステムオーバーヘッドが5％程であることがわかった。

このような，オーバーヘッドは，並列計算機AP1000でもみられ，並列処理を実現するために，スカラーの場合に比べプログラム間の通信に余分な CPU資源を消費する必要がある訳で，今回のパラレルウェアによる並列化のようにこのオーバーヘッドを極力小さくすることで全体的並列処理の絶対速度が大幅に改善される。このことは，並列計算機の性能評価では CPU台数の増加による処理速度向上のような相対値のみでなく，同等のハードウェアを有する計算機等を使用して絶対的な処理速度比較を行ない，このオーバーヘッドによる効果についても評価してゆく必要があろう。

3.3.3　並列化効率向上の方策

並列化効率を向上させるための方法としては，基本的にシリアル処理部分を極力排除し，ホストとノード間やノードとノード間のデータ通信を高速化し，さらに各ノードでアイドルがない様にロードバランスを最適化する事である。これら3種の改善方法の内，ロードバランスに関しては並列版 MCACEコードにあまり問題はない。これは，各ノードに動的にバッチの処理を割り当てているからである。つまり，各ノードにはホストから1バッチ又は2バッチを単位として処理を依頼し，その処理が終了したノードには再び次のバッチを与える様にしてあり各ノードの CPUにアイドルが生じない様なアルゴリズムになっている。

(1)　シリアル処理部分の最適化

ホスト・ノード型の並列化モデルでは「パラレルウェア」のみならず，総てに共通して発生するシリアルなオーバーヘッドとして，各ノードへのプログラムのロードおよびホストからノードへの初期データの送信がある。これは，ホスト・ノード間の通信速度の問題とも言えるが，WSネットワークパラレルでは通信速度はイーサネットを使用しており，あまり高速とは言えない。したがって，プログラムに対し，次の様な最適化を行うことが考えられる。

①　各ノードへのプログラムロードを，ホストプログラムが「パラレルウェア」を介して行うのではなく，NFS(Network File System)により直接実行形式ファイルをノードへロードすれ

ば程度の高速化が行える。これは，ホストプログラムでノードプログラムをロード(KXLOAD)する際，実行形式のノードプログラムを絶対パスで指定すれば実現できる。

② ホストからノードへ送信される初期データは，現在の所，関連するコモンブロックデータを全て送信しているが，これを必要なものだけに制限することで送信データ量の低減化を行う。ただし，このコモンブロックデータの仕分けは，一つの配列を問題依存で細かく区切って利用している様な場合は，容易な作業ではない。

③ 上に示した②の方法の代りに，Fig. 3.6に示すように各ノードで個々に入力データを必要なファイルから読み込み，初期データを作成する方法が考えられる。これも，①と同様 NFSを利用することになる。この時には，各ノードが同一の処理を行うため，無駄なように思えるが，ホストの入力データ処理後に行われるデータ送信時間が節約されるため，全体の処理時間は短縮されることになる。つまり，全セルで共通の処理を，ある1つのセルで行い，その結果を全セルに送信してやる方法は，延べで見たセル合計の CPU時間では確かに無駄がないかも知れぬが，ある問題を処理するのに要する時間という点からは全セルへの送信時間だけ冗長となってしまう。

(2) データ通信の高速化

ホスト・ノード間等の通信速度は，既存のネットワーク通信速度およびネットワーク上のトラフィック量に依存し，プログラムを改良して対応できるものではない。WSが同一の部屋にある場合などは，SCSIバスによる結合が可能であろうし，今後ネットワーク通信量の増大に伴い100Mbpsや1 Gbpsのネットワークも整備されて来よう。

次に，「パラレルウェア」による並列化作業中に見られた，通信に関連する現象について述べる。「パラレルウェア」では，ホスト・ノード間，ノード・ノード間のデータ通信を行う時の手順は次の様になる（ノードからホストへデータを転送する場合）

① ノードプログラム中で，ライブラリルーチンKXWRITによりホストへデータを送信する。

② ホストプログラムではまず，ノードからのデータを受信する部分へ到達したら，データが着信しているかどうかをライブラリルーチンKXTESTにより確かめる。

③ 着信したデータは，ライブラリルーチンKXREADによりホストプログラムへ取り込まれる。

ここで，ホストおよびノードのプログラムは独立して実行されているため，ノードからデータを送信しても，直ぐにホストはデータを受信できない場合や，逆にホストがデータを受信しようとしても，ノードが未だデータを送信していない場合が有り得る。前者の場合には，ホストのプログラムがデータ受信をする所に達すれば問題なくデータ処理が行われる。しかし，後者の場合には，ホストはノードからのデータが着信するまで待機する必要がある（同期待ち）。パラレルウェアでは，この同期機能を提供されるデーモンプロセスが自動的に行ない，着信データをパラレルウェアのシステムバッファに保管されている。そのためユーザプログラムでは，ノード間の通信同期を記述する必要はなく，着信データのタイプとサイズを指定してシステムバッファからデータを取り込むことでデータを受信できる。並列版の MCACEは，ノードから送信されるデータのサイズがデータにより異なるため，パラレルウェアによる並列化では，ユーザプ

ログラムに着信データを取り込む前に，着信しているデータのサイズを調べる必要がある。

そのため着信データのサイズを含めた属性を調べるためのライブラリルーチンKXTESTを，Fig. 3.4に示すように，一定周期でループさせる方法を用いた。Fig. 3.4の中で，SLEEP関数が使用されているが，これが全体の並列処理効率に大きく影響している。効率の面からみれば，SLEEP関数はない方が，データ着信を最小の遅れ時間で確認できて良いように考えられるが，実際にはSLEEP関数を使用しないと効率が大きく低下してしまった。これはホストプログラムを実行するワークステーションは同時にノードプログラムも実行するため，着信を確認するループがbusyループとなり CPU資源の殆どを消費してしまうと，ノードプログラムが実行されなくなり，さらにパラレルウェアのデーモンプロセスのレスポンスも悪化し，結果として効率が低下することがわかった。そのため，SLEEP関数をループの中に導入することでホストプログラムを一定周期で休止させ，他のプロセス特にノードプログラムに CPUを空け渡した。これにより，並列処理効率が向上し，先に述べたような結果を得ることができた。

3.4 簡易並列処理

第2章に示したMCACEコードの並列化アルゴリズムは，基本的にMCACEコードのコピーを別々のCPUで実行し，それぞれの CPUで計算したバッチ毎の結果を一つのファイルへ総べて書き込み，全バッチ処理終了後は，シリアルの MCACEコードと同様の統計処理を行うものである。これを，「パラレルウェア」等の並列用ソフトを使用しないで実現することを試みた。

複数WSによる簡易並列処理では，NFS(Network File System)を利用した。通常のWSは，unixの標準機能としてNFSを有しており，NFSはディスクシステムを複数のWSが共有することを可能にする。したがって，NFS上にある総てのWSは，同一のファイルにアクセス可能であり，第2章で用いたアルゴリズムが適用可能となる。第2章で使用した Monte用プログラムの内，変更を要したのは主にメインプログラムとサブルーチンMORSEである。

3.4.1 NFSを利用した並列処理

複数WSが同一のディスクを共有可能とする NFSは，Fig. 3.7に示す様な構成となっており，同一のファイルを全WSで共有すれば，このファイルを介してWS間の通信を行うことができる。そのため，通信量の比較的少ない MCACEコードでは，十分実用になると思われる。Table 3.5にメインプログラムをTable 3.6にサブルーチン MORSEの関連部分を示す。Table 3.5のメインプログラムでは，MCACEコードを実行するWSを互いに識別するためのノード番号を読み，それに対応する入力データファイル，出力データファイルおよび作業用ファイルをオープンする。ここで，論理機番48と41のファイルは共通に参照されるもので，簡易並列処理の keyとなるものである。48番は，現在までに並列処理された全体のバッチ数を表示するカウンタの役割を果し，41番のファイルは並列処理されたバッチ毎の結果をまとめて記録する。また，Table 3.6の 300行では，41番のファイルをアペンドモードでオープンしている。300行では，既に1バッチ分の処理が終っており，318行目で，そ

の結果が41に書き込まれる。また，ファイル48を 305行目でオープンし，カウンタの値を１つ進めている。

3.4.2　並列化効率の測定と問題点

簡易並列版 MCACEコードを実行するには，あるWS上で，ウィンドウシステムを使い，使用するWSの数だけウィンドウをオープンし，各ウィンドウから NFS上のWSへ１台づつリモートログインをする。次に，簡易並列版 MCACEコードの実行環境ディレクトリへ移動し，各ウィンドウで実行形式ファイルを実行すれば良い。この点は，さらにunixのrpcコマンドをプログラム中から実行すれば，より便利なものとなろう。実行環境およびウィンドウシステム上での実行状況を付録Ａ３に示す。

測定は，当研究室で現用中の以下の３台のWSを使用して行った。

① Sparc-10 Model 30

② Sparc-2

③ Sparc-ipx

さらに，イーサネット上に接続された情報システムセンタ所有の，

④ Sparc-10 Model 30

を加えて４台構成とした場合についても測定した。使用した問題は，バッチ当り1280粒子，全 512バッチのサンプル問題である。Unixのtimeコマンドを使用し，Table 3.7の様な結果を得た。また，各WSでのCPU使用時間，CPU専有率，経過時間はTable 3.8の様である。これらの表から，並列化効率が３台の場合で 103%，４台の場合で97%と非常に良好な結果を得ることができた。この事から,NFSを使って，あるファイルを仮想的な共有メモリとして使う簡易並列処理の有効性が示されたものと言えよう。ここで，３台のWSの場合に，並列化効率が 100%以上となるのは，各WSの性能を SPEC fp値を指標として評価したためと思われる。つまり,MCACEコードを各WSで個々に実際に実行して処理速度を比較すれば，各WS相互間の相対処理速度は,SPEC fp値で見た相対処理速度と多少異なるからと考えられる。

ここで,NFSによる簡易並列化の問題点として，中心的役割を果たす共通ファイルが，同時に各WSからオープンされた場合には，ファイルアクセスを排他的に行えないため，オープンを実行した複数台のWSの内，最後にクローズ処理をしたWSの有する内容が書き込まれる点がある。つまり，一度オープンされたファイルは，その内容がバッファへコピーされ，クローズ命令が発せられるまで，バッファでの書き込み，読み出しが行われ，実際のファイル内容は変更されないためである。本報では，通常の FORTRAN言語のみを使って簡易並列化を行っており，ファイルアクセスを排他的に制御するルーチンを作成するまでには至らなかったが，この様なルーチンの作成については今後検討したい。

ただし，簡易並列版 MCACEコードの実行時には，上述の問題点による不都合は特に発生しなかった。これは，各バッチ単位で計算結果を共有ファイルへ書き込んでいるが，全く同時に任意の２台以上のWSで単位バッチの計算が終了する事が無かったと言うことである。ここで使用した問題は，1280粒子／バッチで全512バッチあり，３～４台のWSで実行しているため，少なくとも100回以上，

共有ファイルへ2台以上のWSが同時にアクセスする可能性があった。それにもかかわらず，1度も同時アクセスが発生しなかったのは，同一粒子数のバッチを計算処理しても計算時間は常に一定とはならない。モンテカルロ計算本来の性質による所もある。

