JAERI-Data/Code
98-007

光量子分子動力学コード［QQQF，MONTEV］のベクトル化および並列化

1998年3月

加藤　香[*1]・功刀資彰・小竹　進[*2]・芝原正彦[*3]

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
印　　刷　いばらき印刷㈱

JAERI-Data/Code 98-007

光量子分子動力学コード[ＱＱＱＦ、ＭＯＮＴＥＶ]のベクトル化および並列化

日本原子力研究所関西研究所光量子科学センター
加藤　香*1・功刀　資彰・小竹　進*2
芝原　正彦*3

(1998年１月26日受理)

本報告書は、光量子物質相互作用シミュレーション用に開発されている、量子分子動力学コードＱＱＱＦの Fujitsu VPP 上でのベクトル並列化および、VPP から Intel Paragon XP/S への移植および並列化、並びに光モンテカルロ－分子動力学ハイブリッドコードＭＯＮＴＥＶに対するVPP から Paragon XP/S への移植および並列化について記述したものである。

日本原子力研究所(東海駐在)：〒319－1195　茨城県那珂郡東海村白方白根２－４
*1 高エネルギー加速器研究機構
*2 東洋大学
*3 大阪大学

JAERI-Data/Code 98-007

Vectorization, Parallelization and Implementation of Quantum Molecular Dynamics Codes [QQQF, MONTEV]

Kaori KATO[*1], Tomoaki KUNUGI, Susumu KOTAKE[*2] and Masahiko SHIBAHARA[*3]

Advanced Photon Research Center
(Tokai Site)
Kansai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received January 26, 1998)

This report describes parallelization, vectorization and implementation for two simulation codes, Quantum molecular dynamics simulation code QQQF and Photon montecalro molecular dynamics simulation code MONTEV, that have been developed for the analysis of the thermalization of photon energies in the molecule or materials. QQQF has been vectorized and parallelized on Fujitsu VPP and has been implemented from VPP to Intel Paragon XP/S and parallelized. MONTEV has been implemented from VPP to Paragon and parallelized.

Keywords: Vectorization, Parallelization, VPP500, VPP300, Paragon XP/S

*1 High Energy Accelerator Research organization

*2 Toyo University

*3 Osaka Univcersity

目　　次

Contents

List of Figures and Tables

This is a blank page.

1. はじめに

日本原子力研究所（以降，原研）で開発されている光量子物質相互作用シミュレーションコードである QQQF，MONTEV に対してベクトル化，並列化，および移植を行った．各コードの詳細については２章から６章に記述し，本章ではそれらの概要を述べる．

光量子分子動力学コード QQQF に対しては，まず原研所有の計算機 Fujitsu VPP 上でオリジナル版のコードのベクトル並列化を行った．VPP はベクトル計算機であり使用可能なメモリ量も大きいという特性を活かし，データ構造の変更（使用メモリ量の増加）によるベクトル化を行った．また，VPP500 は最大１６プロセッシングエレメント（以降 PE）の同時使用が可能であるため，ベクトル化後のコードに対して粒子分割を適用した並列化（７並列）を行った．この結果，VPP300 では 4.14倍，VPP500 では2.45倍の高速化が達成された．次いで VPP から Intel Paragon への移植および並列化を行った．Paragon はスカラー型の計算機であり，１ノードの使用可能メモリ量は少ないが，最大８００ノードの同時使用が可能である．この特性を活かし，計算に必要なメモリ量の少ないオリジナル版のコードに対し，粒子分割を適用後さらに各粒子の波数空間の分割を行った．粒子分割に対しては７分割の固定であるが，位相空間に対しては２のｎ乗（ｎは１から５まで）で任意の分割を可能とした．この結果，ノード数を 4,8,16,32 と変更した場合，速度向上に関しては約 1.9， 3.6， 6.0， 8.9 ,並列化効率に関しては約 0.95, 0.9, 0.75, 0.56 が得られた．また，この7×2^n 並列の複数のプログラムを一括に処理することも可能とした．

光モンテカルロ分子動力学コード MONTEV に対してはすでに VPP でのベクトル並列化が終了しているので Paragon への移植および Paragon 上での粒子分割を適用した並列化を行った．この結果，ノード数10に対しては9.2倍，ノード数50に対しては34.4倍，ノード数100に対しては53.4倍，ノード数500に対しては73倍の性能が得られた．

2. ＶＰＰ５００におけるQQQFベクトル化

まず，光量子分子動力学コード QQQF の，ベクトル計算機 VPP 上でのベクトル化を行った．本ベクトル化により，オリジナル版に対して 2.7 倍の高速化を達成した．

2．1 コード概要

QQQFは光照射による物質の光熱変換機構を解析するためのコードである[1]．QQQF では，光が作用する物質系をイオン（原子）と最外殻の電子に分けて考え，電子に関しては時間依存の波動関数として量子力学的に取り扱い，イオンに関しては質点粒子として古典力学的に取り扱い，光の作用については光電場によるポテンシャル場の変化として扱うモデルに基づいた量子分子動力学法を用いている．

2．2 動的挙動解析

実行時における計算コスト分布を調べるため，VPP500 上での解析ツール sampler[2]を用いた動的解析を行った．動的解析より，サブルーチンschrediの演算負荷が最大であることがわかった．次に，schredi に関するベクトル化情報 [3]を採取した．Fig.2.1 に sampler での解析結果を，Fig.2.2 に schredi に関するベクトル化情報の一部を示す．動的解析結果より，サブルーチン schredi に，82%の計算コストが費やされており，最大負荷であることがわかる．またサブルーチン schredi のベクトル化情報から，最も繰り返し数の多い5重ループ（do 142）のベクトル化が，最内側ループ（do 112）のみであることがわかる．したがってここでは，サブルーチン schredi do 122 のベクトル化を行う．

2．3 ベクトル化

サブルーチン schredi の do 122 のベクトル化を妨げているのは，この部分で参照され，サブルーチン edatas2 において定義される配列 eeclo2 のデータ構造である．

本節ではプログラム記述の変更について述べる．ベクトル化のために，（1）インクルードファイルの整備，（2）配列 eeclo2 のデータ構造，（3）配列 eeclo2 の定義を行うサブルーチン edatas2 の記述，（4）配列 eeclo2 の参照を行うサブルーチン schredi の do 142 ループの記述，（5）FFT 計算サブルーチンの変更を行った．

（1） インクルードファイルの整備

インクルードファイル 'QQQ.h' を新規作成し，オリジナルの各ソースプログラム中のコモン配列・変数を格納した．また各ソースプログラム中に記述されているコモン配列・変数を削除し，代わりに 'QQQ.h' をインクルードするよう変更を行った．コモン配列の追加・削除・サイズ変更には，'QQQ.h' の更新が必要である．添字式にパラメータ kazu を用いる配列（例：xgrid）については，添字を (kazu,nxxgmax,nyygmax) から (nxxgmax,nyygmax,kazu) へ変更した．Fig.2.3 に QQQ.h の内容を示す．

（2）配列 eeclo2 データ構造の変更

common/eedat2/ において定義されている配列 eeclo2 のデータ構造を eeclo2(63,63,14) から eeclo2(32,32,32,32,14) へ変更した．common/eedat2/ は，インクルードファイル QQQ.h に記述した．

（3）サブルーチン edatas2

do 201 ループを例にあげて説明する．eeclo2 の配列要素を do 201 ループ内の各 do 変数 (i101,is11,i201,is21) で直接指定するように，①～③の変更を行った．番号 201,204,205,206,207,208,209,210,211,212,213,9103 の do ループに関しても同様の変更を行った．Fig.2.4 に変更前の do 201 ループ，Fig.2.5 に変更後の do 201 ループを示す．

①配列 eeclo2(63,63,14) の eeclo2(32,32,32,32,14) への変更

②2変数 (nxgvec,nygvec) の検出の中止

③配列 eeclo2 の要素指定方法の変更

例：eeclo2(1,nxgvec,nygvec) から eeclo2(i101,is11,i201,is21)

（4）サブルーチン schredi

本サブルーチンは，上記（3）で配列 eeclo2 に値を代入したときと同様の手順で配列要素を参照するよう変更した．また，配列要素 vclo(mmpo,nnpo,*)を定数 const に代入した．VPP500 ではこの代入を行わないと，vclo(mmpo,nnpo,*) が do 122 ループに対して定数であると見なされず，do 122 のベクトル化が行われないためである．Fig.2.6 に変更前の do 142 ループ，Fig.2.7 に変更後の do 142 ループを示す．なお，Fig.2.6, Fig.2.7 において i100 は定数である．

（5） FFT サブルーチンの変更

ファンクション ibitr 呼び出しと，変数検出の演算負荷を削減するため，FFT サブルーチンの初回実行時のみ，ファンクション ibitr 呼び出しと，変数i,c,s,k1,kln2 の値の検出を行い，値を保存するため次の①～⑤の変更を行った．他の FFT サブルーチン ryfft,invyfft,invxfft,ifftenx,iffteny に関しても同様の変更を行った．

Fig.2.8 に変更前の FFT サブルーチンの例として rxfft を示す．Fig.2.9 に変更後のrxfftを示す．

①保存用新規のコモン領域 common/rxfftcom/ の定義

②ibitr 呼び出しには「時間ステップ初回・粒子番号1」という2つの条件を付加

条件：nwstep=1 .and. nwstepe=1 .and. i100=1

③ibitr の出力をプログラム実行中保持する

④変数 i,c,s,k1,kln2 を，同名の配列（変数i のみ配列名 ibit と変更）に変更

⑤配列 c,s,k1,kln2 要素番号の更新，配列要素参照用のループカウンタ m の導入

m：ループの繰り返しに関して連続に1づつ加算される

2．4　ベクトル化の効果

Fig.2.10 にベクトル化後の sampler による動的解析結果を示す．Fig.2.11 にベクトル化後のサブルーチン schredi の do 122 ループに関するベクトル化情報を示す．これより，サブルーチン schredi の do 122 のベクトル化と schredi，ibitr の計算コストの低下を確認した．また，計算時間の測定を行い，ベクトル化前後でのプログラム性能の比較を行った．計算所要時間測定の結果より，本ベクトル化ではオリジナル版に対して 2.7 倍の高速化を達成した．

Fig.2.12 に，測定時の計算パラメータを記述するインクルードファイル prame4 の内容を示す．Table.2.1 にプログラム各部分の計算所用時間，Table.2.2 に必要メモリ量についてまとめた．なお，時間測定には，VPP500 に用意されているプログラム経過時間を測定するサブルーチン gettod[4]を使用した．

2．5　実行方法

QQQF.VPP_v.tar.Z を解凍後，ロードモジュールの作成，ＮＱＳへの投入の順に行う．QQQF.VPP_v.tar.Z には，VPP500 上でのプログラム実行に必要な，ファイル一式（ソースプログラム・インクルードファイル・Makefile 記述例・nqs 投入用シェルスクリプト記述例・入力データファイル）を，アーカイブ形式にまとめ圧縮して格納してある．Fig.2.13 に QQQF.VPP_v.tar.Z 解凍時のファイルリストを示す．

（1）　アーカイブ形式ファイルについて

・圧縮ファイルの ftp 等による転送は，バイナリモードで行う．

```
転送例  >bin                        [RET]
        >put QQQF.VPP_v.tar.Z       [RET]
```

・ファイルの解凍には，uncompress，tar コマンドを使用する．

```
解凍例  1%uncompress QQQF.VPP_v.tar.Z   [RET]
        2%tar -xvf QQQF.VPP_v.tar       [RET]
```

（2）　ロードモジュールの作成（ソースプログラムのコンパイル・リンク）

ロードモジュールの作成には，Makefile.500 を Makefile と変更して make コマンドを実行する．Fig.2.14 に Makefile の記述例を示す．テスト実行には，最適化オプションは -Oe を用いたが，Fig.2.14 において，最適化オプションは -Of としてある．テスト終了後，-Of を用いても，プログラム実行に支障のないことが判明したためである．

・ファイル名の変更

```
例    %mv Makefile.500 Makefile     [RET]
```

・ロードモジュールの作成

```
例    %make          [RET]
```

（3） プログラムの実行

プログラムの実行は，ＮＱＳ投入用シェルスクリプトを qsub コマンドによってＮＱＳへ投入して行う．Fig.2.15にＮＱＳ投入用シェルスクリプトの記述例を示す．Fig.2.15のようなシェルスクリプトを，ジョブクラス，ディレクトリ名，ファイル名等を実行時の環境に適したように変更し使用する．

・ＮＱＳへの投入
例 **%qsub** go_*sh.vec* [RET]

・プログラム動作環境

計算機	VPP500/42
OS	UXP/M(UNIX)
言語	fortran77EX/VPP
コンパイラ	f77VPP
最適化オプション	-Oe
実行時オプション	なし

2．6 まとめ

本ベクトル化の結果，サブルーチンschredi に関しては約4.5倍，プログラム全体では約2.7倍，オリジナル版コードに対して高速化された．FFT 計算サブルーチンの変更においては，増加メモリは約0.2MB，計算時間は約4.4分短縮された．サブルーチンschredi のベクトル化においては，増加メモリは約11.5MB，計算時間は約40.2分短縮された．これより，プログラム全体の所用計算時間は約71.3分から約26.7分となった．また必要メモリ量は 20MB から 135MB と約6倍になった．これは，配列 eeclo2 のデータ構造変更によって eeclo2 の大きさが 0.42MB から 112MB へ増加したためである．

```
Synthesis Information
    Count|   Percent|    VL| Name
   570325|      82.4|    33|  schredi_
    26710|       3.9|   311|  ionpo4_
    15680|       2.3|    56|  invxfft_
    15481|       2.2|    31|  rxfft_
    14985|       2.2|   959|  interpo_
    10509|       1.5|    62|  invyfft_
     9033|       1.3|    26|  ryfft_
     8724|       1.3|     -|  ibitr_
     7462|       1.1|    48|  ifftenx_
     4737|       0.7|    38|  iffteny_
     2562|       0.4|    32|  fft_
     1869|       0.3|     -|  skaku_
      854|       0.1|    32|  eneaf_
      684|       0.1|     -|  try_
      529|       0.1|    32|  elecden_
      285|       0.0|    32|  enegbe_
      247|       0.0|     -|  gid22_
      201|       0.0|    32|  edatas2_
      172|       0.0|  1024|  before1_
      157|       0.0|     -|  calcu_
      124|       0.0|     -|  after1_
       88|       0.0|  1024|  ionpre_
       81|       0.0|  1024|  nlconsp_
       67|       0.0|    32|  fconjy_
       64|       0.0|    32|  fcnjxy_
       53|       0.0|    32|  fconjx_
       47|       0.0|    32|  abspec_
       29|       0.0|     -|  _start
       28|       0.0|     -|  MAIN__
        4|       0.0|     -|  main
        1|       0.0|     -|  data_

   691792|          |    52|  TOTAL
```

Fig.2.1 Dynamic behavior of the original version of QQQF code.

```
00000020 v              do 142 ibetue= 1,kazu
00000021
00000022 v              if (ibetue .eq. i100) then goto 142
00000023                goto 142
00000024 v              end if
00000025
00000026 s              do 140 nnpo= 1,nyygmax
00000027 s2             do 130 mmpo= 1,nxxgmax
00000028 s2             do 122 jjpo= 1,nyygmax
00000029 v2             do 112 iipo= 1,nxxgmax
00000030 v2             kt2em= nelep(nstate(i100),nstate(ibetue))
00000031 v2             nxabs=iipo-mmpo
00000032 v2             nyabs=jjpo-nnpo
00000033 v2             vwcons(iipo,jjpo,i100)
00000034               & = vwcons(iipo,jjpo,i100)
00000035               & + vclot(mp,np,ibetue) * eeclo2(nxabs,nyabs,kt2em)
00000036
00000037 v2     112   continue
00000038 s2     122   continue
00000039 s2     130   continue
00000040 s      140   continue
00000041 v      142   continue
```

Fig.2.2 Vectorazied information of subroutine schredi do 142.

```
c       include  'QQQ.h'
        common /nowstep1/ nwstepe
        common /nowstep/ nwstep
c     *********    Quantum part    **********
c***********************************************************************
c       common /gridpos/ xgrid(kazu,nxxgmax,nyygmax),
c     &                  ygrid(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
        common /gridpos/ xgrid(nxxgmax,nyygmax,kazu),
     &                  ygrid(nxxgmax,nyygmax,kazu)
        common /gridp1/ xgri1(nxxgmax,nyygmax),ygri1(nxxgmax,nyygmax)
        common /gridp2/ xgri2(nxxgmax,nyygmax),ygri2(nxxgmax,nyygmax)
        common /gridp3/ xgri3(nxxgmax,nyygmax),ygri3(nxxgmax,nyygmax)
        common /gridp4/ xgri4(nxxgmax,nyygmax),ygri4(nxxgmax,nyygmax)
        common /gridp5/ xgri5(nxxgmax,nyygmax),ygri5(nxxgmax,nyygmax)
        common /gridp6/ xgri6(nxxgmax,nyygmax),ygri6(nxxgmax,nyygmax)
        common /gridp7/ xgri7(nxxgmax,nyygmax),ygri7(nxxgmax,nyygmax)
        common /gridp8/ xgri8(nxxgmax,nyygmax),ygri8(nxxgmax,nyygmax)
        common /gridp9/ xgri9(nxxgmax,nyygmax),ygri9(nxxgmax,nyygmax)
c***********************************************************************
c       common /vwavef/  vwgr(kazu,nxxgmax,nyygmax)
c       common /usovw/  usovwgr(kazu,nxxgmax,nyygmax)
c       common /vwbef/ vwbe(kazu,nxxgmax,nyygmax)
c       common /uvwbef/ uvwbe(kazu,nxxgmax,nyygmax)
c       common /vcloud/  vclo(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
        common /vwavef/  vwgr(nxxgmax,nyygmax,kazu)
        common /usovw/  usovwgr(nxxgmax,nyygmax,kazu)
        common /vwbef/ vwbe(nxxgmax,nyygmax,kazu)
        common /uvwbef/ uvwbe(nxxgmax,nyygmax,kazu)
        common /vcloud/  vclo(nxxgmax,nyygmax,kazu)
        common /nocon/   nlconst
        common /oeig/   eigen
        common /gdelt/ vxdelt,vydelt,xsyumd,ysyumd
        common /vtcloud/ tclo(kazu,nxxgmax,nyygmax)
c***********************************************************************
c       common /ukewa/ resum(kazu,nxxgmax,nyygmax),
c     &                usosum(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
        common /ukewa/ resum(nxxgmax,nyygmax,kazu),
     &                usosum(nxxgmax,nyygmax,kazu)
        common /gyabi/ ngyas(nxxgmax)
        common /cscs/ ccos(nxxgmax/2,nsisuu+1),csin(nxxgmax/2,nsisuu+1)
c***********************************************************************
c       common /souvw/  spcwf(kazu,nxxgmax,nyygmax),
c     &                 upcwf(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
        common /souvw/  spcwf(nxxgmax,nyygmax,kazu),
     &                 upcwf(nxxgmax,nyygmax,kazu)
        common /spdata/ rspda(nspec),uspda(nspec),njudge,nsum
        common /spda44/ rspda4(nspec),uspda4(nspec)
c***********************************************************************
c       common /penecal/ potene(kazu,nxxgmax,nyygmax),
c     &                  usopene(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
        common /penecal/ potene(nxxgmax,nyygmax,kazu),
     &                  usopene(nxxgmax,nyygmax,kazu)
```

Fig.2.3 Contents of the include file QQQ.h. (1/2)

```
c*********************************************************************
c      common /unene/ runene(kazu,nxxgmax,nyygmax),
c     &               uunen(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
       common /unene/ runene(nxxgmax,nyygmax,kazu),
     &               uunen(nxxgmax,nyygmax,kazu)

c    ***** dipole moment
       common /dipmo/ xdipm,ydipm,ntasu
c*********************************************************************
c      common /preio/ dxxgg(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
       common /preio/ dxxgg(nxxgmax,nyygmax,kazu)

c    ***** average density of wave func.
       common /avewv/ aveden(kazu,nxxgmax,nyygmax)
c*********************************************************************
c      common /eedat2/ eeclo2(14,63,63)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
c      common /eedat2/ eeclo2(63,63,14)
       common /eedat2/ eeclo2(32,32,32,32,14)
c*********************************************************************

c     ************    MD part    ************
       common /pppo/ pppx(10),pppy(10)
       common /vvvo/ vvvx(10),vvvy(10)
       common /aaao/ aaax(10),aaay(10)
       common /iopo/ xemc,yemc,potexe,poteki
       common /enep1bak/ enep1b,aaaxb,aaayb,bbbxb(10),bbbyb(10)
       common /elecon/ econst
       common /finate/ vaf(kazu,nfine,nfine),
     &       vafpx(kazu,nfine,nfine),vafpy(kazu,nfine,nfine)
       common /d4012/ dundou(kazu,maxtep)
       common /d4013/ denepii(kazu,maxtep),denepei(kazu,maxtep)
       common /d03/ dpppx(kazu,maxtep),dpppy(kazu,maxtep),
     &              dvvvx(kazu,maxtep),dvvvy(kazu,maxtep),
     &              deeex(kazu,maxtep),deeey(kazu,maxtep),
     &              dpxre(kazu,maxtep),dpyre(kazu,maxtep)
       common /d34/ ddnti(nttsp),dsoukan(nttsp),dusok(nttsp)
       common /d38/ denergy(kazu,nttsp),denekin(kazu,nttsp),
     &              denepot(kazu,nttsp)
       common /d14/ dvwgr(nvti,kazu,nsku),dusovwgr(nvti,kazu,nsku)
       common /d15/ dvclo(nvti,kazu,nsku)
       common /eleppp/ elepx(kazu),elepy(kazu),dipox(kazu),dipoy(kazu)
       common /elefor/ eleff(1729),eleer
       common /wwwm/ wavel
       common /nsta/ nstate(kazu)
       common /elepco/ nelep(9,9)
       common /bbvar/ hhhh,hahh,hahhp,hh23p,xceme,yceme
       common /ocon/   anlcst
c*********************************************************************
c      common /muul1/ vwcons(kazu,nxxgmax,nyygmax)
c      common /muul2/ emcsumd(kazu,nxxgmax,nyygmax)
c****************** Modified 1996.04.30 by kaori@love.tokai.jaeri.go.jp
       common /muul1/ vwcons(nxxgmax,nyygmax,kazu)
       common /muul2/ emcsumd(nxxgmax,nyygmax,kazu)
```

Fig.2.3 Contents of the new include file QQQ.h. (2/2)

```
      subroutine edatas2
            :
      do  201 i201= 1,nyygmax
      do  101 i101= 1,nxxgmax
      do 1201 is21= 1,nyygmax
      do 1101 is11= 1,nxxgmax
         nxgvec= i101 -is11  + 32
         nygvec= i201 -is21  + 32

         xgs1= xgri1(i101,i201)
     &       -xgri1(is11,is21)
         ygs1= ygri1(i101,i201)
     &       -ygri1(is11,is21)
         if (xgs1 .eq. 0.00D0) then
         if (ygs1 .eq. 0.00D0) then
         eeclo2(1,nxgvec,nygvec) = 1.000D0/handa
         goto 1101
         endif
         endif

         tokyo= dsqrt(xgs1**2+ygs1**2)
         eeclo2(1,nxgvec,nygvec) = 1.000D0/tokyo

1101  continue
1201  continue
 101  continue
 201  continue
            :
      return
      end
```

Fig.2.4 Original subroutine edatas2.

```
      subroutine edatas2
            :
      do  201 i201= 1,nyygmax
      do  101 i101= 1,nxxgmax
      do 1201 is21= 1,nyygmax
      do 1101 is11= 1,nxxgmax

         xgs1= xgri1(i101,i201)-xgri1(is11,is21)
         ygs1= ygri1(i101,i201)-ygri1(is11,is21)

        if (xgs1.eq.0.00D0 and. ygs1.eq.0.00D0) then
          eeclo2(is11,is21,i101,i201,1)=handa
          goto 1101
        endif

         tokyo= dsqrt(xgs1**2+ygs1**2)
         eeclo2(is11,is21,i101,i201,1)=1.000D0/tokyo

1101  continue
1201  continue
 101  continue
 201  continue
            :
      return
      end
```

Fig.2.5 Modified subroutine edatas2.

```
      subroutine schredi(i100)
                :
      do 142 ib= 1,kazu*npe
      if (ib .eq. i100) goto 142

      kt2em= nelep(nstatet(i100),nstatet(ib))
      do 140 np = 1,nyygmax
      do 130 mp = 1,nxxgmax
      do 122 jp = 1,nyygmax
      do 112 ip = 1,nxxgmax

          nxabs= ip - mp + 32
          nyabs= jp - np + 32

          vwcons(ip,jp,i100) = vwcons(ip,jp,i100)
     & + vclot(mp,np,ibetue)*eeclo2(nxabs,nyabs,kt2em)

  112 continue
  122 continue
  130 continue
  140 continue
  142 continue
                :
      return
      end
```

Fig.2.6 Original do 142 loops in subroutine schredi

```
      subroutine schredi(i100)
                :
      do 142 ibetue= 1,kazu
      if (ibetue .eq. i100) goto 142

      kt2em = nelep(nstate(i100),nstate(ibetue))
      do 140 nnpo= 1,nyygmax
      do 130 mmpo= 1,nxxgmax
         const=vclo(mmpo,nnpo,ibetue)
      do 122 jjpo= 1,nyygmax
      do 112 iipo= 1,nxxgmax

      vwcons(iipo,jjpo,i100) = vwcons(iipo,jjpo,i100)+ const
     & * eeclo2(iipo,jjpo,mmpo,nnpo,kt2em)

  112 continue
  122 continue
  130 continue
  140 continue
  142 continue
                :
      return
      end
```