ここで，もし共有ファイルへ2台以上のWSが同時アクセスした場合について考える。共有ファイルとしては，Table 3.5中に示した。41番と48番のファイルである。41番のファイルは，300 行のオープン文でアペンドモードでアクセスしている。そのため，41番のファイルが2台以上のWSでオープンされた場合は，2つ以上のバッチ処理計算結果は，上書きされたような形になって，最後にクローズ文を実行したWSの結果のみが41番のファイルへアペンドされる。したがって，入力データで指定した数以下のバッチ数処理結果しか41番に残らない。この時，最終的な統計処理部で，41番のファイルから指定バッチ数分の処理結果を読み出し不能となり，プログラムは「EOF detected」でアボートする。一方，バッチ数のカウンタとして使用している48番のファイルは，同時アクセスの場合はカウンタは1つしか進まないことになる。したがって，48番のファイルを使ったカウンタは一つづつ指定バッチ数まで正確にカウント可能である。ただし，同時アクセスをした時のバッチは，逆に一つしかカウントされないため，計算結果自体も一つしか得られなかったと見てしまうので，最終的に入力データで指定したバッチ数分以上の冗長な計算が含まれることになる。

上述のような共有ファイルの同時アクセスによる問題点を解決するための方策としては次の様な事が考えられる。

① 同時アクセスを許さない排他的ファイルアクセスを可能とするシステムルーチン的なものを作成する。ただし，このルーチンの導入による CPUへのオーバーヘッドは微小なものでなければ，導入のメリットはない。

② 先の48番の様なカウンタ的ファイルを使い，41番の様な使い方をしない。この時に発生する冗長計算は許容する必要がある。

③ プログラムのアルゴリズムを多少変えて，41番のファイルの最終処理では，このファイル中に記録されたバッチ数分だけの処理を行い。入力データによる指定を無視する。これは，モンテカルロ法による計算である事から許容されよう。

3.5 今後の展望

本章では，複数WSによる並列処理について検討したが，モンテカルロ法による遮蔽計算では，その計算アルゴリズムが非常にシンプルであるため，あまり複雑な並列処理用データ通信を必要としない。極端な言い方をすれば，全く独立にモンテカルロ計算を多数のWSで行い，全WSの計算終了後，全部の結果を統計処理するプログラムを別途実行しても何ら不都合はない訳である。このレベルに少し工夫をしたものが，前節で示したNFS利用の並列処理と考えられる。NFSを利用した並列処理では，並列化による CPUへのオーバーヘッドが特に見られず理想的な並列化効率を示した。したがって，「パラレルウェア」の様な一般的並列化用ライブラリは，特にモンテカルロ遮蔽計算では並列化効率を悪化させてしまい，有効とは言い難い。むしろ，プログラム中で使用されている並列

化用ライブラリを見て，並列処理実現用の最適化されたデーモンを自動生成する様な機能が求められよう。

また，MCACEコード等，従来のコードは大型計算機によるバッチ処理を念頭に作成されている。そのため，最終的に必要な結果（例えば，統計誤差が１％未満の結果）を得るまで計算を実行するのではなく，CPU時間やこれに対応する総粒子を制限とした計算が行われていた。しかしながら，WSでは CPU時間に対する制約は無くなり，ユーザーは最終的に必要な結果が得られるまでプログラムを実行することが可能となった。したがって，複数WSによる並列処理でも，途中経過を必要に応じグラフィック表示し，統計誤差等の収束状況を確認できるような機能が重要なものとなって来よう。

また，NFS利用の並列処理はNFSがunix標準機能であることから，WS機種に依存しないため有望な方法と言えよう。今後，大型機もunixで動作する用になり，大型機も含めた並列処理も容易に実現可能となろう。したがって，共有ファイルの排他的アクセスを実現するルーチンを作成することは重要であろう。いずれにしろ数年前のスカラ大型計算機と同等またはそれ以上の処理性能を有するWSが，原研内に次々と導入され，それらが依前として従来のパソコンと同様な使われ方しかされないとすれば，余りにも大きな計算機資源の無駄としか言いようがない。

Table 3.1 Result of Parallel Efficiency Measurement by using upto 8 Work Stations (Case 1)

No.	Model (SUN)	MIPS (Catalogue Value) Alone	Total	(Ratio)	Measured Elapse Time (Sec.) Loading	Data Transfer	Processing for Batches	Total Elapse Time (Sec.)	Speedup Ratio (Efficiency) Total Time		For Batches	
1	Sparc-2	28.0	28.0	(1.00)	7	23	1819	1871	1.00	(100%)	1.00	(100%)
2	Sparc-2	28.0	56.0	(2.00)	9	32	926	975	1.92	(96%)	1.96	(98%)
3	Sparc-2	28.0	84.0	(3.00)	-	-	-					
4	Sparc-2	28.0	112.0	(4.00)	13	38	494	545	3.43	(86%)	3.68	(92%)
5	4/370	16.0	128.0	(4.57)	-	0	-					
6	4/490	22.0	150.0	(5.36)	-	-	-					
7	Sparc-2	28.0	178.0	(6.36)	-	-	-					
8	Sparc-1+	15.8	193.8	(6.92)	25	50	334	409	4.57	(66%)	5.45	(79%)

(Problem Size ... 400 particles/batch, Total 64 batches)

Table 3.2 Result of Parallel Efficiency Measurement by using upto 4 Work Stations (Case 2)

No.	Model (SUN)	$SPEC_{fp}$ (Catalogue Value) Alone	Total	(Ratio)	Total Elapse Time (Sec.)	Speedup Ratio (Efficiency)
1	Sparc-10 (Model-30)	52.9	52.9	(1.00)	13,281*	1.00 (100%)
2	Sparc-10 (Model-30)	52.9	105.8	(2.00)	—	—
3	Sparc-2	22.8	128.6	(2.45)	—	—
4	Sparc-ipx	21.5	150.1	(2.84)	7,380	1.80 (63%)

(Problem Size ... 1280 particles/batch, Total 512 batches)

WS No.1 belongs to a different laboratory (Information Systems Center)
WSs No.1 to No.3 belong to our laboratory (Fuel Cycle Safety Evaluation Laboratory)
* Measured by using the original scalar version of MCACE code
(Free from Parallel Ware)

Table 3.3 Overhead of Processing Time by Parallelization

Case	Model (SUN)	$SPEC_{fp}$ (Catalogue Value) (Ratio)	Parallelization	Total Elapse Time (Sec.)	Speedup Ratio	
1	Sparc-2	22.8 (1.00)	No	1492	1.00	-
2	Sparc-2	22.8 (1.00)	Yes	2208	0.67	1.00
3	Sparc-2 Sparc-ipx	Total 44.3 (1.94)	Yes	1096	1.35	2.03

(Problem Size ... 1280 particles/batch, Total 24 batches)

Table 3.4 Effects of Buffer Optimization and Version Upgrade on Overhead

Case	Parallel Ware	Buffer Optimization	Elapse Time (sec.)	Overhead (Case n/Case 1)
1	No (Serial)	N/A	1492	(1.00)
2	V. 3.2.4	No	2208	1.48
3	V. 3.2.4	Yes	2008	1.35
4	V. 3.2.5	Yes	1650	1.11

Table 3.5 Simple Parallelization of MCACE Code (Main Program)

```
 1  CMT   THIS IS MAIN PROGARAM
 2        READ(5,*) INODE
 3        IF(INODE.EQ.1) THEN
 4        OPEN(51,FILE='INP.51',STATUS='OLD')
 5        OPEN(61,FILE='OUT.61',STATUS='OLD')
 6        OPEN(21,FILE='WK21',FORM='FORMATTED')
 7        OPEN(48,FILE='ITER',FORM='FORMATTED')
 8        WRITE(48,*) '    0'
 9        CLOSE (48)
10        OPEN(41,FILE='WK1',FORM='FORMATTED')
11        CLOSE (41,STATUS='DELETE')
12        ENDIF
13        IF(INODE.EQ.2) THEN
14        OPEN(52,FILE='INP.52',STATUS='OLD')
15        OPEN(62,FILE='OUT.62',STATUS='OLD')
16        OPEN(22,FILE='WK22',FORM='FORMATTED')
17        ENDIF
18        IF(INODE.EQ.3) THEN
19        OPEN(53,FILE='INP.53',STATUS='OLD')
20        OPEN(63,FILE='OUT.63',STATUS='OLD')
21        OPEN(23,FILE='WK23',FORM='FORMATTED')
22        ENDIF
23        IF(INODE.EQ.4) THEN
24        OPEN(54,FILE='INP.54',STATUS='OLD')
25        OPEN(64,FILE='OUT.64',STATUS='OLD')
26        OPEN(24,FILE='WK24',FORM='FORMATTED')
27        ENDIF
28        CALL ELM000(INODE)
29        STOP
30        END
```

Table 3.6 Simple Parallelization of MCACE Code (Major Part of Subroutine MORSE)

```
298  CMT ********************
299     590 CONTINUE
300    9200 OPEN(41,FILE='WK1',FORM='UNFORMATTED',ACCESS='APPEND',ERR=9900)
301         GO TO 9100
302    9900 CALL SLEEP(1)
303         GO TO 9200
304    9100 CONTINUE
305         OPEN(48,FILE='ITER',STATUS='OLD')
306         READ(48,*) ITBUF
307         ITBUF=ITBUF+1
308         REWIND (48)
309         WRITE(48,*) ITBUF
310         CLOSE (48)
311         ITERS=ITERX-ITBUF
312         WRITE(6,*) 'NPID=',NPID,'ITERS=',ITERS,'ITBUF',ITBUF
313  CMT    IF(ITERS.LT.0) THEN
314  CMT       CALL PLUNLOCK(LOCKVA)
315  CMT       RETURN
316  CMT    ENDIF
317         CALL BANKR(-3)
318         CALL OUTPT(2)
319         CLOSE (41)
320         IF(ITERS.EQ.-3) GO TO 630
321         GO TO 610
322  CMT ************************
323     610 IF(NSOUR)                              160,160,620
324     620 ITSTR=1
325  C   END OF BATCH
326         GO TO 160
```

Table 3.7 Result of Parallel Efficiency Measurement
(Simple Parallelization of MCACE Code)

No.	Model (SUN)	SPEC (Catalogue Value) Alone	Total	(Ratio)	Total Elapse Time (Sec.)	Speedup Ratio (Efficiency)
1	Sparc-10 (Model-30)	52.9	52.9	(1.00)	13,281*1	1.00 (100%)
2	Sparc-2	22.8	75.7	(1.43)		
3	Sparc-ipx	21.5	97.2	(1.84)	7,025*2	1.89 (103%)
4	Sparc-10 (Model-30)	52.9	150.1	(2.84)	4,808*2	2.76 (97%)

(Problem Size ... 1280 particles/batch, Total 512 batches)
*1 Measured by using the original scalar version of MCACE code
*2 Maximum elapse time among Work Stations

Table 3.8 Analysis of Processing Times
(Simple Parallelization of MCACE Code)

(1) Execution by 3 Work Stations

No.	Model (SUN)	CPU Time (Sec.)	CPU Occupancy (%)	Elapse Time (Sec.)
1	Sparc-10	6897	98	6970
2	Sparc-2	6984	99	7018
3	Sparc-ipx	6983	99	7025

(2) Execution by 4 Work Stations

No.	Model (SUN)	CPU Time (Sec.)	CPU Occupancy (%)	Elapse Time (Sec.)
1	Sparc-10	4080	98	4730
2	Sparc-2	3258	67*	4808
3	Sparc-ipx	4695	99	4727
4	Sparc-10	4661	98	4714

(Problem Size ... 1280 particles/batch, 512 batches)
* Sparc-2 executed other processes during the measurement and worked as NFS server.