Fig.2.7 Modified do 142 loops in subroutine schredi

```
      subroutine rxfft(i100)
      implicit double precision(a-h,o-z)
      include'prame4'
      common /nowstep1/ nwstepe
      common /ukewa/ resum(kazu,nxxgmax,nyygmax),
     &               usosum(kazu,nxxgmax,nyygmax)

      n= nxxgmax
      nu= nsisuu
      n2=n/2
      nu1=nu-1
      k=0
cc
      nu1=nu-1
      n2=n/2
      do 100 l=1,nu
  102   do 101 i=1,n2
          p=ibitr(k/(2**nu1),nu)
          arg=6.283185*p/float(n)
          c=cos(arg)
          s=sin(arg)
          k1=k+1
          k1n2=k1+n2
        do 1000 lly=1,nyygmax
          treal=resum(i100,k1n2,lly)*c + usosum(i100,k1n2,lly)*s
          timag=usosum(i100,k1n2,lly)*c - resum(i100,k1n2,lly)*s
          resum(i100,k1n2,lly)=resum(i100,k1,lly)-treal
          usosum(i100,k1n2,lly)=usosum(i100,k1,lly)-timag
          resum(i100,k1,lly)= resum(i100,k1,lly)+treal
          usosum(i100,k1,lly)= usosum(i100,k1,lly)+timag
 1000   continue
  101     k=k+1
        k=k+n2
        if(k.lt.n) goto 102
        k=0
        nu1=nu1-1
        n2=n2/2
  100   continue
  999   format('l,lly,i,nu1',4i6)
cc
      do 103 k=1,n
        i=ibitr(k-1,nu)+1
        if(i.le.k) goto 103
      do 2000 lly1=1,nyygmax
        treal=resum(i100,k,lly1)
        timag=usosum(i100,k,lly1)
        resum(i100,k,lly1)=resum(i100,i,lly1)
        usosum(i100,k,lly1)=usosum(i100,i,lly1)
        resum(i100,i,lly1)=treal
        usosum(i100,i,lly1)=timag
 2000 continue
  103 continue

      return
      end
```

Fig.2.8 Original FFT subroutine rxfft

```
      subroutine rxfft(i100)
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
      common/rxfftcom/ibit(nxxgmax)
     &                ,c(nxxgmax/2,nsisuu,nxxgmax/2)
     &                ,s(nxxgmax/2,nsisuu,nxxgmax/2)
     &                ,kl(nxxgmax/2,nsisuu,nxxgmax/2)
     &                ,kln2(nxxgmax/2,nsisuu,nxxgmax/2)

        if( nwstep.eq.1 .and. nwstepe.eq.1
     &                  .and. i100.eq.1 ) then
        n= nxxgmax
        nu= nsisuu
        k=0
        nu1=nu-1
        n2=n/2
        do 100 l=1,nu
        m=0
  102   do 101 i=1,n2
        m=m+1
          p=ibitr(k/(2**nu1),nu)
          arg=6.283185*p/float(n)
          c(i,l,m)=cos(arg)
          s(i,l,m)=sin(arg)
          kl(i,l,m)=k+1
          kln2(i,l,m)=kl(i,l,m)+n2
  101   k=k+1
        k=k+n2
        if(k.lt.n) goto 102
        k=0
        nu1=nu1-1
        n2=n2/2
  100   continue

      do 103 k=1,n
        ibit(k)=ibitr(k-1,nu)+1
  103 continue

      end if
           :
           :
      return
      end
```

Fig.2.9 Modified FFT subroutine rxfft.(1/2)

```
      subroutine rxfft(i100)
           :
           :
      n= nxxgmax
      nu= nsisuu
      k=0
      nu1=nu-1
      n2=n/2

      do 200 l=1,nu
      m=0
202   do 201 i=1,n2
      m=m+1
    do 1000 lly=1,nyygmax
        treal= resum(kln2(i,l,m),lly,i100)*c(i,l,m)
   &          + ususum(kln2(i,l,m),lly,i100)*s(i,l,m)
        timag= ususum(kln2(i,l,m),lly,i100)*c(i,l,m)
   &          - resum(kln2(i,l,m),lly,i100)*s(i,l,m)
        resum(kln2(i,l,m),lly,i100) =resum(kl(i,l,m),lly,i100)-treal
        ususum(kln2(i,l,m),lly,i100)=ususum(kl(i,l,m),lly,i100)-timag
        resum(kl(i,l,m),lly,i100)   = resum(kl(i,l,m),lly,i100)+treal
        ususum(kl(i,l,m),lly,i100)  = ususum(kl(i,l,m),lly,i100)+timag
1000  continue

 201  k=k+1
      k=k+n2
      if(k.lt.n) goto 202
      k=0
      nu1=nu1-1
      n2=n2/2
 200  continue

    do 203 k=1,n
      if(ibit(k).le.k) goto 203
    do 2000 lly1=1,nyygmax
      treal=resum(k,lly1,i100)
      timag=ususum(k,lly1,i100)
      resum(k,lly1,i100) =resum(ibit(k),lly1,i100)
      ususum(k,lly1,i100)=ususum(ibit(k),lly1,i100)
      resum(ibit(k),lly1,i100) =treal
      ususum(ibit(k),lly1,i100)=timag
2000 continue
 203 continue

    return
    end
```

Fig.2.9 Modified FFT subroutine rxfft.(2/2)

```
Synthesis Information (after)
    Count|   Percent|    VL| Name
    68372|      45.1|  1022|  schredi_
    26531|      17.5|   311|  ionpo4_
    15801|      10.4|   564|  interpo_
    10185|       6.7|    37|  invxfft_
     9116|       6.0|    32|  rxfft_
     5403|       3.6|    41|  ifftenx_
     3405|       2.2|    49|  invyfft_
     3257|       2.1|    32|  ryfft_
     2503|       1.6|    38|  fft_
     2088|       1.4|    67|  iffteny_
     2014|       1.3|     -|  skaku_
      635|       0.4|     -|  try_
      467|       0.3|    32|  eneaf_
      390|       0.3|    32|  elecden_
      250|       0.2|    32|  edatas2_
      246|       0.2|     -|  gid22_
      201|       0.1|    32|  enegbe_
      161|       0.1|   984|  before1_
      155|       0.1|     -|  calcu_
      118|       0.1|  1024|  after1_
       75|       0.0|    32|  fconjy_
       74|       0.0|    32|  fconjx_
       64|       0.0|    32|  fcnjxy_
       64|       0.0|  1024|  ionpre_
       58|       0.0|    32|  abspec_
       47|       0.0|  1024|  nlconsp_
       26|       0.0|     -|  MAIN__
       10|       0.0|     -|  _start
        4|       0.0|     -|  data_
        2|       0.0|     -|  fftho_
   151722|          |   655|  TOTAL
```

Fig.2.10 Dynamic behavior of the modified version of QQQF code.

```
00000022 s            do 142 ibetue= 1,kazu
00000023
00000024 v            if (ibetue .eq. i100) goto 142
00000025
00000026 m            kt2em= nelep(nstate(i100),nstate(ibetue))
00000027 s            do 140 nnpo= 1,nyygmax
00000028 s2           do 130 mmpo= 1,nxxgmax
00000029 s2              const=vclo(mmpo,nnpo,ibetue)
00000030 v2           do 122 jjpo= 1,nyygmax
00000031 v2           do 112 iipo= 1,nxxgmax
00000032
00000033 v2           vwcons(iipo,jjpo,i100)
00000034             & = vwcons(iipo,jjpo,i100)
00000035             & + const * eeclo2(iipo,jjpo,mmpo,nnpo,kt2em)
00000036
00000037 v2     112  continue
00000038 v2     122  continue
00000039 s2     130  continue
00000040 s      140  continue
00000041 v      142  continue
```

Fig.2.11 Vectorized information of modified subroutine schredi do 142.

```
ccccccccccccccccccccccccccccccccccccc
c   include prame4
ccccccccccccccccccccccccccccccccccccc
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          maxtepe=20,kazu=7,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1,ndatas=20,ndata1=500, akousi=5.333D-10,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat= 0.1000000D-16,gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,hbar= 1.055D-34,
     &          epusi0=0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida= 1024, naida= 10,
     &          kstart= 1,nspec= 32768,mspec= 15,
     &           situk=6.4931916D-26,situxe=6.4931916D-26,
     &           situka=6.4931916D-26,sigkk=4.00D-10,
     &           epukk=1.625D-21*100.00D0,
     &           sigke=1.000D0*sigkk,epuke=epukk/500.0D0,
     &           cutke= 1.0000D0,boltu= 1.381D-23,
     &           nfine=311, nggai=10
     &           )
```

Fig.2.12 Contents of the include file prame4.

```
QQQF.vpp_v/ ——— *.f, includefile,makefile
    ├── SH/
    └── DATA/
```

```
QQQF.vpp_v/
Makefile.300  vpp300 ベクトル実行用 Makefile 記述例
Makefile.500  vpp500 ベクトル実行用 Makefile 記述例
QQQ.h     インクルードファイル (配列宣言など)
prame4    インクルードファイル (パラメータファイル)

*.f     以下, ソースファイル一覧
abspec.f  after.f   after1.f  allset.f  before.f  before1.f
calcu.f   clear.f   conju.f   convr.f   data.f    datass.f
edatas2.f elecden.f eneaf.f   enegbe.f  fcnjxy.f  fcnvrx.f
fcnvry.f  fconjx.f  fconjy.f  fft.f     fftho.f   frcnvx.f
frcnvy.f  gid22.f   grids.f   ibitr.f   ifftenx.f iffteny.f
interpo.f invxfft.f invyfft.f ionbou.f  ionpo4.f  ionpre.f
lasdat.f  nlconho.f nlconsp.f qqq.f     rconv.f   rxfft.f
ryfft.f   schdiho.f schredi.f skaku.f   try.f
```

```
SH/
go_sh.vec    nqs 投入用シェルスクリプト記述例
```

```
DATA/
stakxe.data  unit01 よりの入力データ
stawv.data   unit37 よりの入力データ
namlist      unit05 ネームリストファイル
```

Fig.2.13 File lists of the QQQF.VPP_v.tar.Z.

```
#
# Center Sample Makefile
#                Write Day : Wed May  1 09:03:44 1996
#

F77      = f77vpp
MAKE     = Makefile
OPT      = -Ud -Of -Ps
LIBES    =
INCLUDES = prame4 QQQ.h
TERGET   = a.out

OBJS = \
       qqq.o  abspec.o  after.o  after1.o  allset.o\
       before.o  before1.o  calcu.o  clear.o  conju.o\
       convr.o  data.o  datass.o  edatas2.o  elecden.o\
       eneaf.o  enegbe.o  fcnjxy.o  fcnvrx.o  fcnvry.o\
       fconjx.o  fconjy.o  fft.o  fftho.o  gid22.o\
       frcnvx.o  frcnvy.o  grids.o  ifftenx.o  iffteny.o\
       interpo.o  invxfft.o  invyfft.o  ionbou.o  ionpo4.o\
       ionpre.o  lasdat.o  nlconho.o  nlconsp.o  rconv.o\
       rxfft.o  ryfft.o  schredi.o  skaku.o\
       try.o  ibitr.o

.f.o :
 $(F77) $(OPT) $(INCLUDES) -c $<

all :$(OBJS)
 $(F77) -o $(TERGET) $(OPT) $(OBJS) $(LIBES)

clean :
 rm -f $(OBJS)
```

Fig.2.14 Makefile for making load-module.

```
#!/bin/csh -f
#################################################
#@$-q  vppm      # set queue class
#@$-lt 60:00      # set CPU time limit to XX:XX (mm:ss)
#@$-C  QQQF     # set code name to XXXXX
#@$-eo
#################################################
cd $HOME/QQQF

# read data
setenv fu01 $HOME/QQQF/wkvfl/stakxe.data
setenv fu37 $HOME/QQQF/wkvfl/stawv.data

setenv fu02 $HOME/QQQF/wkvfl/sscc_vec2/tempa.data
setenv fu03 $HOME/QQQF/wkvfl/sscc_vec2/molpv.data

#  check wave.data last step
setenv fu14 $HOME/QQQF/wkvfl/sscc_vec2/wave.data.vec2
setenv fu15 $HOME/QQQF/wkvfl/sscc_vec2/tmitu.data.vec2

setenv fu16 $HOME/QQQF/wkvfl/sscc_vec2/mitu.data
setenv fu34 $HOME/QQQF/wkvfl/sscc_vec2/spec.data

#  check enevw.data every step
setenv fu38 $HOME/QQQF/wkvfl/sscc_vec2/enevw.data.vec2

setenv fu39 $HOME/QQQF/wkvfl/sscc_vec2/jyusin.data
setenv fu40 $HOME/QQQF/wkvfl/sscc_vec2/enegg1.data.vec
setenv fu41 $HOME/QQQF/wkvfl/sscc_vec2/enegg2.data.vec
setenv fu42 $HOME/QQQF/wkvfl/sscc_vec2/enegg3.data.vec
setenv fu43 $HOME/QQQF/wkvfl/sscc_vec2/enegg4.data.vec
setenv fu44 $HOME/QQQF/wkvfl/sscc_vec2/enegg5.data.vec
setenv fu45 $HOME/QQQF/wkvfl/sscc_vec2/enegg6.data.vec
setenv fu46 $HOME/QQQF/wkvfl/sscc_vec2/enegg7.data.vec
setenv fu47 $HOME/QQQF/wkvfl/sscc_vec2/disou.data
setenv fu48 $HOME/QQQF/wkvfl/sscc_vec2/avewv.data
setenv fu49 $HOME/QQQF/wkvfl/sscc_vec2/enetest.data
setenv fu50 $HOME/QQQF/wkvfl/sscc_vec2/force.data
setenv fu52 $HOME/QQQF/wkvfl/sscc_vec2/coef.dddd.vec
setenv FOR51 $HOME/QQQF/wkvfl/sscc_vec2/avf.100

a.out < namlist
```

Fig.2.15 Shell script for vector execution.