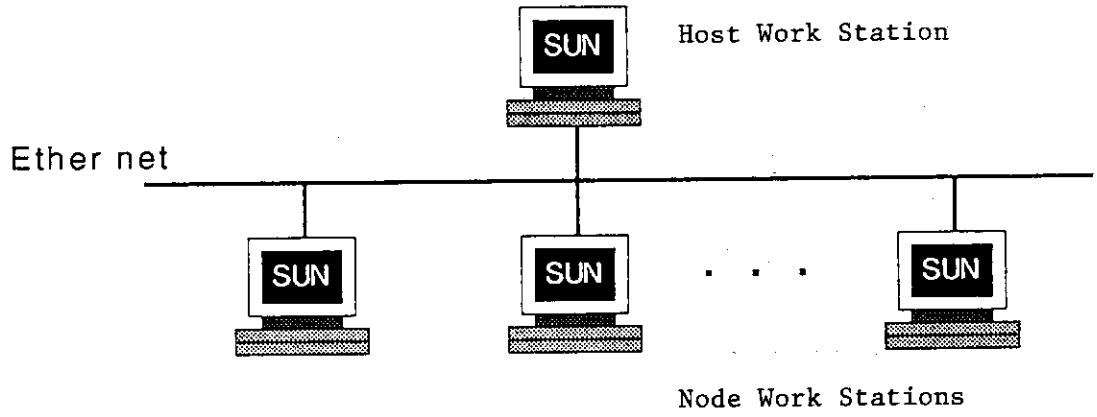


Fig. 3.1 Configuration of Work Stations for Parallel Computing using Network Connection

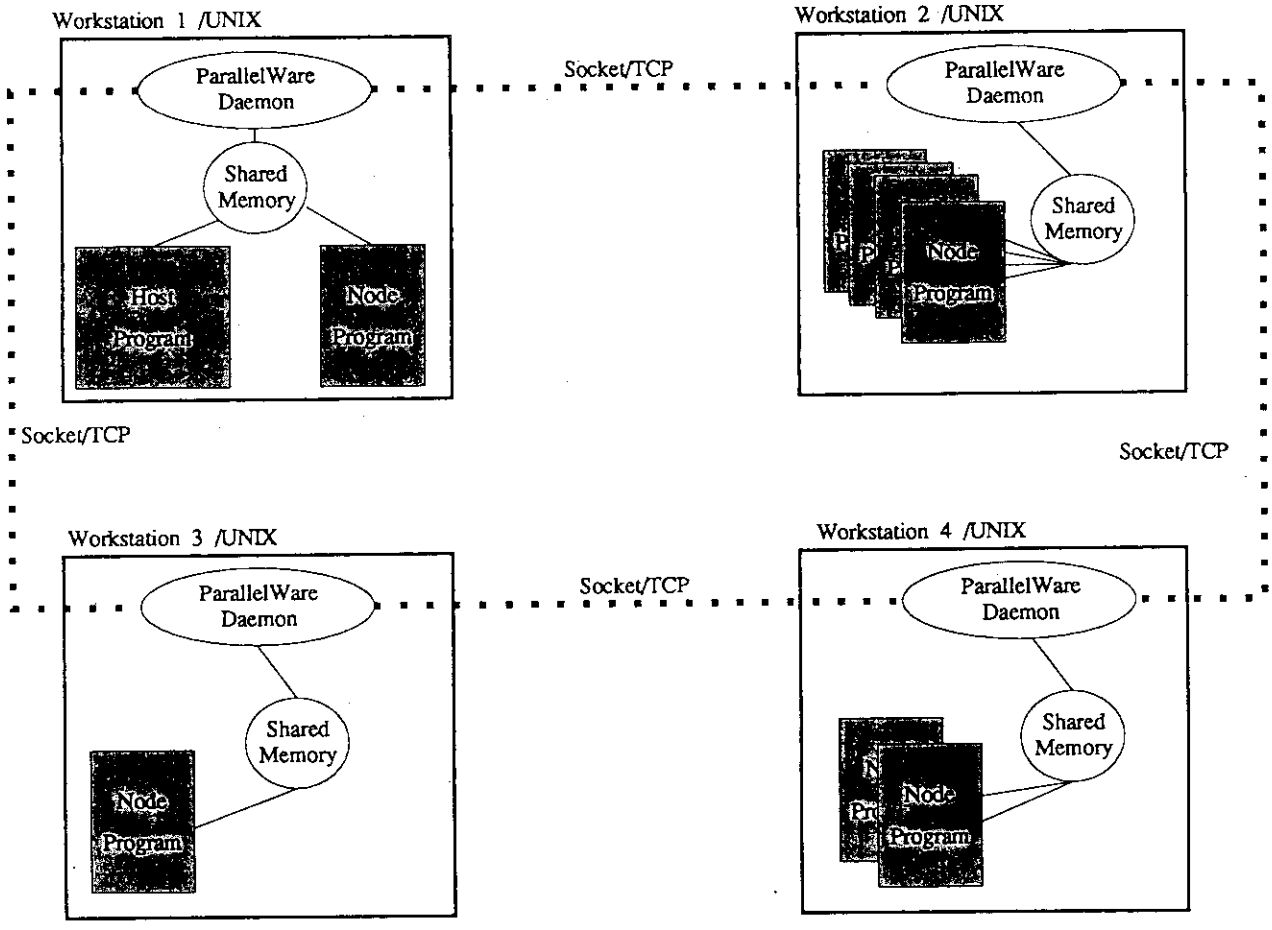


Fig. 3.2 Parallel Computing by "Parallel Ware"

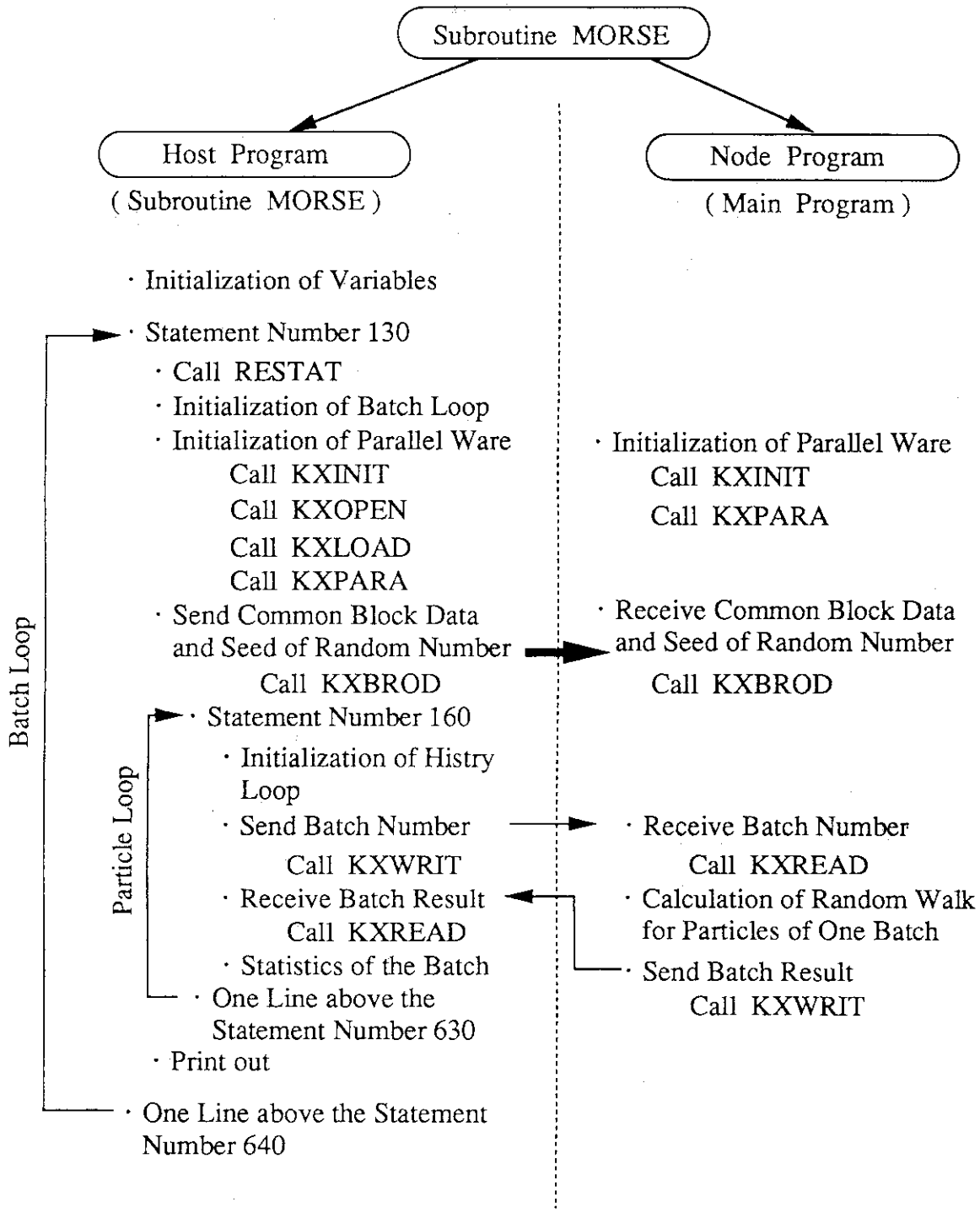


Fig. 3.3 Parallelization of Subroutine MORSE
(Separation of MORSE into Host and Node Programs)

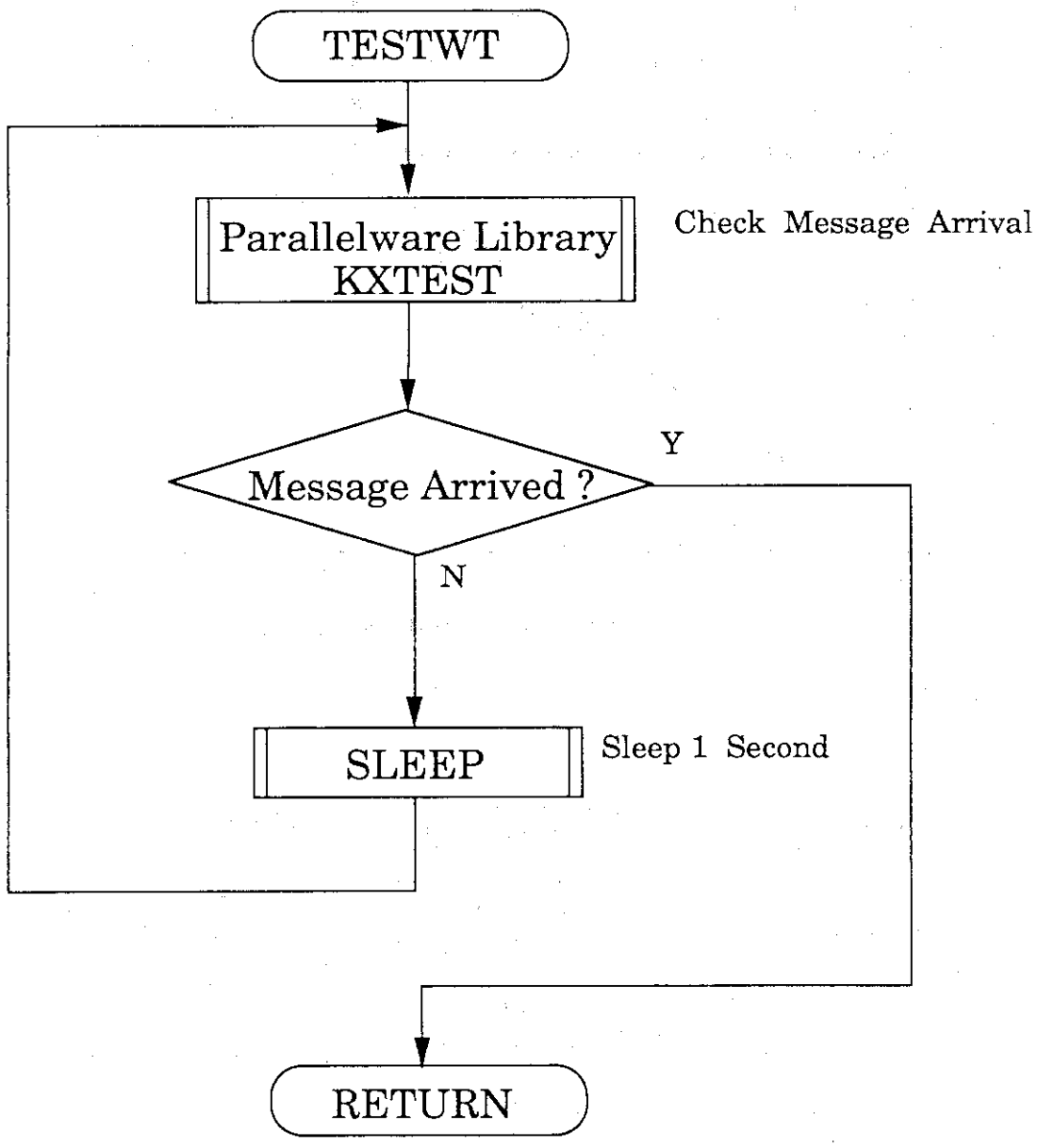


Fig. 3.4 Subroutine TESTWT to wait Messages from Node or Host Programs

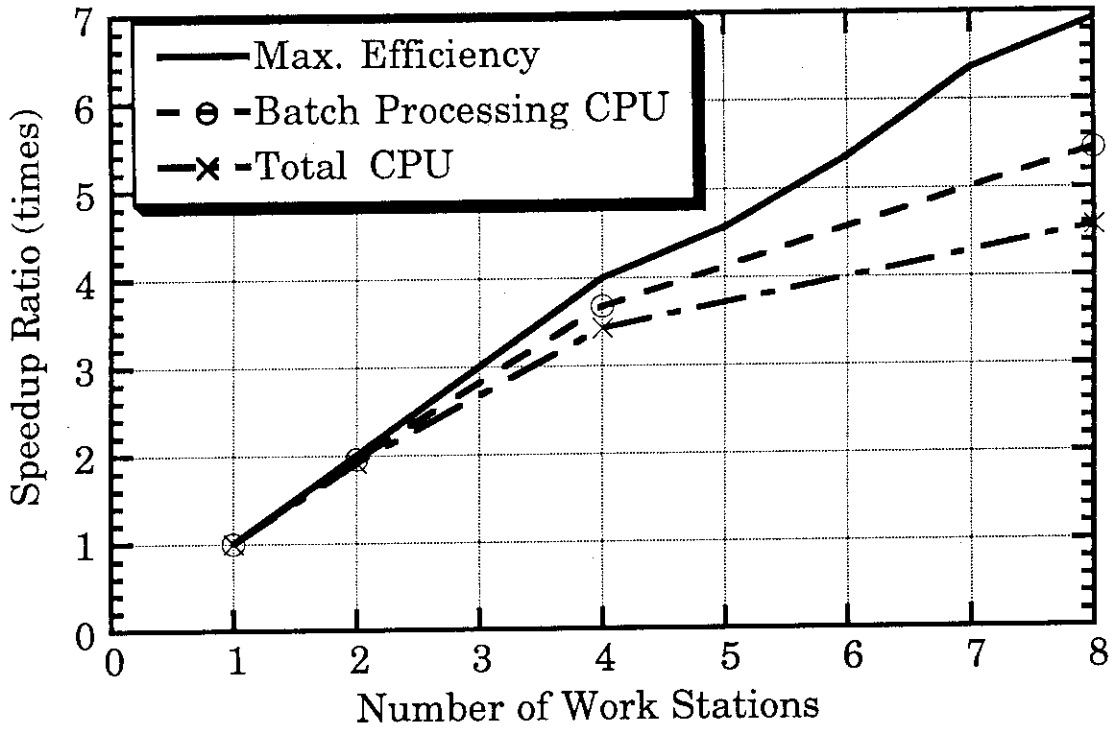


Fig. 3.5 Speedup by Parallel Computing by using upto 8 Work Stations

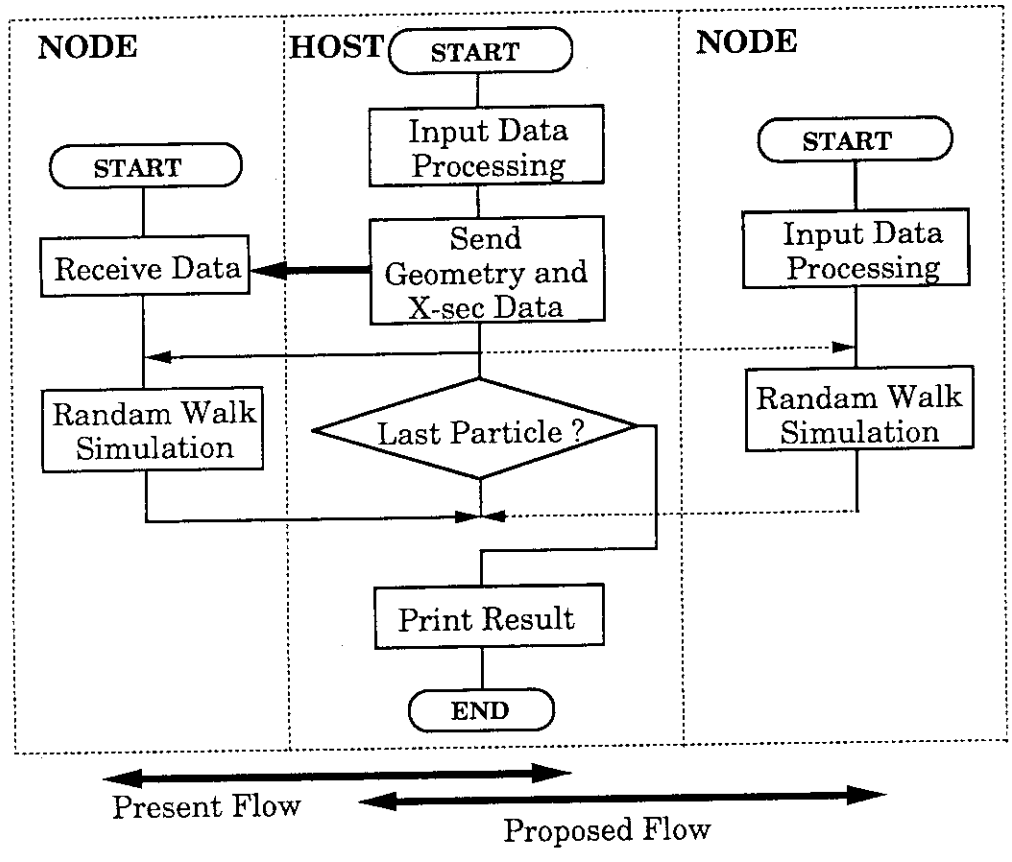


Fig. 3.6 Improvement for Higher Parallel Efficiency

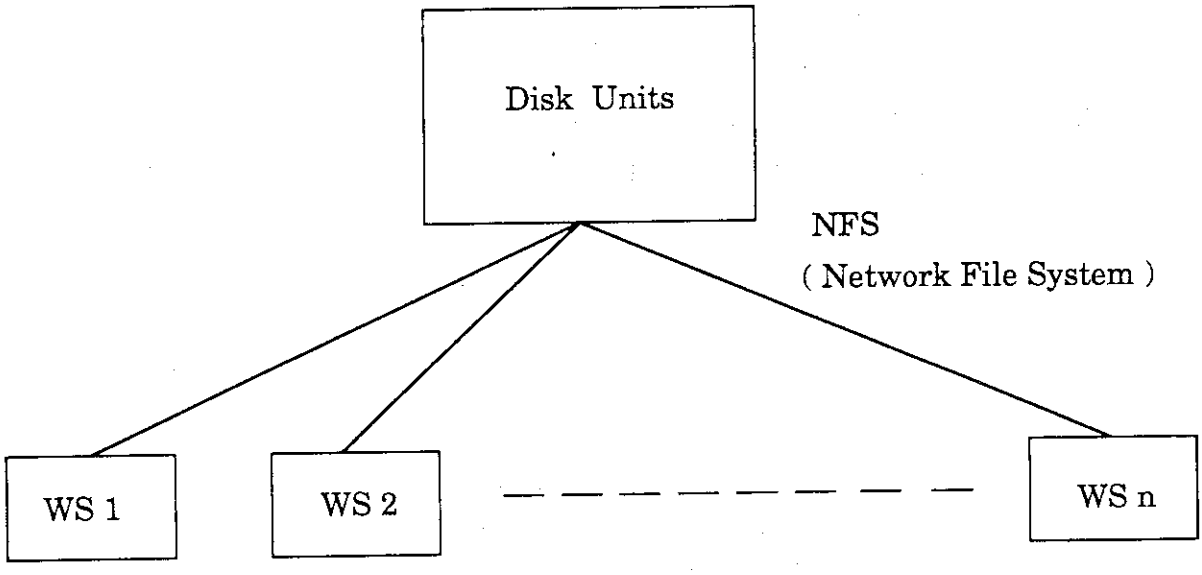


Fig. 3.7 Common File for Multiple Work Stations under Network File System

4. 結　　論

原研に導入された共有メモリ型並列計算機Monte-4および，複数ワークステーションを使った並列処理について遮蔽安全評価用モンテカルロコードMCACEを例として検討した。