Table.2.1 Cpu time of original version and modified version.

手続き名	ベクトル化前	ベクトル化後
schredi	0.3115e+04 sec	0.6977e+03 sec
全体	0.4279e+04 sec	0.1606e+04 sec

Table.2.2 Memory size

ベクトル化前	22.4 MB
ベクトル化後	134.5 MB

3. ＶＰＰ３００におけるQQQF並列化

ＶＰＰ５００上で実行されていた，量子分子動力学コードＱＱＱＦをＶＰＰ３００へ移植し，複数台のプロセッサエレメント（以降ＰＥと記す）を用いて，並列実行できるよう変更した．なお，プログラムはベクトル化後のものを用いている．

3．1 ＶＰＰ３００への移植

ＶＰＰ３００，ＶＰＰ５００では，コンパイラ，コンパイル・リンクオプションが異なるが，移植に際してプログラム記述の変更はない．

3．2 プログラム解析

gettod によるプログラム各部分の計算所要時間の測定結果，最も高負荷である手続きは，サブルーチン try の do 45 であり，時間発展計算部分の72%を占めている．なおサブルーチン try の do 45 は，それぞれ独立に計算を行える７回の繰り返し計算である．

3．3 並列化

本並列化では，１電子・イオン（以降，粒子と記す）に関する計算を行うＰＥを，７台並列に動作させる．つまり，１ＰＥは１粒子関する情報の所有と更新を行う．このため，各ＰＥの識別番号と，各粒子の識別番号を対応させる．また，全粒子に関する情報が必要な場合は，ＰＥ間通信を行う．本並列化において，分割ローカル配列[5]は使用せず，配列は全て重複ローカル配列[5]として使用する．以降，特に断らず重複ローカル配列を配列と記す．並列化にあたり，各ソースプログラムには xocl 文[5] を挿入した．

（1） インクルードファイル prame4

並列プログラムに使用するＰＥ台数を指定するパラメータ npe を与える．また，粒子数を指定するパラメータ kazu を７から１へ変更した．Fig.3.1 に変更前のインクルードファイル prame4 を示す．Fig.3.2 に変更後の内容を示す．

（2） メインプログラム

プロセッサグループの宣言，並列プロセス生成・終了の指示および変数 mynode に idvproc[5]より取得したＰＥ識別番号，nodesに使用ノード数を与えた．mynode，nodes はコモン変数として/paralell/ に格納した．Fig.3.3 に変更前のメインプログラム，Fig.3.4 に変更後の概要を示す．do 1091 ループはサブルーチン calcu 内に移動した．

（3） サブルーチン allset

vpp では，parallel region 内での入出力は，マスタＰＥのみが行う．入力したデータを全ＰＥに配置するには，マスタＰＥがグローバル空間[5]への入力を行い（Fig.3.6 do 2），この後各ＰＥがグローバル領域より自己のローカルメモリ領域上にコピーする（Fig.3.6 の do 740）という方法を採用した．Fig.3.5 にサブルーチン allset の変更前の概要，Fig.3.6 に変更後の概要を示す．変更内容は次の通りである．

①コモン文の追加（/paralell/）

②作業グローバル配列[5]の追加（wrk1g,wrk2g, wrk3g,wrk4g）

③do 2 ループの回転数の変更（kazu から kazu*npe）

④入力方法

（4） サブルーチン gid22

本サブルーチンにおけるデータ入力も parallel region 内のものである．上記(3)と同様の方法で入力データを各ＰＥに配置する．ただし，入力データを格納する配列(vwgr,vclo,usovwgr,uvwbe)が，元の７分の１の大きさの重複ローカル配列となったのに対し，入力データは元の大きさのままである．このため各入力データと同じ大きさの作業グローバル配列(wrk1g,wrk2g,wrk3g,wrk4g)を用意し，これに入力を代入(do 50) した後，各ＰＥとも作業グローバル配列から必要なデータを抽出して，配列 (vwgr, vclo,usovwgr,uvwbe)に代入するという方法を採用した．

本サブルーチンにおける変更は，次の４点である．Fig.3.7 にサブルーチン gid22 の変更前の概要，Fig.3.8 に変更後の概要を示す．

①コモン文の追加（/paralell/）

②作業グローバル配列の追加（wrk1g,wrk2g,wrk3g,wrk4g）

③do 50 ループの回転数の変更（kazu からkazu*npe）

④入力方法の変更

（5） サブルーチン try

サブルーチン try から呼ばれるサブルーチン schredi は，配列 nstate,vclo について全粒子のデータを必要とする．ところが，本並列化において１ＰＥが持つのは，１粒子のデータのみである．このため，全粒子のデータを参照するには，ＰＥ間通信が必要となる．この場合，本並列化では総和計算を用いて全粒子に関するデータを各ＰＥに配置することにした．したがって，配列 nstate,vclo に関する全粒子のデータは，本サブルーチン try において総和計算を用いて作成し，コモン領域 /schredicom/ を用いてサブルーチン schredi に引き渡すことにした．

本サブルーチンにおける変更点は，①②の２点である．また，配列 vclot を例に取り総和計算の手順を③～⑥に説明する．配列 nstate に関しても同様の手順を用いた．Fig.3.9 にサブルーチン try の変更前の概要，Fig.3.10 に変更後の概要を示す．

①コモン文の(/paralell/,/schredicom/)追加
②重複ローカル配列 nstatet,vclot の追加，総和計算
③重複ローカル配列 vclo(*)の７倍の大きさの作業用重複ローカル配列
vclot(*,7) を各ＰＥ上に用意し，ゼロクリア (do 440) する．
④回転数がnpeであるダミーループ(do 500)を作成し，各ＰＥに spread 分割し
割り当てる．
⑤spread 分割されたループ内では，番号ipe のＰＥが vclot(*,ipe)=vclo(*)
のように，作業配列の各要素に配列を格納 (do 540) する．
⑥格納終了後，spraed sum を用いて vclot(*,ipe) の全ＰＥ間での総和を取る．

（６） サブルーチン schredi

本サブルーチンにおいては，以下５点の変更を行った．Fig.3.11 に変更前の概要，Fig.3.12 に変更後の概要を示す．
①コモン文の追加 (/paralell/,/schredicom/)
②配列名の変更 (nstate から nstatet および，vclo から vclot)
③do 142, do 989 ループの回転数の変更 (kazu から kazu*npe)
④配列 nstatet,pppx の参照される配列要素番号の変更 (i100 から is)
⑤do 142 ループでの判別式において変数 ibetue と比較される変数の変更
(i100 から is)

（７） サブルーチン calcu

本サブルーチンにおいても，３．３．５と同様の理由で他ＰＥが持つ粒子のデータを参照するため，３．３．５と同様に重複ローカル配列 vaftmp,vafpxtmp,vafpytmp の総和計算を行う．本サブルーチンにおける変更は，次の９点である．Fig.3.13 に変更前のサブルーチン calcu 概要，Fig.3.14 に変更後の概要を示す．
①コモン文の追加(/paralell/,/ionpo4com/)
②作業配列の追加(aaaxt,aaayt,vvvxt,vvvyt,pppxt,pppyt)
③配列dipoxtmpの総和演算(do 7750,do 77 00)
④do 1000 ループの回転数の変更(kazuからkazu*npe)
⑤do 1000 ループでの判別式において変数 imoll と比較される変数の変更
(ionum から is)
⑥do 1000ループにおいて変数 t126 の定義に使用する配列名の変更
(dipox からdipoxtmp)および配列要素番号の変更(ionum から is)
⑦コモン領域/ionpo4com/を用いて，サブルーチンionpo4に渡す
配列vaftmp, vafpxtmp ,vafpytmp の作成(do 750,do 700)
⑧do 1100 での配列 aaax,aaay,vvvx,vvvy,pppx,pppyの定義参照における
配列要素番号の変更(i11からis)
⑨配列aaaxt,aaayt,vvvxt,vvvyt,pppxt,pppyt の総和演算 (do 960,do 670)

（8） サブルーチン ionpo4

本サブルーチンにおける変更は，次の５点である．Fig.3.15 に変更前のサブルーチン ionpo4 概要，Fig.3.16 に変更後の概要を示す．

①コモン文の追加(/paralell/,/ionpo4com/)

②配列 pppx,pppy の参照における配列要素番号の変更(i100 から is)

③do 300 ループの回転数の変更(kazu からkazu*npe)

④配列名の変更(vafpx,vafpy,vafからvafpxtmp,vafpytmp,vaftmp)

⑤変数名の変更(xmec,ymec,poteki から zxmec,zymec,zpoteki)

（9） サブルーチン skaku

本並列化では１粒子分の計算を行うＰＥを粒子数台並列に動作させ，また各ＰＥ上には１粒子分のデータを重複ローカルデータとして配置するようにした．全粒子に関する計算結果を出力するために，サブルーチン skaku において全粒子に関する重複ローカル配列を全ＰＥ上に配置し，これをマスタＰＥより出力することにした．なお，全粒子分のデータの作成には，３．３．５と同様の方法を用いた．本サブルーチンにおける変更は，コモン文(/paralell/) の追加および作業配列の追加・総和計算である．Fig.3.17 に変更前のサブルーチンskaku概要，Fig.3.18 に変更後の概要を示す．

3．4　並列化の効果

並列化効果の評価を行う．インクルードファイル prame4 に記述されている時間ステップパラメータは，maxtepe=20,maxtep=500，逐次実行においては粒子数は kazu=7，並列実行においては粒子数は kazu=1，使用ＰＥ数は npe=7 とした．Table.3.1 にプログラム各部分の計算所用時間，Table.3.2 に必要メモリ量についてまとめた．なお，時間測定には，vpp300に用意されているプログラム経過時間を測定するサブルーチン gettod を使用した．

７ＰＥを使用した本並列化によって，プログラム全体は 4.1 倍高速化された．全計算所要時間の 37% が通信に費やされており，通信コストが非常に高いために台数分の効果は得られなかった．

3．5　実行方法

QQQF.vpp_p.tar.Z を解凍後，ロードモジュールの作成，ＮＱＳへの投入の順に行う．QQQF.vpp_p.tar.Z には，vpp300 上でのプログラム実行に必要な，ファイル一式（ソースプログラム・インクルードファイル・Makefile 記述例・nqs 投入用シェルスクリプト記述例・入力データファイル）を，アーカイブ形式にまとめ圧縮して格納してある．QQQF.vpp_p.tar.Z 解凍時のファイルリストを Fig.3.19 に 示す．

（1）　アーカイブ形式ファイルについて

・圧縮ファイルの ftp 等による転送は，バイナリモードで行う．

```
転送例  >bin                          [RET]
        >put QQQF.vpp_p.tar.Z  [RET]
```

・ファイルの解凍には，uncompress，tar コマンドを使用する．

```
解凍例  1%uncompress QQQF.vpp_p.tar.Z   [RET]
        2%tar -xvf QQQF.vpp_p.tar       [RET]
```

（2）　ロードモジュールの作成（ソースプログラムのコンパイル・リンク）

ロードモジュールの作成には，Makefile.300 を Makefile と変更して make コマンドを実行する．Fig.3.20 に Makefile の記述例を示す．

・ファイル名の変更

```
例    %mv Makefile.300 Makefile    [RET]
```

・ロードモジュールの作成

```
例    %make          [RET]
```

(3) プログラムの実行

プログラムの実行は，ＮＱＳ投入用シェルスクリプトを qsub コマンドによってＮＱＳへ投入して行う．Fig.3.21にＮＱＳ投入用シェルスクリプトの記述例を示す．Fig.3.21のようなシェルスクリプトを，ジョブクラス，ディレクトリ名，ファイル名等を実行時の環境に適したように変更し使用する．

・ＮＱＳへの投入
例　**%qsub** *go_h.p* [RET]

・テスト計算に用いたプログラム動作環境

計算機	vpp300/12
OS	UXP/V(UNIX)
言語	fortran90/VPP
コンパイラ	frt
最適化オプション	-Oe
実行時オプション	なし
使用プロセッサ数	逐次実行時１，並列実行時７

3．6 まとめ

本並列化によって，サブルーチン try に関しては 7.3 倍高速化され，計算所要時間は約12.6 分短縮された．サブルーチン calcu に関しては約 3.5 倍高速化され，計算所要時間は約 3.9 分短縮されたが，計算所要時間の約 37% が通信に費やされており，通信コストが非常に高い．vpp500 でのテスト実行では，通信速度が vpp300 より遅いためサブルーチン calcu の高速化は達成されなかった．プログラム全体に関しては，約 4.1 倍高速化され，プログラム全体の計算所要時間は20.6 分から 5.0 分へ減少した．必要メモリ量の変化に関しては，新たに作業配列を宣言したため減少しなかった．また並列化前後の計算結果については，完全な一致は得られなかった．これは，演算順序の変更などによるものと考えられる．

```
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          maxtepe=20,kazu=7,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1,ndatas=20,ndata1=500, akousi=5.333D-10,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat= 0.1000000D-16,gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,hbar= 1.055D-34,
     &          epusi0=0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida= 1024, naida= 10,
     &          kstart= 1,nspec= 32768,mspec= 15,
     &          situk=6.4931916D-26,situxe=6.4931916D-26,
     &          situka=6.4931916D-26,sigkk=4.00D-10,
     &          epukk=1.625D-21*100.00D0,
     &          sigke=1.000D0*sigkk,epuke=epukk/500.0D0,
     &          cutke= 1.0000D0,boltu= 1.381D-23,
     &           nfine=311, nggai=10
     &           )
```

Fig.3.1 Original include file prame4.

```
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          npe=7,
c    &          maxtepe=20,kazu=7,nttsp=10000,nsku=1024,nvti=1,
     &          maxtepe=20,kazu=1,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1,  ndatas=20,ndata1=500, akousi=5.333D-10,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat = 0.1000000D-16,  gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,  hbar= 1.055D-34,
     &          epusi0 = 0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida  = 1024,           naida= 10,
     &          kstart = 1, nspec= 32768,mspec= 15,
     &          situk  = 6.4931916D-26,  situxe=6.4931916D-26,
     &          situka = 6.4931916D-26,  sigkk=4.00D-10,
     &          epukk  = 1.625D-21*100.00D0,
     &          sigke  = 1.000D0*sigkk,  epuke=epukk/500.0D0,
     &          cutke  = 1.0000D0,       boltu= 1.381D-23,
     &          nfine  = 311,            nggai=10
     &           )
```

Fig.3.2 Modified include file prame4.

```
      program QQQ
      implicit
     & double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

        call clear
        call allset
        nwstep=1
        call lasdat
        call gid22
        call edatas2
            :
c--------------------------------
      do 100 i=1,maxtep
        nwstep=i
        call try
        call calcu
  888   call datass
      do 1091 iiku2= 1,kazu
            :
 1091 continue
  100 continue
c--------------------------------
      call skaku

      stop
      end
```

Fig.3.3 Original main program.

```
      program QQQ
      implicit
     & double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
      common /paralell/ mynode,nodes

!xocl processor p(npe)
!xocl parallel region
      mynode=idvproc()
      nodes=npe
c--------------------------------
        call clear
        call allset
        nwstep=1
        call lasdat
        call gid22
        call edatas2
            :
c--------------------------------
      do 100 i=1,maxtep
        nwstep=i
        call try
        call calcu
  888   call datass

  100 continue
c--------------------------------
      call skaku

!xocl end parallel

      stop
      end
```

Fig.3.4 Modified main program.

```
      subroutine allset
                :
      do 2 kkkazu= 1,kazu
        read(1,9000)n1,xxx,yyy,vxxx,vyyy
        pppx(kkkazu)=xxx
        pppy(kkkazu)=yyy
        vvvx(kkkazu)=vxxx
        vvvy(kkkazu)=vyyy
    2 continue

      return
      end
```

Fig.3.5 Original subroutine allset.

```
      subroutine allset
                :
      common /paralell/ mynode,nodes
      dimension wrk1g(kazu*npe),wrk2g(kazu*npe)
     &          ,wrk3g(kazu*npe),wrk4g(kazu*npe)

!xocl processor p(npe)
!xocl subprocessor sp(npe)=p(1:npe)
!xocl index  partition ip=(sp,index=1:npe*kazu,part=band)
!xocl global wrk1g(/ip),wrk2g(/ip),wrk3g(/ip),wrk4g(/ip)

      do 2 kkkazu= 1,kazu*npe
           read(1,9000)n1,xxx,yyy,vxxx,vyyy
           wrk1g(kkkazu)=xxx
           wrk2g(kkkazu)=yyy
           wrk3g(kkkazu)=vxxx
           wrk4g(kkkazu)=vyyy
    2 continue

         do 740 ii = 1,kazu*npe
            pppx(ii)=wrk1g(ii)
            pppy(ii)=wrk2g(ii)
            vvvx(ii)=wrk3g(ii)
            vvvy(ii)=wrk4g(ii)
  740 continue

      return
      end
```

Fig.3.6 Modified subroutine allset.

```
      subroutine gid22
          :
          :
      do 50 i200=1,kazu
      do 40 j1=1,nyygmax
      do 30 i1=1,nxxgmax
         read(37,999)iiix,iiiy,vrexp,sss
         vwgr(i1,j1,i200) = vrexp
         vclo(i1,j1,i200) = vrexp*vrexp + sss*sss
         usovwgr(i1,j1,i200)= sss
         uvwbe(i1,j1,i200)= 0.0000D0
   30 continue
   40 continue
   50 continue
          :
      return
      end
```

Fig.3.7 Original subroutine gid22.

```
      subroutine gid22
          :
      common /paralell/ mynode,nodes
      dimension wrk1g(nxxgmax,nyygmax,kazu*npe)
     &          ,wrk2g(nxxgmax,nyygmax,kazu*npe)
     &          ,wrk3g(nxxgmax,nyygmax,kazu*npe)
     &          ,wrk4g(nxxgmax,nyygmax,kazu*npe)
!xocl processor p(npe)
!xocl subprocessor sp(npe)=p(1:npe)
!xocl index  partition ip=(sp,index=1:npe*kazu,part=band)
!xocl global wrk1g(:,:,/ip),wrk2g(:,:,/ip) &
!xocl       ,wrk3g(:,:,/ip),wrk4g(:,:,/ip)
          :
          :
      do 50 i200=1,kazu*npe
      do 40 j1=1,nyygmax
      do 30 i1=1,nxxgmax
         read(37,999)iiix,iiiy,vrexp,sss
         wrk1g(i1,j1,i200)=vrexp
         wrk2g(i1,j1,i200)=vrexp*vrexp + sss*sss
         wrk3g(i1,j1,i200)=sss
         wrk4g(i1,j1,i200)=0.0000D0
   30 continue
   40 continue
   50 continue

!xocl spread do
      do 52 ipe=1,npe
      do 51 i200=1,kazu
         is=kazu*(mynode-1)+i200
      do 41 j1=1,nyygmax
      do 31 i1=1,nxxgmax
         vwgr(i1,j1,i200)   = wrk1g(i1,j1,is)
         vclo(i1,j1,i200)   = wrk2g(i1,j1,is)
         usovwgr(i1,j1,i200)= wrk3g(i1,j1,is)
         uvwbe(i1,j1,i200)  = wrk4g(i1,j1,is)
   31 continue
   41 continue
   51 continue
   52 continue
!xocl end spread

      return
      end
```

Fig.3.8 Modified subroutine gid22.

```
      subroutine try
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

        do 100 i=1,maxtepe
        nwstepe=i
 9000   format(1i7)

          call nlconsp
          call elecden
          call data

           do 45 i45 = 1,kazu
              :
           call schredi(i45)
              :
  45      continue
 100    continue

        return
        end
```

Fig.3.9 Original subroutine try.

```
      subroutine try
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
      common /paralell/ mynode,nodes
      common /schredicom/
     &        nstatet(npe*kazu)
     &       ,vclot(nxxgmax,nyygmax,npe*kazu)
      dimension
     &        nstatel(npe*kazu)
     &       ,vclol(nxxgmax,nyygmax,npe*kazu)
!xocl processor p(npe)
!xocl subprocessor sp(npe)=p(1:npe)
!xocl index partition ip=(sp,index=1:npe*kazu,part=band)

       do 100 i=1,maxtepe
          :
       call data
c-------------------------------------------------------------
       do 440 ii=1,kazu*npe
         nstatet(ii)=0
        do 460 jjpo= 1,nyygmax
        do 480 iipo= 1,nxxgmax
         vclot(iipo,jjpo,ii)=0.0d0
  480  continue
  460  continue
  440  continue

!xocl spread do
       do 500 ipe=1,npe
       do 540 ii=1,kazu
          is=(ipe-1)*kazu+ii
          nstatet(is)=nstatet(is)+nstate(ii)
        do 560 jjpo= 1,nyygmax
        do 580 iipo= 1,nxxgmax
         vclot(iipo,jjpo,is)
        =vclot(iipo,jjpo,is)+vclo(iipo,jjpo,ii)
  580  continue
  560  continue
  540  continue
  500  continue
!xocl end spread sum(vclot), sum(nstatet)
c-------------------------------------------------------------
          do 45 i45 = 1,kazu
             :
          call schredi(i45)
             :
   45   continue
  100   continue

        return
        end
```

Fig.3.10 Modified subroutine try.

```
      subroutine schredi(i100)
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
             :
             :
        do 142 ibetue= 1,kazu
        if (ibetue .eq. i100) goto 142
        kt2em= nelep(nstate(i100),nstate(ibetue))
        do 140 nnpo= 1,nyygmax
        do 130 mmpo= 1,nxxgmax
           const=vclo(mmpo,nnpo,ibetue)
             :
130   continue
140   continue
142   continue
             :
          if (mod(nwstepe,20) .eq. 1) then
     do 989 i989= 1,kazu
             :
             :
989  continue
             :
          end if
             :
          xppt   = pppx(i100)
             :
          do 120 jjpo= 1,nyygmax
             :
             :
120     continue
             :
      return
      end
```

Fig.3.11 Original subroutine schredi.

```
      subroutine schredi(il00)
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
      common /paralell/ mynode,nodes
      common /schredicom/
     &         nstatet(npe*kazu)
     &        ,vclot(nxxgmax,nyygmax,npe*kazu)

        is=kazu*(mynode-1)+il00

                :
        do 142 ibetue= 1,kazu*npe
        if (ibetue .eq. is) goto 142

        kt2em= nelep(nstatet(is),nstatet(ibetue))
        do 140 nnpo= 1,nyygmax
        do 130 mmpo= 1,nxxgmax
           const=vclot(mmpo,nnpo,ibetue)
                :
130   continue
140   continue
142   continue
                :
         if (mod(nwstepe,20) .eq. 1) then
                :
      do 989 i989= 1,kazu*npe
                :
989 continue
                :
        endif
                :
                :
         xppt  = pppx(is)
                :
         do 120 jjpo= 1,nyygmax
                :
                :
120     continue

      return
      end
```

Fig.3.12 Modified subroutine schredi.

```
      subroutine calcu
                :
      do 9090 ionum= 1,kazu
                :
         xmolpo= pppx(ionum)
         ymolpo= pppy(ionum)

         do 1000 imoll= 1,kazu
         if (imoll .eq. ionum) goto 1000
                :
                t126= dipox(ionum)+dipox(imoll)
                :
1000    continue
9090  continue

      do 1101 i101=1,kazu
        call interpo(i101)
1101  continue

      do 1100 i11=1,kazu

      call ionpo4(i11)

      akx= vvvx(i11)
      aky= vvvy(i11)
      vvvx(i11)=vvvx(i11)+aaax(i11)*deltt
      vvvy(i11)=vvvy(i11)+aaay(i11)*deltt
      vvpx= akx+aaax(i11)*deltt*0.500D0
      vvpy= aky+aaay(i11)*deltt*0.500D0
      subpox=pppx(i11)+vvvx(i11)*deltt
      subpoy=pppy(i11)+vvvy(i11)*deltt
      pppx(i11)=subpox
      pppy(i11)=subpoy

1100  continue

      return
      end
```

Fig.3.13 Original subroutine calcu.

```
      subroutine calcu
                :
c.para <<
      common /trncheck/ trs,tre
      common /paralell/ mynode,nodes
      common /ionpo4com/ vaftmp(npe*kazu,nfine,nfine)
     &,vafpxtmp(npe*kazu,nfine,nfine),vafpytmp(npe*kazu,nfine,nfine)

      dimension
     & aaaxt(kazu*npe),aaayt(kazu*npe),vvvxt(kazu*npe)
     &,vvvyt(kazu*npe),pppxt(kazu*npe),pppyt(kazu*npe)

!xocl  processor p(npe)
!xocl  subprocessor sp(npe)=p(1:npe)
!xocl  index  partition ip=(sp,index=1:npe*kazu,part=band)
                :
         do 7750 ii= 1,kazu*npe
            dipoxtmp(ii)=0.0d0
 7750  continue

!xocl spread do
         do 7700 ipe= 1,npe
         do 7710 ii = 1,kazu
            iis=kazu*(ipe-1)+ii
            dipoxtmp(iis)=dipox(ii)
 7710  continue
 7700  continue
!xocl end spread sum(dipoxtmp)

      do 9090 ionum= 1,kazu
        is=kazu*(mynode-1)+ionum
                :
        xmolpo = pppx(is)
        ymolpo = pppy(is)
                :
c   force to ion by iner-electron deformation
         do 1000 imoll= 1,kazu*npe
         if (imoll .eq. is) goto 1000
                :
               t126= dipoxtmp(is)+dipoxtmp(imoll)
                :
 1000  continue
 9090  continue
                :
      do 1101 il01=1,kazu
        call interpo(il01)
 1101  continue
                :
                :
                :
      return
      end
```