Monte-4はベクトルプロセッサを4台有する共有メモリ型の並列計算機であり，本報ではMCACEコードのコピーを作り，4台のCPU上で独立に実行させ，結果のみ4CPUに共通した同一ファイルに書き込むことで並列処理を実現した。採用した並列化アルゴリズムが単純であり，かつMCACEコード中，並列実行されない部分は全体のごく一部であるため，4CPUを使う事により，1CPUの場合のほぼ3倍の高速化を達成できた。

一方，複数ワークステーションを使った並列処理では，ワークステーション間のソケット通信機能を使った並列処理用ソフト「パラレルウェア」を用いて，MCACEコードの並列処理をホスト・ノード型モデルにより行った。並列版MCACEコードをワークステーション台数を増やしながら実行速度を測定した所，台数にほぼ比例した処理速度倍率を得ることができた。しかしながら，並列処理を実現するために必要となるオーバーヘッドがかなり大きく，MCACEコードの場合には，CPU資源の約30％を消費していることがわかった。このため，並列処理用ソフトや実行環境の最適化を行い，オーバーヘッドを極力小さくする必要があろう。このことから，並列処理用ソフトや並列用OSを使用している環境では，シリアル処理用環境よりオーバーヘッドが大きくなっていることが考えられ，CPU台数の増加とともに処理速度も相対的に向上したとしても，絶対的な処理速度で見ると，思った程の効果になっていないことがあるので注意する必要があろう。

一方，標準のNFS(Network File System)環境では，複数ワークステーションが同一ファイルを参照できる。このファイルを共有メモリとして考え，先のMonte-4で使用した並列化アルゴリズムを適用し，複数ワークステーションによる並列処理を実現した。この並列処理では，特別な並列用ソフトは使っておらず，Fortran言語とNFSだけを使っており，どこのワークステーションでも見られる環境で簡単に並列処理を実現しうる可能性を示した。また，MCACEコードによる処理速度測定では，オーバーヘッドの殆どない，良好な処理速度向上を示した。ただし，複数のワークステーションが，この共有ファイルを同時にアクセスした場合に対する対策を考える必要がある。さらに，ワークステーション間の通信速度は，ネットワークのデータ伝送速度で制限されてしまうため，ネットワーク自体も，より高速なものや，ワークステーション並列処理専用ネットワークなどが今後必要となるかも知れない。

ネットワークを介した並列処理は，今後ワークステーションのみならず，ネットワーク伝送速度が向上するにつれて，大型計算機をも巻き込んだものとなる可能性がある。現在，米国では1Gbpiの超高速ネットワークを国内に張り巡らす計画が進行中で，日本もこの計画に参加する意向の様である[8]。このような高速のネットワーク網が実現すれば，全米の大型計算機を使った並列処理が可能となる訳で，米国内の大型計算機の台数はかなりのものと思われ，並列処理能力は想像を絶す

る莫大なものとなろう。したがって，ネットワーク並列処理用のソフトウァア開発は，今後，ますます重要なものになると考えられる。

ここで，Table 4.1に同一の問題を各種の計算機で実行した結果を示す。同表から，4台のワークステーションを使えば，原研で現用中のスカラーメインフレームM-780の約1.4倍程度の処理速度が達成されていることがわかる。したがって，新型のワークステーションを数台導入して並列処理すれば，モンテカルロ法による遮蔽解析等を十分精度良く行える計算処理環境を容易に実現できると考えられる。

Table 4.1 Comparison of Processing Times

No.	Machine	No. of CPUs	Optimization	Parallel Model	Processing Time (Sec.)	Ratio
1	AP-1000	512	Yes	Host•Node	167	1.0
2	M-780	1	Yes	(Serial)	6884	41.2
3	M-780	1	No	(Serial)	9186	55.0
4	WS×4*1	4	Yes	Host•Node (Parallel Ware)	7380	44.2
5	WS×4	4	Yes	Copy Model (NFS File)	4808	28.8
6	Convex C-3840	4	Yes	Copy Model	3002	18.0
7	Monte-4	1	Yes	(Serial)	7457	44.7
8	Monte-4	1	No	(Serial)	11209	67.1
9	Monte-4	4	No	Copy Model	3898	23.3

(Problem Size ... 1280 particles/batch, Total 512 batches)
*1 Sun Sparc-10 Model 30×2 sets, Sparc-2 and Sparc-ipx, Version 3.2.4 without buffer optimization.

謝　　辞

本報作成に当り，情報システムセンターの富山峯秀氏，樋口健二氏および長谷川幸弘氏には，Monte-4の利用，バグの対応および並列処理時間の測定さらには筆者らが使用しているワークステーション上での末端設定方法等で，何度も御協力を戴いた。同じく，情報システムセンターの横川三津夫氏には，パブリックドメインソフト「PVM」の入手方法について御教示戴き，また，海老原健一氏にはセンター所有のワークステーションの使用を御許可戴いた。

また，（株）富士通には，分散メモリ型並列計算機 AP-1000を利用させて戴き，さらに（株）東京エレクトロンには，4CPUのベクトル・並列機Convex C-3840を利用させて戴いた。

ここに，これらの方々および機関に深く感謝の意を表します。

参 考 文 献

(1) 川添他：「並列計算機を利用した遮蔽安全評価用モンテカルロコード MCACEの高速化(1) MCACEコードの並列化とシミュレーターによる性能評価」，JAERI－M 91-066 (1991).

(2) 高野他：「並列計算機を利用した遮蔽安全評価用モンテカルロコード MCACEの高速化(2) 512セルの並列計算機上での性能評価」JAERI－M 92-193 (1992).

(3) TAKANO M., et al., : "Parallelization of Monte Carlo Code MCACE for Shielding Analysis and Estimation of Its Efficiency by Simulator", J. of Nucl Sci. and Technol, Vol 28, No.5, p.p. 1143～1146 (1991).

(4) 高野他：「モンテカルロ法による遮蔽計算コード MCACEと臨界計算コードKENO-Ⅳの並列化」，日本原子力学会誌, Vol, 34, No.6 (1992).

(5) ASAI K. et al.,:JAERI Monte Carlo Machine", Int, Conf. on Mathematical Methods and Super Computing in Nuclear Applications (M&C＋SNA'93) Karlsruhe, Germany (1993).

(6) HIGUCHI K., et al.,:"Effective Performance of JAERI Monte Carlo Machine", Int. Conf. on Mathematical Methods and Super Computing in Nuclear Applications (M&C＋SNA'93), Karlsruhe, Germany (1993).

(7) YAMANO N., et al.,:"Data Pool:A Direct-Access Data Base for Large-Scale Nuclear Codes", JAERI－M 91-201 (1991).

(8) YAMANO N., et al.,:"RADHEAT-V4:A Code System to Generate Multigroup Constants and Analyze Radiation Transport for Shielding Safety Evaluation",JAERI 1316 (1989).

(9) 読売新聞，平成5年3月31日.

謝　　辞

本報作成に当り，情報システムセンターの富山峯秀氏，樋口健二氏および長谷川幸弘氏には，Monte-4の利用，バグの対応および並列処理時間の測定さらには筆者らが使用しているワークステーション上での末端設定方法等で，何度も御協力を戴いた。同じく，情報システムセンターの横川三津夫氏には，パブリックドメインソフト「PVM」の入手方法について御教示戴き，また，海老原健一氏にはセンター所有のワークステーションの使用を御許可戴いた。

また，（株）富士通には，分散メモリ型並列計算機 AP-1000を利用させて戴き，さらに（株）東京エレクトロンには，4CPUのベクトル・並列機Convex C-3840を利用させて戴いた。

ここに，これらの方々および機関に深く感謝の意を表します。

参 考 文 献

(1) 川添他：「並列計算機を利用した遮蔽安全評価用モンテカルロコード MCACEの高速化(1) MCACEコードの並列化とシミュレーターによる性能評価」，JAERI－M 91-066 (1991).

(2) 高野他：「並列計算機を利用した遮蔽安全評価用モンテカルロコード MCACEの高速化(2) 512セルの並列計算機上での性能評価」JAERI－M 92-193 (1992).

(3) TAKANO M., et al., : "Parallelization of Monte Carlo Code MCACE for Shielding Analysis and Estimation of Its Efficiency by Simulator", J. of Nucl Sci. and Technol, Vol 28, No.5, p.p. 1143～1146 (1991).

(4) 高野他：「モンテカルロ法による遮蔽計算コード MCACEと臨界計算コードKENO-IVの並列化」，日本原子力学会誌, Vol, 34, No.6 (1992).

(5) ASAI K. et al.,:JAERI Monte Carlo Machine", Int, Conf. on Mathematical Methods and Super Computing in Nuclear Applications (M&C＋SNA'93) Karlsruhe, Germany (1993).

(6) HIGUCHI K., et al.,:"Effective Performance of JAERI Monte Carlo Machine", Int. Conf. on Mathematical Methods and Super Computing in Nuclear Applications (M&C＋SNA'93), Karlsruhe, Germany (1993).

(7) YAMANO N., et al.,:"Data Pool:A Direct-Access Data Base for Large-Scale Nuclear Codes", JAERI－M 91-201 (1991).

(8) YAMANO N., et al.,:"RADHEAT-V4:A Code System to Generate Multigroup Constants and Analyze Radiation Transport for Shielding Safety Evaluation",JAERI 1316 (1989).

(9) 読売新聞，平成5年3月31日.

付録Ａ１　Monte-４用並列版MCACEコードの実行

Ａ1.1　実行環境

Monte-4用に並列化したMCACEソースプログラムおよび，実行形式ファイルや入力データは，以下のディレクトリに格納されている。

(1) ソースプログラム

ディレクトリ：/lab3/g0943/j3520/emi/mcace/para

ソースプログラムは，サブルーチン毎に分割されて入っている。

(2) 実行形式ファイルと入力データ

ディレクトリ：/lab3/g0943/j3520/emi/mcace/run.para

ファイル名	内　　容
a-para.out	並列版MCACE実行形式ファイル
INP.51 INP.52 INP.53 INP.54	MCACE入力データファイル （シード乱数以外は，同一内容）
fort.61 fort.62 fort.63 fort.64	MCACE計算結果出力ファイル
WK21 WK22 WK23 WK24	MCACEコード作業用ファイル
WK 1	単位バッチ毎の計算結果格納用ファイル
dp 1 dp 2 dp 3 dp 4	Data Pool形式断面積データファイル （同一内容）

(3) 実行時の注意事項

MCACEコードが使用しているData Pool形式の断面積ファイル（dp1～dp4）は，SUNのWS上で動作するMCACEコードが使用していたバイナリファイルをftpでMonte上にコピーしたものである。FORTRANで作成されたSUN上のバイナリファイルはIEEE形式であり，Monte上でもIEEE形式として使用するため，.loginファイルに次行を挿入する必要がある。

setenv F__UFMTIEEE 91,92,93,94

さらに，SUNのFORTRANでは，OPEN文で使用されるRECLパラメタの単位はバイトであり，Monteの場合のワード単位とはデフォールが異なるため，次行を.loginファイルに挿入した。

setenv F__RECLUNIT BYTE

以上，２つの環境設定がないと実行時にエラーを生じてアボートする原因となる。

また，並列処理効率等を計測するための CPU時間情報を得るため，次行も.loginファイルに加えた。

setenv F__PROGINF YES

ここで，実行環境とは直接関係しないが，telnetで monteへログインした後，viエディタやmoreをウィンドウ上で正常に動作させるために，

setenv TERM xterm

が必要である。またホストとなるWSでもxtermのウィンドウを使用してMonteへログインする必要がある。さらに，ログイン後，xtermウィンドウのサイズを変更した時には，

eval 'resize'