Fig.3.14 Modified subroutine calcu.(1/3)

```
      subroutine calcu
                  :
         do 750 ii= 1,kazu*npe
         do 760 j  = 1,nfine
         do 770 i  = 1,nfine
            vaftmp(ii,i,j)  =0.0d0
            vafpxtmp(ii,i,j)=0.0d0
            vafpytmp(ii,i,j)=0.0d0
  770  continue
  760  continue
  750  continue

!xocl spread do
         do 700 ipe= 1,npe
         do 710 ii = 1,kazu
            iis=kazu*(ipe-1)+ii
         do 720 j  = 1,nfine
         do 730 i  = 1,nfine
            vaftmp(iis,i,j)  =vaf(ii,i,j)
            vafpxtmp(iis,i,j)=vafpx(ii,i,j)
            vafpytmp(iis,i,j)=vafpy(ii,i,j)
  730  continue
  720  continue
  710  continue
  700  continue
!xocl end spread sum(vaftmp),sum(vafpxtmp),sum(vafpytmp)

c         ------------------
        do 1091 ii= 1,kazu
          call fconjx(ii)
          call fconjy(ii)
          call fcnjxy(ii)
 1091   continue
c         ------------------

        do 1100 il1=1,kazu
         is=kazu*(mynode-1)+il1
                  :
          call ionpo4(il1)
                  :
        aaax(is)= (xforce(il1)+subefx+ffqqq(il1)+fffcha)/situk
        aaay(is)= (yforce(il1)+subefy )/situk
        akx= vvvx(is)
        aky= vvvy(is)
        vvvx(is)=vvvx(is)+aaax(is)*deltt
        vvvy(is)=vvvy(is)+aaay(is)*deltt
        vvpx= akx+aaax(is)*deltt*0.500D0
        vvpy= aky+aaay(is)*deltt*0.500D0
        subpox=pppx(is)+vvvx(is)*deltt
        subpoy=pppy(is)+vvvy(is)*deltt
        pppx(is)=subpox
        pppy(is)=subpoy
                  :
 1100   continue
                  :
                  :
        return
        end
```

Fig.3.14 Modified subroutine calcu.(2/3)

```
      subroutine calcu
                  :
                  :
                  :
        do 960 ii=1,kazu*npe
           aaaxt(ii)=0.0d0
           aaayt(ii)=0.0d0
           vvvxt(ii)=0.0d0
           vvvyt(ii)=0.0d0
           pppxt(ii)=0.0d0
           pppyt(ii)=0.0d0
  960    continue

!xocl spread do
         do 970 ipe=1,npe
         do 975 ii=1,kazu
            i=kazu*(ipe-1)+ii
            aaaxt(i)=aaax(i)
            aaayt(i)=aaay(i)
            vvvxt(i)=vvvx(i)
            vvvyt(i)=vvvy(i)
            pppxt(i)=pppx(i)
            pppyt(i)=pppy(i)

  975    continue
  970    continue
!xocl end spread sum(aaaxt), sum(aaayt), sum(vvvxt) &
!xocl          , sum(vvvyt), sum(pppxt), sum(pppyt)

         do 987 ii=1,kazu*npe
            aaax(ii)=aaaxt(ii)
            aaay(ii)=aaayt(ii)
            vvvx(ii)=vvvxt(ii)
            vvvy(ii)=vvvyt(ii)
            pppx(ii)=pppxt(ii)
            pppy(ii)=pppyt(ii)
  987    continue

       return
       end
```

Fig.3.14 Modified subroutine calcu.(3/3)

```
      subroutine ionpo4(il00)
                 :
            xmolpo= pppx(il00)
            ymolpo= pppy(il00)

        do 300 iother=1,kazu
        do 200 j= 1,nfine
        do 100 i= 1,nfine
            xgenpo= vafpx(iother,i,j)
            ygenpo= vafpy(iother,i,j)
                 :
        totvaf= vaf(iother,i,j)
                 :
               xemc= xemc + xsubme/rkyo3*totvaf
               yemc= yemc + ysubme/rkyo3*totvaf
               poteki= poteki + pofeleb

  100   continue
  200   continue
  300   continue
                 :
      return
      end
```

Fig.3.15 Original subroutine ionpo4.

```
      subroutine ionpo4(il00)
                :
      common /paralell/ mynode,nodes
      common /ionpo4com/ vaftmp(npe*kazu,nfine,nfine)
     &,vafpxtmp(npe*kazu,nfine,nfine),vafpytmp(npe*kazu,nfine,nfine)

!xocl processor p(npe)
!xocl subprocessor sp(npe)=p(1:npe)

      is=kazu*(mynode-1)+il00
                :
      xmolpo= pppx(is)
      ymolpo= pppy(is)

        do 300 iother=1,kazu*npe
        do 200 j= 1,nfine
        do 100 i= 1,nfine
            xgenpo= vafpxtmp(iother,i,j)
            ygenpo= vafpytmp(iother,i,j)
                 :
            totvaf= vaftmp(iother,i,j)
                 :
               zxemc= zxemc + xsubme/rkyo3*totvaf
               zyemc= zyemc + ysubme/rkyo3*totvaf
               zpoteki= zpoteki + pofeleb
                 :
  100   continue
  200   continue
  300   continue
                 :
        xemc  = zxemc
        yemc  = zyemc
        poteki= zpoteki
                 :
      return
      end
```

Fig.3.16 Modified subroutine ionpo4.

```
      subroutine skaku

      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

     do 2000 maa=1,maxtep
     do 1000 i11=1,kazu
       write(3,9199) maa,dpppx(i11,maa),dpppy(i11,maa),
    &              dvvvx(i11,maa),dvvvy(i11,maa),
    &              deeex(i11,maa),deeey(i11,maa),
    &              dpxre(i11,maa),dpyre(i11,maa)
       write(39+i11,9099)  dundou(i11,maa),
    &                      denepii(i11,maa),
    &                      denepei(i11,maa)
1000  continue
2000  continue

      do 3000 i30=1,nttsp
       do 1001 i11=1,kazu
        write(38,9299) i30,denergy(i11,i30),denekin(i11,i30),
    &                  denepot(i11,i30)
        write(34,9399) ddnti(i30),dsoukan(i30),dusok(i30)
1001    continue
3000    continue

      do 4004 mn=1,kazu
      do 4003 k=1,nvti
      do 4002 j=1,nyygmax
      do 4001 i=1,nxxgmax
         nxygm= nyygmax*(j-1)+i
       write(15,9533)i,j,dvclo(k,mn,nxygm)
       write(14,9433)i,j,dvwgr(k,mn,nxygm),dusovwgr(k,mn,nxygm)
4001   continue
4002   continue
4003   continue
4004   continue

       return
       end
```

Fig.3.17 Original subroutine skaku.

```
      subroutine skaku
c.para <<
      common /paralell/ mynode,nodes
!xocl  processor p(npe)
!xocl  subprocessor sp(npe)=p(1:npe)

      dimension
     &  dpppxg(kazu*npe,maxtep)    ,dpppyg(kazu*npe,maxtep)
     & ,dvvvxg(kazu*npe,maxtep)    ,dvvvyg(kazu*npe,maxtep)
     & ,deeexg(kazu*npe,maxtep)    ,deeeyg(kazu*npe,maxtep)
     & ,dpxreg(kazu*npe,maxtep)    ,dpyreg(kazu*npe,maxtep)
     & ,dundoug(kazu*npe,maxtep)   ,denepiig(kazu*npe,maxtep)
     & ,denepeig(kazu*npe,maxtep)  ,denergyg(kazu*npe,nttsp)
     & ,deneking(kazu*npe,nttsp)   ,denepotg(kazu*npe,nttsp)
     & ,dvclog(nvti,kazu*npe,nsku),dvwgrg(nvti,kazu*npe,nsku)
     & ,dusovwgrg(nvti,kazu*npe,nsku)
     & ,coefljg(kazu*npe,kazu*npe,maxtep)
     & ,coefclg(kazu*npe,kazu*npe,maxtep)


      do 1500 i11=1,kazu*npe
      do 2500 maa=1,maxtep
        dpppxg(i11,maa)  =0.0d0
        dpppyg(i11,maa)  =0.0d0
        dvvvxg(i11,maa)  =0.0d0
        dvvvyg(i11,maa)  =0.0d0
        deeexg(i11,maa)  =0.0d0
        deeeyg(i11,maa)  =0.0d0
        dpxreg(i11,maa)  =0.0d0
        dpyreg(i11,maa)  =0.0d0
        dundoug(i11,maa) =0.0d0
        denepiig(i11,maa)=0.0d0
        denepeig(i11,maa)=0.0d0
2500  continue

      do 3500 i30=1,nttsp
         denergyg(i11,i30)=0.0d0
         deneking(i11,i30)=0.0d0
         denepotg(i11,i30)=0.0d0
3500  continue

      do 4504 jj=1,nsku
      do 4503 k=1,nvti
         dvclog(k,i11,jj)   =0.0d0
         dvwgrg(k,i11,jj)   =0.0d0
         dusovwgrg(k,i11,jj)=0.0d0
4503  continue
4504  continue

      do 5504 istep=1,maxtep
      do 5503 imol1=1,kazu*npe
         coefljg(imol1,i11,istep)=0.0d0
         coefclg(imol1,i11,istep)=0.0d0
5503  continue
5504  continue
1500  continue
```

Fig.3.18 Modified subroutine skaku.(1/3)

```
                        :
!xocl spread do
      do 9000 ipe=1,npe
      do 1550 ill=1,kazu

      iis=kazu*(ipe-1)+ill
      do 2550 maa=1,maxtep
        dpppxg(iis,maa)  =dpppx(ill,maa)
        dpppyg(iis,maa)  =dpppy(ill,maa)
        dvvvxg(iis,maa)  =dvvvx(ill,maa)
        dvvvyg(iis,maa)  =dvvvy(ill,maa)
        deeexg(iis,maa)  =deeex(ill,maa)
        deeeyg(iis,maa)  =deeey(ill,maa)
        dpxreg(iis,maa)  =dpxre(ill,maa)
        dpyreg(iis,maa)  =dpyre(ill,maa)
        dundoug(iis,maa) =dundou(ill,maa)
        denepiig(iis,maa)=denepii(ill,maa)
        denepeig(iis,maa)=denepei(ill,maa)
 2550  continue

      do 3550 i30=1,nttsp
         denergyg(iis,i30)=denergy(ill,i30)
         deneking(iis,i30)=denekin(ill,i30)
         denepotg(iis,i30)=denepot(ill,i30)
 3550  continue

      do 4554 jj=1,nsku
      do 4553 k=1,nvti
         dvclog(k,iis,jj)   =dvclo(k,ill,jj)
         dvwgrg(k,iis,jj)   =dvwgr(k,ill,jj)
         dusovwgrg(k,iis,jj)=dusovwgr(k,ill,jj)
 4553  continue
 4554  continue

      do 5554 istep=1,maxtep
      do 5553 imoll=1,kazu*npe
         coefljg(imoll,iis,istep)=coeflj(imoll,ill,istep)
         coefclg(imoll,iis,istep)=coefcl(imoll,ill,istep)
 5553  continue
 5554  continue

 1550  continue
 9000  continue
!xocl end spread sum(dpppxg),sum(dpppyg),sum(dvvvxg),sum(dvvvyg)       &
!xocl    ,sum(deeexg),sum(deeeyg),sum(dpxreg),sum(dpyreg),sum(dundoug)&
!xocl    ,sum(denepiig),sum(denepeig),sum(denergyg),sum(deneking)      &
!xocl    ,sum(denepotg),sum(dvclog),sum(dvwgrg),sum(dusovwgrg)         &
!xocl    ,sum(coefljg),sum(coefclg)
                        :
```

Fig.3.18 Modified subroutine skaku.(2/3)

```
                          :
      do 2000 maa=1,maxtep
      do 1000 ill=1,kazu*npe
        write(3,9199) maa,dpppxg(ill,maa),dpppyg(ill,maa),
     &               dvvvxg(ill,maa),dvvvyg(ill,maa),
     &               deeexg(ill,maa),deeeyg(ill,maa),
     &               dpxreg(ill,maa),dpyreg(ill,maa)
        write(39+ill,9099)  dundoug(ill,maa),
     &                      denepiig(ill,maa),
     &                      denepeig(ill,maa)
1000  continue
2000  continue

      do 3000 i30=1,nttsp
      do 1001 ill=1,kazu*npe
        write(38,9299) i30,denergyg(ill,i30),deneking(ill,i30),
     &                     denepotg(ill,i30)
        write(34,9399) ddnti(i30),dsoukan(i30),dusok(i30)
1001  continue
3000  continue

      do 4004 mn=1,kazu*npe
      do 4003 k=1,nvti
      do 4002 j=1,nyygmax
      do 4001 i=1,nxxgmax
         nxygm= nyygmax*(j-1)+i
       write(15,9533)i,j,dvclog(k,mn,nxygm)
       write(14,9433)i,j,dvwgrg(k,mn,nxygm),dusovwgrg(k,mn,nxygm)
4001   continue
4002   continue
4003   continue
4004   continue

      do 1900 istep=1,maxtep
      do 1950 ionum=1,kazu*npe
      do 1970 imoll=1,kazu*npe
         if (imoll .eq. ionum) goto 1970
              write(52,9696)coefljg(imoll,ionum,istep)
     &                      ,coefclg(imoll,ionum,istep)
1970  continue
1950  continue
1900  continue

       return
       end
```

Fig.3.18 Modified subroutine skaku.(3/3)

```
QQQF.vpp_p/ ——— *.f, includefile,makefile
    ├——— SH/
    └——— DATA/

QQQF.vpp_p/
Makefile.300  vpp300 並列実行用 Makefile 記述例
Makefile.500  vpp500 並列実行用 Makefile 記述例
QQQ.h     インクルードファイル（配列宣言など）
prame4    インクルードファイル（パラメータファイル）

*.f     以下，ソースファイル一覧
abspec.f  after.f   after1.f  allset.f  before.f  before1.f
calcu.f   clear.f   conju.f   convr.f   data.f    datass.f
edatas2.f elecden.f eneaf.f   enegbe.f  fcnjxy.f  fcnvrx.f
fcnvry.f  fconjx.f  fconjy.f  fft.f     fftho.f   frcnvx.f
frcnvy.f  gid22.f   grids.f   ibitr.f   ifftenx.f iffteny.f
interpo.f invxfft.f invyfft.f ionbou.f  ionpo4.f  ionpre.f
lasdat.f  nlconho.f nlconsp.f qqq.f     rconv.f   rxfft.f
ryfft.f   schdiho.f schredi.f skaku.f   try.f

SH/
go_sh.p       nqs 投入用シェルスクリプト記述例

DATA/
stakxe.data   unit01 よりの入力データ
stawv.data    unit37 よりの入力データ
namlist       unit05 ネームリストファイル
```