を実行する必要がある。当初，Open Windowsのウィンドウを使って Monteへアクセスしてviエディタを使おうとしたが，viエディタの画面が正常に表示されず，カーソル等のキーも正常に作動せず，止むなく ftpによりファイルを転送してローカルでエディット作業を行い，再び Monte へ転送するという作業を余儀なくされ，非常に効率の悪い作業環境であった。ここで，現在使用中の.loginファイルリストをTable A1.1に示す。

A1.2 プログラム変更部のリスト

第2.2.2節の(1)に示したプログラム全体に対する変更を除き，(2)(3)に示した変更について主要ルーチンの変更部分リストを示す。ここで，プログラム実行時のエラーに対するバグフィックスには，デバッガーpdbxが有用であった。これは，コンパイル時に▽－g▽ オプションを付けて実行形式フィイル a.outを作成する。実行時エラーが発生した時に，入力プロンプトに対しpdbxと入力し，次に whereと入力すればエラーを発生した場所が特定され，便利である。

(1) MAIN

```
 1  CMT    THIS IS MAIN PROGARAM
 2         EXTERNAL ELM000
 3         GLOBAL COMMON /LOCKDT/ LOCKVA
 4         GLOBAL COMMON /ITCONT/ ITCN(4)
 5         DATA ITCN/0,0,0,0/
 6         OPEN(51,FILE='INP.51',STATUS='OLD')
 7         OPEN(52,FILE='INP.52',STATUS='OLD')
 8         OPEN(53,FILE='INP.53',STATUS='OLD')
 9         OPEN(54,FILE='INP.54',STATUS='OLD')
10         OPEN(21,FILE='WK21',FORM='FORMATTED')
11         OPEN(22,FILE='WK22',FORM='FORMATTED')
12         OPEN(23,FILE='WK23',FORM='FORMATTED')
13         OPEN(24,FILE='WK24',FORM='FORMATTED')
14         OPEN(41,FILE='WK2',FORM='UNFORMATTED')
15  C       CALL ELM000(1)
16         CALL PLASGN(LOCKVA)
17         CALL PTFORK(KTID1,TP1,ELM000,1)
18         CALL PTFORK(KTID2,TP2,ELM000,2)
19         CALL PTFORK(KTID3,TP3,ELM000,3)
20         CALL ELM000(4)
21         CALL PTJOIN(KTID1,KTID2,KTID3)
22         STOP
23         END
```

(2) ELM000

```
 1         SUBROUTINE ELM000(NPID)
 2  C * * THIS IS THE MAIN ROUTINE * * * * * * * * * * * * * * * * * * * * * *
 3  C * *
 4  C * * THE FOLLOWING CARD DETERMINES THE SIZE ALLOWED FOR BLANK COMMON  *
 5  C      COMMON NC(35000)
 6          LOCAL COMMON /EMI/
 7        *       NC(500000)

50  CMT
51         LOCAL COMMON /DT009/ NOUTGM
52  CMT
53  C * *
54  C      DATA JUNK/Z48484848/
55  C      DATA JUNK/0/
56  C      NLFT=(LOC(DUM)-LOC(1))/4
57  CMT    Add 1 line
58         LIM=500000
59  CMT
60  CMT   ADD another line
61         NOUTGM=NPID+60
62  CMT

67         ITOUT = 60+NPID
68  C------------- by akemi -----------------
69  CMT    ITIN = 19
70         ITIN = NPID+50
71  C      ITIN = 5
72  C---------------------------------------
73  C      NLFT=(LOC(AGSTRT)-LOC(NC(1)))/4
74  C      DTLIST IS ONLY USED IN JEARI
75         CALL DTLIST(NPID)
76         CALL MORSE(NLFT,NPID)
77         RETURN
78         END
```

(3) DILIST

```
  1  C-------------------------------------
  2
  3        SUBROUTINE  DTLIST(NPID)
  4        DIMENSION  A(20)
  5  C---------- by akemi ----------
  6  C     NIN = 5
  7        NIN = NPID+50
  8  C-----------------------------
  9        NOU = NPID+60
```

(4) MORSE

```
  1        SUBROUTINE MORSE(NLFT,NPID)

 40  CMT***************
 41        GLOBAL COMMON /LOCKDT/ LOCKVA
 42        GLOBAL COMMON /ITCONT/ ITCN(4)
 43  CMT

 50  CMT   Modified to stop MCACE normally 05/02/93
 51        MTFLAG=0
 52        CALL INPUT(MTFLAG,NPID)
 53        IF(MTFLAG.EQ.1) RETURN
 54  CMT   CALL INPUT

 88        ITERS=NITS
 89  CMT **************
 90        ITERX=NITS

296  CMT ********************
297    590 CONTINUE
298        CALL PLLOCK(LOCKVA)
299        ITCN(NPID)=ITCN(NPID) + 1
300        ITERS = ITERX -(ITCN(1)+ITCN(2)+ITCN(3)+ITCN(4))
301        WRITE(6,*) 'NPID=',NPID,'ITERS=',ITERS,'ITCN(1-4)',ITCN
302  C ------------------------------------------------------ 93/03/02 ---
303  C-------- IF(ITERS.LT.0) THEN
304  C          CALL PLUNLOCK(LOCKVA)
305  C          RETURN
306  C         ENDIF
307        CALL BANKR(-3)
308        CALL OUTPT(2)
309        CALL PLUNLOCK(LOCKVA)
310        IF(ITERS.LE.3 .AND. ITERS.GE.1) RETURN
311  C ------------------------------------------------------------------
312        IF(ITERS.EQ.0) GO TO 630
313        GO TO 610
314  CMT ************************
315    610 IF(NSOUR)                              160,160,620
```

(5) INPUT

```
  1        SUBROUTINE INPUT(MTFLAG,NPID)

144  CMT   STOP STATEMENT CAN NOT BE USED IN CELL (PTFORK)
145  789   CONTINUE
146        MTFLAG=1
147        RETURN
148  CMT   STOP
149  C
150    180 CONTINUE
```

(6) JOMIN

```
      SUBROUTINE JOMIN(NADD,I1,I0,NPID)

CMT   DIMENSION  IBIAS(9),ITY(10),COM(15)
      DIMENSION                  COM(15)
      DIMENSION IIBIAS(9),JTY(15)
      CHARACTER*4 ITY,JTYPE,IIBIAS,IOR,IBL,IEND,COM
C  1   2   3   4   5   6   7   8   9   10
C ARB SPH RCC REC TRC ELL BOX WED RPP END
C 24  -2   1   6   2   1   6   6   0
      GLOBAL COMMON /DT016/ IBIAS(9),ITY(10),IOR,IBL,IEND
      DATA IBIAS/ 24,-2,1,6,2,1,6,6,0 /
      DATA ITY/3HARB,3HSPH,3HRCC,3HREC,3HTRC,3HELL,3HBOX,3HWED,3HRPP,
     1         3HEND/
      DATA IOR,IBL,IEND/2HOR,3H   ,3HEND/
       LOCAL
     *COMMON/TAPE/INT,IOT
      IOT=I0
CMT   IOUT=16
      IOUT=20+NPID
C--------- by akemi ---------------
C     REWIND IOUT
      REWIND (UNIT=IOUT)
CMT   REWIND (UNIT=16)
C---------------------------------
```

(7) POPEN

```
      SUBROUTINE POPEN(NUNIT,JCONTR)

       LOCAL
     *COMMON /DPWORK/ LBUFFR,LRECOD,IBUFFR(1000),NRECOD,NODE1,NODE2,
     *                NADWN,NADAT,NDASET,NDATE(9),NINFOM(5),NUTOLD,
     *                NTHOLD,NODOLD(10,2),NA1
       LOCAL
     *COMMON /DD/ DDNAME,IT(3)
      DIMENSION       JCONTR(1)
      CHARACTER BLANK*4,BUF,BL*40,DDNAME*40,NDATE*4
      CHARACTER*8 NDATEL,TDUM
      EQUIVALENCE (NDATEL,NDATE(1))
C     DATA DDNAME/'FT  F001'/
      GLOBAL COMMON /DT025/ BLANK,BL
      DATA BLANK/'    '/
      DATA BL/'                                        '/

C---------- by akemi -----
CMT   OPEN(92,FILE=DDNAME,ACCESS='DIRECT',RECL=3600,STATUS='OLD')
c     OPEN(92,FILE='dp',ACCESS='DIRECT',RECL=3600,STATUS='OLD')
C-------------------------
C
C-------- 2. READ CONTROL SECTION AND SET DATE
C
      IF(NUNIT.EQ.91) DDNAME='dp1'
      IF(NUNIT.EQ.92) DDNAME='dp2'
      IF(NUNIT.EQ.93) DDNAME='dp3'
      IF(NUNIT.EQ.94) DDNAME='dp4'
      OPEN(NUNIT,FILE=DDNAME,ACCESS='DIRECT',RECL=3600,STATUS='OLD')
```

Table A1.1 List of .login

```
umask 77
/bin/stty dec
#bin/stty -echo
#/bin/stty icanon
set path=(. $HOME/bin /xmu/bin /xmu/usr/bin /xmu/usr/ucb /bin /usr/bin /usr/ucb /us
set history=32
set  prompt="[\! $cwd] "
#set prompt="[$LOGNAME/Monte-UX(\!)] "
set mail=(0 /var/mail/$LOGNAME)
setenv MAILCHECK 0
setenv MAILPATH "/var/mail/$LOGNAME%New mail arrived"
setenv PS1 "[$LOGNAME] "
setenv TZ JST-9
setenv EXINIT 'set number | set showmode'
setenv F_RECLUNIT BYTE
setenv F_UFMTIEEE 91,92,93,94,95
setenv F_PROGINF YES
setenv TERM xterm
/bin/date
finger
eval `resize`
alias a alias
a h history
a rm "rm -i"
#source $HOME/.alias
a ls "ls -lF | more "
a fort ' f77m4 -Nv -L f=fort.comp mrgmsg source summary -multi -b -float1 -NW -NO '
a forn ' f77m4 -Nv -L f=fort.comp summary -multi -b -float1 -NO -NW '
a fors ' f77m4 -Nv -L f=fort.comp mrgmsg source summary -b -float1 -NO -NW '
a fopt ' f77m4 -pvctl loopcnt=10000 noassume -multi -NO nodarg '


a cd  'chdir \!* ; set prompt="[\! $cwd]" '
a rm "rm -i"
a rs eval `resize`
a so source
a whos finger
a ps 'ps -u j3520'
#source $HOME/.alias
```

付録Ａ２　「パラレルウェア」並列版MCACEコード実行環境

「パラレルウェア」を使用するための環境設定をまず初めに行う必要がある。実行環境ディレクトリは

/home2/takano/parallelware

である。

(1) 必要なパス設定とネットワーク環境の設定

実行環境ディレクトリにおいて，

<S4a>/home2/takano/parallelware% source .cshrc

を入力し，必要なパスおよび環境設定を行う。ファイル.cshrcの内容をTable Ａ2.1に示す。次に，並列処理に使用するWSのホスト名等の情報を「パラレルウェア」に与えるため，▽domtool▽を起動する。▽domtool▽を入力すると，Table Ａ2.2に示すように２回の問合わせがあり，これに答えると，Fig.Ａ2.1のようなウィンドウが表示される。この例では，４台のWSが相互にネットワーク接続されている。この画面の▽More Options▽をクリックすると，Fig.Ａ2.2に示すようなウィンドウが得られ，先の初期画面より細かいオプションを有している。ここで，WS▽s4a▽に対して与えられている情報は▽Show Machine▽オプションにより，Fig.Ａ2.3の様に表示される。ここで，各マシンに対するユーザー名は，表示されているWS相互間でパスワードなしでリモートログイン可能な状態にWSが設定されている必要がある。このためには，

/etc/hosts　（相手WSのIPアドレス）

～/.rhost　（相手WSのホスト名）

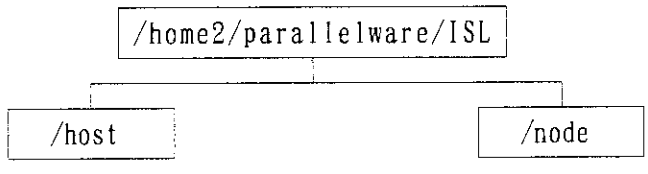
の２つのファイルをエディットする必要がある。また，Fig.Ａ2.3中の▽Server pathname▽のSEXPSEVは，先の.cshrc内で設定されたものである。その他のパラメタは他のWSに対しても同一に設定されている。新たなWSを並列処理用に追加する場合には，これらの情報を入力することとなる。

(2) ディレクトリ構成

パラレルウェアによって並列化されたMCACEコードは以下に示すディレクトリ構成でソースプログラム及び実行モジュールが格納されている。

① /home2/parallelware/ISL

- ホスト及びノード・プログラム共通ソースファイル
- ホスト及びノード・プログラムの実行モジュール
- makeファイル



ディレクトリ構成図

② /host

・ホスト・プログラム専用ソースファイル

・makeファイル

③ /node

・ノード・プログラム専用ソースファイル

・makeファイル

(3) 実行モジュールの生成

実行モジュール生成にはmekefileを使用する。

① cd /home2/parallelware/ISL

② make

(4) MCACEの実行方法

MCACEの実行のための操作法を次に説明する。

① cd /home2/parallelware/ISL

② exinit

③ mcace_h

④ KEY-IN INPUT DATA NAME

<入力データ，フィイル名> （ここではMCACE0.DATA）

⑤ INPUT DATA NAME OK？(y/n)

'y'又は'n'

⑥ Number of calculate nodes？

使用可能なワークステーション台数を入力

⑦ KEY-IN DATAPOOL DATA SET NAME

<データプールファイル名> （ここではdp）

⑧ DATAPOOL DATA SET NAME OK？(y/n)

'y'又は'n'

Table A2.1 List of the file ▼.cshrc▼
(Setup of Parallelware Environment)

```
1  # @(#)Cshrc 1.5 90/11/01 SMI
2  source /.cshrc.owsu
3  #          Network-Parallelware Environment
4  set EXPHOME = /home2/takano/parallelware
5  set path = ($path $EXPHOME/bin/net/sun4 $EXPHOME/bin/tools/sun4)
6  setenv EXPRESS $EXPHOME/bin/net/sun4/express.cst
7  setenv EXPSEV  $EXPHOME/bin/net/sun4
```

Table A2.2 Initiation of ▼domtool▼

```
88 <s4a>/home2/takano/parallelware % domtool
Domtool Version 3.2.4 -- Copyright (C) 1991 ParaSoft
Before changing configuration it is a good idea to run exclean.
Do you wish to exit and run it? [y/n]: n
The file "/home2/parallelware/bin/net/sun4/netfile" already
exists.
Do you wish to modify this configuration or destroy it and start
again?

Continue with existing file? [y/n]: y
```

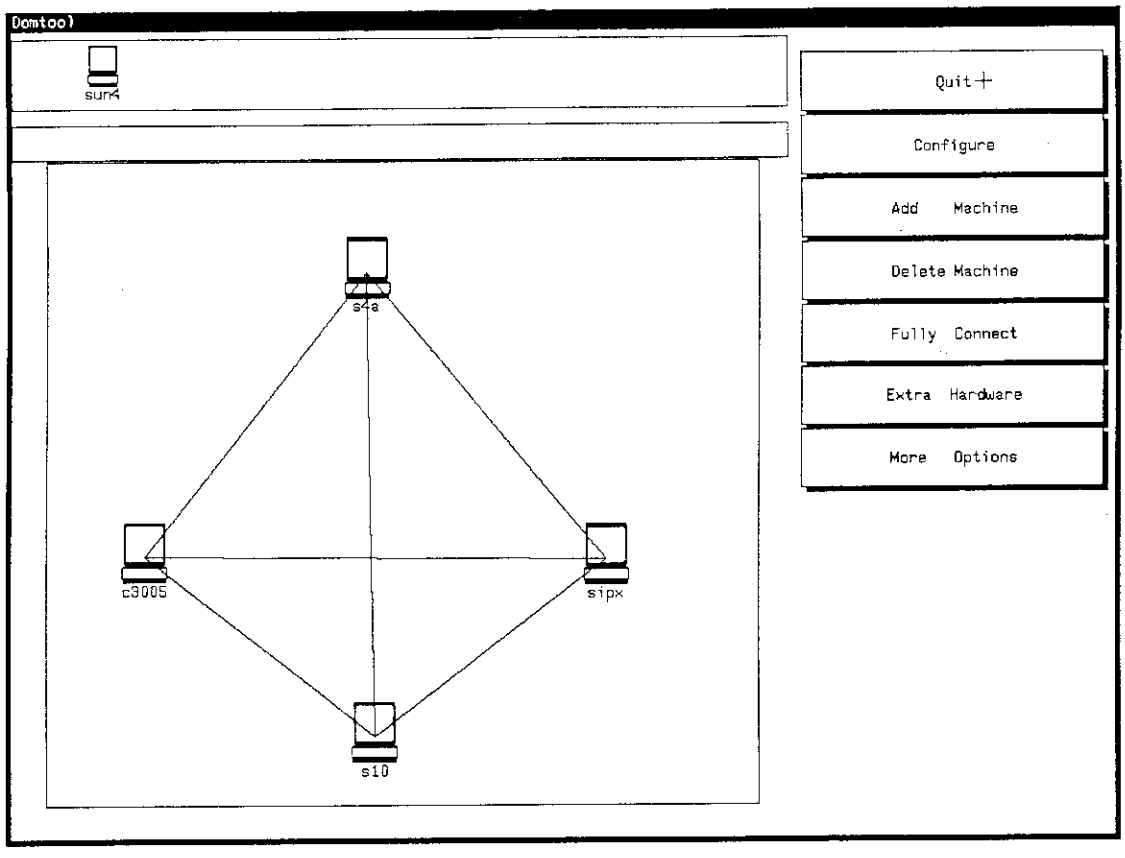


Fig. A2.1 Initial "domtool" Window

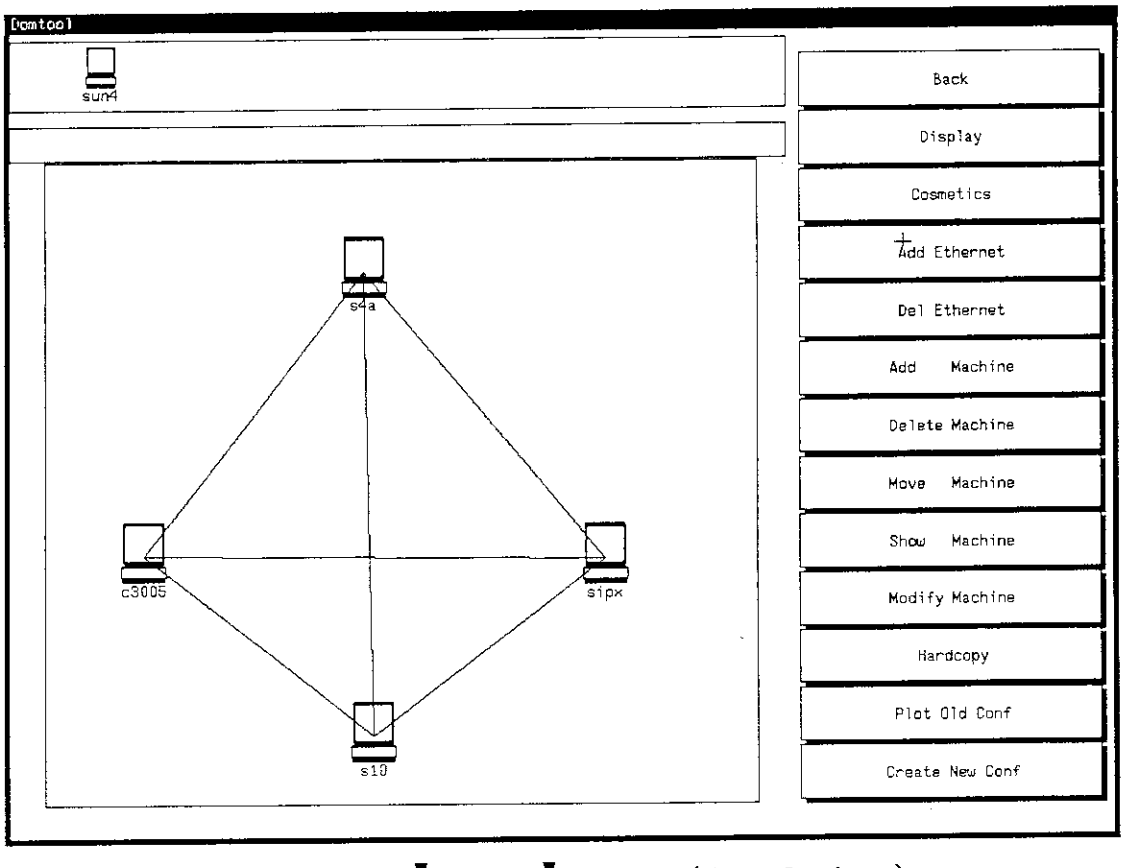


Fig. A2.2 "domtool" Window (More Options)