Fig.3.19 File lists of the QQQF.vpp_p.tar.Z.

```
#
# Center Sample Makefile
#                Write Day : Wed May  1 09:03:44 1996
#

F77      = frt
MAKE     = Makefile
OPT      = -Wx -Oe -Ps -AB
LIBES    =
INCLUDES = prame4 QQQ.h
TERGET   = a.out

OBJS = \
       qqq.o  abspec.o  after.o  after1.o  allset.o\
       before.o  before1.o  calcu.o  clear.o  conju.o\
       convr.o  data.o  datass.o  edatas2.o  elecden.o\
       eneaf.o  enegbe.o  fcnjxy.o  fcnvrx.o  fcnvry.o\
       fconjx.o  fconjy.o  fft.o  fftho.o  gid22.o\
       frcnvx.o  frcnvy.o  grids.o  ifftenx.o  iffteny.o\
       interpo.o  invxfft.o  invyfft.o  ionbou.o  ionpo4.o\
       ionpre.o  lasdat.o  nlconho.o  nlconsp.o  rconv.o\
       rxfft.o  ryfft.o  schredi.o  skaku.o\
       try.o  ibitr.o

.f.o :
 $(F77) $(OPT) $(INCLUDES) -c $<

all :$(OBJS)
 $(F77) -o $(TERGET) $(OPT) $(OBJS) $(LIBES)

clean :
 rm -f $(OBJS)
```

Fig.3.20 Makefile for making load-module.

```
#!/bin/csh -f
#####################################################
#@$-q  vpp8      # set queue class
#@$-lPv 8        # set PE Number
#@$-lt 2:00:00      # set CPU time limit to XX:XX (mm:ss)
#@$-C  QQQF     # set code name to XXXXX
#@$-eo
#####################################################
cd $HOME/QQQF

# read data
setenv fu01 $HOME/QQQF/wkvfl/stakxe.data
setenv fu37 $HOME/QQQF/wkvfl/stawv.data

setenv fu02 $HOME/QQQF/wkvfl/sscc_vec2/tempa.data
setenv fu03 $HOME/QQQF/wkvfl/sscc_vec2/molpv.data

#  check wave.data last step
setenv fu14 $HOME/QQQF/wkvfl/sscc_vec2/wave.data.vec2
setenv fu15 $HOME/QQQF/wkvfl/sscc_vec2/tmitu.data.vec2

setenv fu16 $HOME/QQQF/wkvfl/sscc_vec2/mitu.data
setenv fu34 $HOME/QQQF/wkvfl/sscc_vec2/spec.data

#  check enevw.data every step
setenv fu38 $HOME/QQQF/wkvfl/sscc_vec2/enevw.data.vec2

setenv fu39 $HOME/QQQF/wkvfl/sscc_vec2/jyusin.data
setenv fu40 $HOME/QQQF/wkvfl/sscc_vec2/enegg1.data.vec
setenv fu41 $HOME/QQQF/wkvfl/sscc_vec2/enegg2.data.vec
setenv fu42 $HOME/QQQF/wkvfl/sscc_vec2/enegg3.data.vec
setenv fu43 $HOME/QQQF/wkvfl/sscc_vec2/enegg4.data.vec
setenv fu44 $HOME/QQQF/wkvfl/sscc_vec2/enegg5.data.vec
setenv fu45 $HOME/QQQF/wkvfl/sscc_vec2/enegg6.data.vec
setenv fu46 $HOME/QQQF/wkvfl/sscc_vec2/enegg7.data.vec
setenv fu47 $HOME/QQQF/wkvfl/sscc_vec2/disou.data
setenv fu48 $HOME/QQQF/wkvfl/sscc_vec2/avewv.data
setenv fu49 $HOME/QQQF/wkvfl/sscc_vec2/enetest.data
setenv fu50 $HOME/QQQF/wkvfl/sscc_vec2/force.data
setenv fu52 $HOME/QQQF/wkvfl/sscc_vec2/coef.dddd.vec
setenv FOR51 $HOME/QQQF/wkvfl/sscc_vec2/avf.100

a.out < namlist
```

Fig.3.21 Shell script for parallel execution.

Table.3.1 Execution time on VPP300.

	modified version (通信所要時間)	original version	Speedup ratio
QD 計算部総計	0.1193e+03 sec (0.5768e+01 sec)	0.8755e+03 sec	7.34
MD 計算部総計	0.9538e+02 sec (0.3562e+02 sec)	0.3315e+03 sec	3.48
時間発展計算部分総計	0.2147e+03 sec (0.4142e+02 sec)	0.1207e+04 sec	5.62
プログラム全体	0.2997e+03 sec	0.1240e+04 sec	4.14

Table.3.2 memory size.

並列化前	134.5 MB
並列化後	136.4 MB

4. ParagonにおけるQQQF並列化

VPP上で開発した，ベクトル版量子分子動力学コードQQQFをParagonへ移植し，複数台のノードを用いて，並列実行できるよう変更した．並列化用プログラムにはオリジナルコードのFFTサブルーチンのみ変更したのものを用いた．

4．1 VPP500からParagonへの移植

Paragonでは，FORTRAN77 がサポートされており，VPP 固有のサービスサブルーチン以外は変更なく移植ができる．ただし，プログラムの実行方法に関して若干の違いがあり，4．4実行方法に述べる．移植には，オリジナルコードにおいてFFTサブルーチンの変更後のものを用いた．また，プログラムに挿入した時間計測用のサブルーチンは全てParagon 用の時間計測サブルーチンに置き換えた．

4．2 並列化

Paragon はスカラー型の超並列計算機であり，800ノードの同時使用が可能である．まず，「3．VPP300上での並列化」と同様の方法で，7並列実行用にプログラムを変更し，正常動作を確認した後，さらに多数のノードが使用できるように変更を行った．このため，7並列実行プログラムへの変更については，「3．VPP300上での並列化」とは異なる点を中心に説明を行う．

4．2．1 7並列実行プログラムへの変更

「3．VPP300上での並列化」と同様に，1電子・イオン（以降，粒子と記す）に関する計算を行うノードを，7台並列に動作させる．このため，粒子数を指定するパラメータkazu を1とする．ノードがどの番号の粒子を計算しているか判別する場合は，システムコールから取得するノード識別番号を用いる．また，必要に応じてノード間通信を行う．

（1） インクルードファイル prame4

並列プログラムに使用するノード台数を指定するパラメータ npe を与える．npe は，インクルードファイル prame4 に記述した．また，粒子数を指定するパラメータ kazu を7から1へ変更した．Fig.4.1 に変更前のインクルードファイル prame4 を，Fig.4.2 に変更後の prame4 を示す．

（2） メインプログラム

Paragon 上で nx ライブラリ[6]を用いた並列実行を行うには，実行する全てのソースファイル中でインクルードファイル 'fnx.h' をインクルードする必要がある．またプログラムの問い合わせ mynode(),numnodes() [7]を行い，変数 iam に ノード識別番号，変数 nodes に使用ノード数を与えた．iam,nodes はコモン変数として /paralell/ に格納した．装置番号５番からのネームリスト入力については，ノード識別番号０(iam=1) のノードが入力を行い，ノード間の通信用サブプログラム csend, crecv [7]を用いて，他の全ノードと入力データを送受信する方法を採用した．

Fig.4.3 に変更前のメインプログラム，Fig.4.4 に変更後の概要を示す．なお，do 1091 ループはサブルーチン calcu 内に移動させた．

（3） サブルーチン allset

入力部分に関しては，上記（2）と同様の方法を用いた．Fig.4.5 に変更前のサブルーチン allset の概要，Fig.4.6 に変更後の概要を示す．なお，入力部分以外の変更は「VPP300 における QQQF 並列化３．３．３」と同様である．

（4） サブルーチン gid22

入力部分に関しては，上記（2）と同様の方法を用いた．ただし，入力データを格納する配列が，元の７分の１の大きさの配列となったのに対し，入力データは元の大きさのままである．このため入力データと同じ大きさの作業用配列を用意し，識別番号０のノードがこの配列に入力を格納した後，各ノードに必要なデータの抽出と送受信を行う方法を採用した．また csend,crecv が必要とする作業用バッファとして，新たに配列 wrk1g, wrk2g,wrk3,wrk4g の宣言を行った．

Fig.4.7 に変更前のサブルーチン gid22 の概要，Fig.4.8 に変更後の概要を示す．なお，入力部分以外の変更は「VPP300 におけるQQQF 並列化３．３．４」と同様である．

（5） サブルーチン try

本並列化では，他ノードが持つ粒子のデータを参照するには，ノード間通信が必要となる．この場合，グローバルオペレーション gdsum [7] を用いた総和計算により，全粒子に関するデータを各ノードに配置する．配列 vclot を例に取り，総和計算の手順を①～③に説明する．配列 nstate に関しても同様の手順を用いた．また，gdsum が必要とする作業用バッファとして，配列 tmp1 tmp2 を新たに宣言した．

Fig.4.9 に変更前のサブルーチンtry の概要，Fig.4.10 に変更後の概要を示す．なお，総和計算以外の変更は「VPP300におけるQQQF並列化３．３．５」と同様である．

①各ノード上に配列vclo(*)の作業用配列vclot(*,7)を用意してゼロクリアする．

②各ノードが所有する粒子のデータを，作業配列の該当要素に格納する．

例としてノード識別番号 iam の場合； vclot(*,iam)=vclo(*)

③格納後， gdsum を用いて vclot(*,7) の全ノード間での総和を取る．

（6） サブルーチン schredi

Fig.4.11 に変更前のサブルーチン schredi の概要，Fig.4.12 に変更後の概要を示す．変更内容は，「VPP300におけるQQQF並列化3．3．6」と同様である．

（7） サブルーチン calcu

本サブルーチンにおいても，上記（5）と同様の理由で他ノードが持つ粒子のデータを参照するため総和計算を行う．総和計算には，上記（5）と同様の方法を用いた．また，gdsum の作業用バッファとして，配列 tmp1 tmp2,tmp3 を新たに宣言した．
Fig.4.13 に変更前のサブルーチン calcu の概要，Fig.4.14 に変更後の概要を示す．総和計算以外の変更は「VPP300 におけるQQQF 並列化3．3．7」と同様である．

（8） サブルーチン ionpo4

変数 xemc の定義を例にあげ，Fig.4.15 に変更前のサブルーチン ionpo4 の概要，Fig.4.16 に変更後の概要を示す．本サブルーチンにおける変更内容は，「VPP 300 におけるQQQF並列化3．3．8」と同様である．

（9） サブルーチン skaku

本並列化では1粒子分の計算を行うノードを粒子数台並列に動作させ，また各ノード上には1粒子分のデータを配置するようにした．全粒子に関する計算結果を出力するために，サブルーチン skaku において全粒子に関するデータを全ノード上に配置し，これをノード識別番号0のノードより出力するよう変更した．なお，全粒子分のデータの作成には，上記（5）と同様の方法を用いた．本サブルーチンにおける変更は，コモン文(/paralell/) の追加および，作業用配列の追加・総和計算である．Fig.4.17 に変更前のサブルーチン skaku 概要，Fig.4.18 変更後の概要を示す．

4．2．2 使用ノード数の拡大

（1） 7並列実行用への変更後のプログラム解析

時間ステップ maxtepe=20,maxtep=10，1ノード当たりの粒子数 kazu=1，使用ノード数 npe=7 において，7並列実行用への変更および自動並列化[8]後のプログラム各部分計算所要時間の測定を行った．自動並列化機能の適用については付録1に記す．自動並列化の結果を Table.4.1に示す．この結果，最も計算コストの高い部分は，サブルーチンschredi であり，次はサブルーチン iompo4 であった．この2か所のコストの和は1時間ステップの約98%を占める．サブルーチン schredi,iompo4 の2か所を並列化の対象とた．

（2） ノードグループの定義

使用ノード数の拡大においては，サブルーチン try の do 45 の並列処理と両立するよう，サブルーチン schredi の並列処理を行った．このため，複数ノードから構成されるノードのグループを考え，各グループでサブルーチン try の do 45 を並列処理し，かつグループ内の複数ノードでサブルーチン schredi の do 142 を並列処理することにした．

このように並列化を行うと，ノード間の通信動作をグループ間もしくはグループ内というように，グループごとに制御する必要が生じる．本並列化では，ノードの通信動作を総和計算に限定したため，ノード間の通信動作はグループ間もしくはグループ内での総和計算を考えればよい．ところが，nx ライブラリにはグループ間・内総和計算を行うシステムコールが用意されていない．このため，グループ間・内総和計算を行うサブルーチン test を新規に開発した．このサブルーチン test のプログラムリストは付録2に示す．