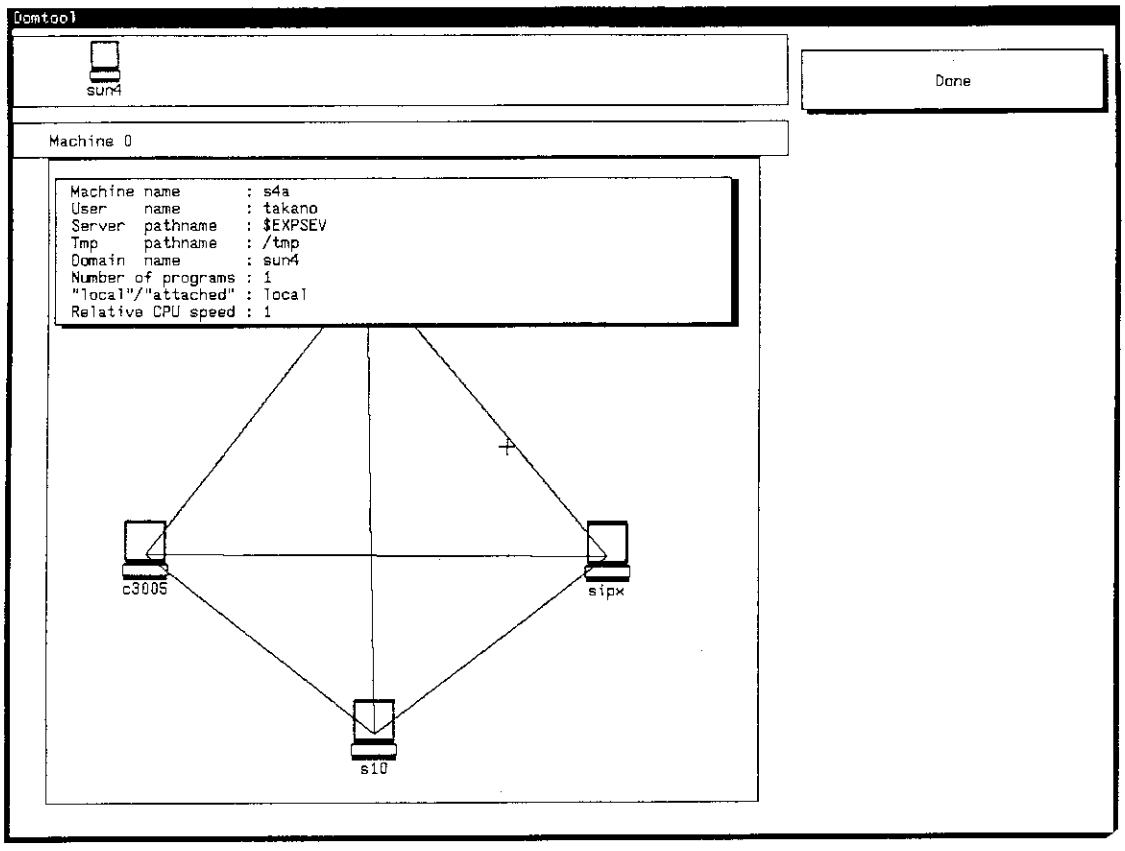


Fig. A2.3 Machine Information of "s4a"

付録Ａ３　簡易並列版MCACEコード実行環境

現在，簡易並列版MCACEコードはディレクトリ

/home1/takano/mcace/para.nfs

以下にあり，実行形式ファイルや入力データ等は，ディレクリ runに格納され，ソースプログラムはディレクトリsourceに格納されている。これらをTable Ａ3.1に示す。また，３台のWSを使用して実行した時のディスプレイの状況をFig.Ａ3.1に示す。各ウィンドウ上に示された値は，Table 3.5の312行のWRITE文による出力である。

Table A3.1　Directory and Files used for Simple Parallelization

(1) プログラム

ディレクトリ：/home1/takano/mcace/para.nfs/source

ファイル名	内　　容
*. f	フォートランソース

(2) 実行形式とデータ

ディレクトリ：/home1/takano/mcace/para.nfs/run

ファイル名	内　容（表3.5参照）
INP.51 INP.52 INP.53 INP.54	MCACE入力データ（乱数シードのみ異なるもの）
ITER	バッチ数カウント用のファイル
OUT.61 OUT.62 OUT.63 OUT.64	MCACE出力ファイル
WK 1	バッチ計算結果書き込み用ファイル
WK21 WK22 WK23 WK24	作業用ファイル
a.out	
dp 1 dp 2 dp 3 dp 4	データプール形式断面積ファイル（全部同一）

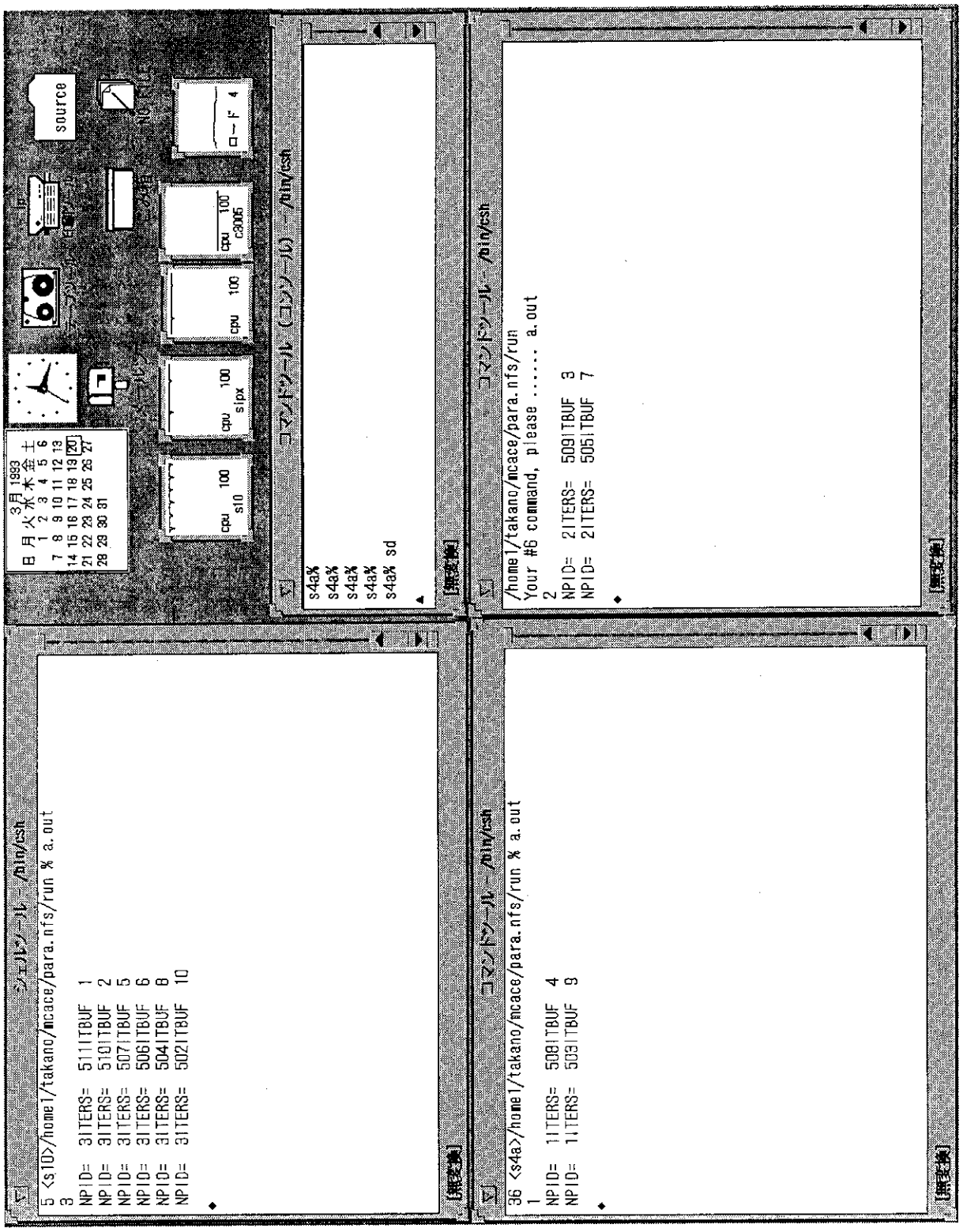


Fig. A3.1 Execution of Simply Parallelized MCACE using 3 Work Stations

付録Ａ４　本報告書で使用したサンプル入力データ

Table Ａ4.1に本報で使用したサンプルデータリストを示す。これは，文献(8)中にMCACEコードのサンプルデータとして示されているものと同じである。Table Ａ4.1では粒子数1280個，総バッチ数512回の場合を示している。

Table A4.1　Sample Data used in the Report

```
 1  MCACE SAMPLE PROBLEM NO.1
 2   1280 2800   512     1     9     0     9     9     0     0    10     2     0     0
 3      0     1     0     0 1.0         0.02    +6 0.0          0.0          2.2    +5
 4      0
 5    0.0
 6    1.0
 7    1.33  +6  1.0   +6  0.8   +6  0.6   +6  0.4   +6  0.3   +6  0.2   +6
 8    0.1   +6  0.05  +6
 9  0123456789
10      0     1     0     0     0     4     9     0
11      1     1     9     1     1     4 100.0     1.0     -3 1.0     -2  0.0
12     -1
13      0
14      0     0              SKYSHINE PROBLEM
15    RCC     1  0.0          0.0          0.0          0.0          0.0          7.6   +2
16                4.1   +2
17    RCC     2  0.0          0.0          0.0          0.0          0.0          7.8103+2
18                7.1   +2
19    RPP     3 -6.0   +4 6.0    +4 -6.0   +4 6.0    +4 0.0          6.0    +4
20    RPP     4 -6.01  +4 6.01   +4 -6.01  +4 6.01   +4 -10.0        6.01   +4
21    END
22    AIR             +1
23    WAL             +2       -1
24    AIR             +3       -2
25    VID             +4       -3
26    END
27      1     2     3     4
28      1     2     1     0
29    CROSS SECTION FROM DATA-POOL
30      0     0     9     9     9    12     4     2     2     0     0   -16     0
31      0     0     0     0     0     0    -1     0
32   &DPUNIT  NLIB=92  &END
33   G09  FX16 AIR  CONC
34    SAMBO INPUT TOTAL FLUX AND DOSE RATE
35      3     1     9     9     0     0    -1     1     1
36    1.0   +4  0.0        1.5     +2
37    2.0   +4  0.0        1.5     +2
38    3.0   +4  0.0        1.5     +2
39    RESPONSE FUNCTION (MR/HR)
40   &DPUNIT  RESD=92  &END
41   G09  RESD DOSE
42    PHOTON/SEC/CM**2/EV
43      1     2     3     4     5     6     7     8     9
44  PRT       1     1     1     0     0     0     9     9     0     0     0
45   0.0          0.0          1.5     +2 0.0          0.0          1.0
46   1.0          0.0          0.0
47   70.0    +2 90.0    +2 110.0   +2 140.0   +2 180.0   +2 220.0   +2 250.0   +2
48   280.0   +2 320.0   +2 380.0   +2
49      1     1     1     1     1     1     1     1     1
50   0.0          6.283        0.0          6.283        0.0          6.283        0.0
51   6.283        0.0          6.283        0.0          6.283        0.0          6.283
52   0.0          6.283        0.0          6.283
53   SOURCE          1     5     3  1.33  +6 0.02    +6     3
54                0.0          0.0          660.0
55                0.0          0.0          0.0
56                0.273        0.0
57      1
58   1.25   +6  1.0
59   END OF SAMPLE PROBLEM : PROGRAM MCACE TERMINATED
```