なお，サブルーチン try do 45 の回転数は7，サブルーチン schredi の do 142 の回転数は32(=2^5)である．これより，グループ数は7,グループ内ノード数は2のn乗(ただし，n は 0 から 5 まで)，使用ノード数は7×2^n（グループ数×グループ内ノード数）とした．

（3） インクルードファイル prame4

プログラムに，グループ数(npey)およびグループ内ノード数(npex)を指定するパラメータを与える．npex,npeyは，インクルードファイル prame4 に記述した．Fig.4.19 に使用ノード数の拡大後のインクルードファイル prame4 を示す．

（4） メインプログラム

メインプログラムを，ノード識別番号，グループ数，グループ内ノード数から，グループ識別番号，グループ内識別番号の2変数を決定するよう変更した．また，変数 iam を要素数2の配列に変更し，iam(1) にグループ内識別番号，iam(2) にグループ識別番号を格納した．iam(1),iam(2) は，コモン領域 /parallel/を用いて各ソースプログラムに渡す．各ソースプログラムについてもコモン領域/parallel/の変更を行った．Fig.4.20 に変更後のメインプログラムにおけるノードグループ定義の概要を示す．

（5） サブルーチン try

使用ノード数の拡大では，各グループで do 45 ループを並列処理する．このような並列化を行うと，各粒子に関するデータから，全粒子に関するデータを作成する場合の総和計算（例：本節４．２．１（５））は，グループ間での総和計算となる．このため，サブルーチン test を用いてグループ間総和計算を行うよう変更した．また，test が必要とする作業用配列 tmp1,tmp2,dnstat の宣言を追加した．Fig.4.21 に使用ノード数拡大後の try 概要を示す．

（6） サブルーチン schredi

使用ノード数の拡大では，グループ内の複数ノードで do 142 ループを並列処理する．まず do 142 ループ内の処理を明確にするためのプログラム記述の変更を行い，次に do 142 ループの並列化を行った．

do 142 ループの並列処理のため本並列化では，do 142 ループ内部の do 122 ループを分割し do 122 の範囲をノードごとに異なる値の２変数 istart,iend で与えた．この istart,iend は時間ステップによらないので，時間ステップが１の場合のみ検出しコモン領域 /indxcom/ に格納することで，プログラム実行中保存するようにした．

do 122 ループを分割した場合，配列 vwcons のグループ内総和計算が必要である．このため，グループ内総和計算を行うサブルーチン test を用いた．Fig.4.22 に do 142 ループの記述変更前の内容，Fig.4.23 に変更後の内容を示す．なお，Fig.4.22，Fig.4.23 中，変数 is,i100 はともに定数である．また，Fig.4.24 に並列化後の schredi do 142 の記述を示す．

（7） サブルーチン calcu

本並列化においては，各グループで do 9090,do 1101,do 1091 ループが並列処理される．本サブルーチンにおいて，配列aaaxt,aaayt,vvvxt,vvvyt,pppxt,pppyt のグループ間総和計算時に使用するサブルーチンの呼び出しを，１回にするために行った３点の変更を①～③に述べる．

①作業配列の変更(aaaxt,aaayt,vvvxt,vvvyt,pppxt,pppyt から pppgl(*,6))

②配列 pppglの総和計算後この各要素から，上記６配列(aaax,..,pppy)抽出への変更

③作業用配列 pppglbf の宣言の追加（配列 pppglの総和計算用）

また，サブルーチン ionpo4 に渡す３配列 vaftmp,vafpxtmp,vafpytmp の総和演算による作成時の通信コストが非常に高いため，次（８）に述べるサブルーチン ionpo4 の変更により，この総和演算を中止した．この中止にあわせて，本サブルーチン calcu において④⑤の変更を行った．Fig.4.25 に変更後の calcu の概要を示す．

④コモン文 /ionpo4com/ の削除

⑤作業領域 tmp1,tmp2,tmp3 の削除

（8） サブルーチン ionpo4

上記（7）の，サブルーチン calcu での3配列の作成を中止のため，本サブルーチンにおいて，3変数 xemc,yemc,poteki の定義手順を変更した．本サブルーチンでの，宣言等の変更を①②，変更後の3変数の定義手順を③～⑤に述べる．変数 xemc を例にあげ，Fig.4.26 に変更前の ionpo4 の概要，Fig.4.27 に変更後の概要を示す．Fig.4.26,Fig.4.27 のプログラム中 i100 はともに定数である．

①作業配列dsmsg(3),drmsg(3),iclstr(2),ndtop(2)の宣言

②変数から配列への変更

xemc,yemc,poteki から zxemc(npey),zyemc(npey),zpoteki(npey)

③部分和 zxemc(istep) の作成(do 1000)

④メッセージ（部分和）の送信(do 2000 csend)

⑤受信メッセージ（部分和）より総和値 xemc の作成(do 4000)

さらに，使用ノード数の拡大において本節4．2．2（5），（6），（7）と両立するように，また同様の方法を用いて，本サブルーチン ionpo4 の do 300 ループを並列化した．すなわち，calcu do 1100 を各グループで，サブルーチン ionpo4 の do 200 ループをグループ内の各ノードで，並列に処理するよう変更した．なお，do 200 ループは do 300 ループ内部のものである．変数 xemc に必要となるグループ内総和計算にはサブルーチン test を用いた．変数 yemc,poteki に関しても同様である．Fig.4.28 に，本サブルーチン do 300 ループ並列化後の記述を示す．

（9） サブルーチン skaku

本並列化では，1粒子に関する計算を行うノードを7グループ並列に動作させるよう変更を行った．このため，各グループ上に1粒子に関するデータが配置されている．サブルーチン skaku において全粒子に関するデータを全ノード上に配置するため，グループ間の総和計算を行った．グループ間総和計算にはサブルーチン test を用いた．このようにして得た全粒子に関する計算結果を，ノード識別番号0のノードより出力するよう変更した．Fig.4.29 に変更後の skaku 総和演算部分を示す．

4．3 並列化の効果

「4．2．1」から「4．2．2」への変更に関する並列化効果の評価として，プログラム各部分の計算所要時間の測定を行った．7並列実行における並列化の効率は非常に良く，使用ノード数を拡大した場合の効率はノード総数58程度まで良好である．

測定は，時間ステップmaxtepe=20,maxtep=500，1ノード当たりの粒子数 kazu=1，グループ数 npey=7 において，グループ内ノード数 npex が可変の場合とした．なお，npex=1 のテスト実行には，使用ノード数の拡張以前のプログラムを用いた．これらの計算パラメータはインクルードファイル prame4 に記述した．Table.4.2にプログラム各部分の計算所用時間，Table.4.3に速度向上，Table.4.4 に必要メモリ量を示す．

4．4　実行方法

QQQF.paragonII_p.tar.Z を解凍後，ロードモジュールの作成，NQS への投入の順に行う．QQQF.paragonII_p.tar.Z には，paragon 上でのプログラム実行に必要な，ファイル一式（ソースプログラム・インクルードファイル・Makefile 記述例・nqs 投入用シェルスクリプト記述例・入力データファイル）を，アーカイブ形式にまとめ圧縮して格納してある．

4．4．1　アーカイブ形式ファイルについて

（1）圧縮ファイルの ftp 等による転送は，バイナリモードで行う．

転送例　**>bin** [RET]
　　>put *QQQF.paragonII_p.tar.Z* [RET]

（2）ファイルの解凍には，uncompress，tar コマンドを使用する．

解凍例　1%**uncompress** *QQQF.paragonII_p.tar.Z* [RET]
　　2%**tar -xvf** *QQQF.paragonII_p.tar* [RET]

（3）Fig.4.30 に QQQF.paragonII_p.tar.Z 解凍時のファイルリストを示す．

4．4．2　ロードモジュールの作成（ソースプログラムのコンパイル・リンク）

ロードモジュールの作成には，make コマンドを実行する．Fig.4.31 に Makefile の記述例を示す．

（1）ロードモジュールの作成

例　**%make** [RET]

4．4．3　プログラムの実行

プログラムの実行は，ＮＱＳ投入用シェルスクリプトを qsub コマンドによってＮＱＳへ投入して行う．Fig.4.32 に14並列実行におけるＮＱＳ投入用シェルスクリプトの記述例を示す．Fig.4.32 のようなシェルスクリプトを，ジョブクラス，ディレクトリ名，ファイル名等を実行時の環境に適したように変更し使用する．また，実行オプション -sz で指定する使用ノード数は，インクルードファイル prame4 で定義したグループ数とグループ内ノード数の積（npey×npex）に従う必要がある．

（1）ＮＱＳへの投入

例　　**%qsub** *go_h.014p*　　[RET]

（2）テスト実行に用いたプログラム動作環境

計算機 paragon (OS UNIX)	作業前	作業後
言語	fortran77	fortran77
並列化ライブラリ	nx	nx
コンパイラ	if77	if77
最適化オプション	-O2 -Mconcur	-O2 -Mconcur
実行時オプション	-plk -sz 7	-plk -sz *NODES* -mbf 5242880

4．5　まとめ

npex を 1 から 2 とした場合，サブルーチン schredi,ionpo4 とも使用したノード数以上の性能が得られている．サブルーチン schredi に関しては，schredi do 142 の記述変更が原因である．また，サブルーチン ionpo4 に関しては，大きさ約 5MB の３つの配列がそれぞれ７分割されたことによる，メモリのアクセス効率向上が原因と考えられる．計算結果については，完全な一致は得られなかった．これは，演算順序の変更によるものと考えられる．

７並列実行における並列化の効率は非常に良く，使用ノード総数も58並列程度までは良好である．56 並列程度でほぼ性能が飽和したのは，FFT 部分の逐次計算が主たる原因である．今後，QQQF が対象とする粒子の増加に対しては粒子数の増加が起こる．このとき，npex を 4 もしくは 8 ととれば十分な効率が得られ，したがって本作業における高速化は十分に達成できたと考える．

```
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          maxtepe=20,kazu=7,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1,ndatas=20,ndata1=500, akousi=5.333D-10,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat= 0.1000000D-16,gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,hbar= 1.055D-34,
     &          epusi0=0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida= 1024, naida= 10,
     &          kstart= 1,nspec= 32768,mspec= 15,
     &          situk=6.4931916D-26,situxe=6.4931916D-26,
     &          situka=6.4931916D-26,sigkk=4.00D-10,
     &          epukk=1.625D-21*100.00D0,
     &          sigke=1.000D0*sigkk,epuke=epukk/500.0D0,
     &          cutke= 1.0000D0,boltu= 1.381D-23,
     &           nfine=311, nggai=10
     &           )
```

Fig.4.1 Original include file prame4.

```
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          npe=7,
C    &          maxtepe=20,kazu=7,nttsp=10000,nsku=1024,nvti=1,
     &          maxtepe=20,kazu=1,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1,  ndatas=20,ndata1=500, akousi=5.333D-10,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat = 0.1000000D-16,  gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,  hbar= 1.055D-34,
     &          epusi0 = 0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida  = 1024,           naida= 10,
     &          kstart = 1, nspec= 32768,mspec= 15,
     &          situk  = 6.4931916D-26,  situxe=6.4931916D-26,
     &          situka = 6.4931916D-26,  sigkk=4.00D-10,
     &          epukk  = 1.625D-21*100.00D0,
     &          sigke  = 1.000D0*sigkk,  epuke=epukk/500.0D0,
     &          cutke  = 1.0000D0,       boltu= 1.381D-23,
     &          nfine  = 311,            nggai=10
     &           )
```

Fig.4.2 Modified include file prame4.

```
      program QQQ
      implicit
     & double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

         call clear
         call allset
         nwstep=1
         call lasdat
         call gid22
         call edatas2
             :
c--------------------------------
        do 100 i=1,maxtep
          nwstep=i
          call try
          call calcu
  888     call datass
        do 1091 iiku2= 1,kazu
             :
 1091   continue
  100   continue
c--------------------------------
        call skaku

        stop
        end
```

Fig.4.3 Original main program.

```
      program QQQ
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
      include'fnx.h'
      common /paralell/ iam,nodes

      call gsync()
      iam=mynode()
      iam=iam+1

      len     = isized
      msgid1  = 10
      msgid2  = 20
      if( iam .eq. 1) then
         read(*,nam)
      call csend(msgid1,enepara,len,-1,0)
      call csend(msgid2,enemitu,len,-1,0)
      else
      call crecv(msgid1,enepara,len)
      call crecv(msgid2,enemitu,len)
      end if

         call clear
         call allset
         nwstep=1
         call lasdat
         call gid22
         call edatas2
             :
       do 100 i=1,maxtep
         call try
         call calcu
  888    call datass
  100   continue

       call skaku

       stop
       end
```

Fig.4.4 Modified main program.

```
      subroutine allset
            :
      do 2 kkkazu= 1,kazu
        read(1,9000)n1,xxx,yyy,vxxx,vyyy
        pppx(kkkazu)=xxx
        pppy(kkkazu)=yyy
        vvvx(kkkazu)=vxxx
        vvvy(kkkazu)=vyyy
    2 continue

      return
      end
```

Fig.4.5 Original subroutine allset.

```
      subroutine allset
            :
      include'fnx.h'
      common /paralell/ iam,nodes
      dimension wrk1g(kazu*npe),wrk2g(kazu*npe)
     &          ,wrk3g(kazu*npe),wrk4g(kazu*npe)

      len    = isized * kazu*npe
      msgid1 = 100
      msgid2 = 200
      msgid3 = 300
      msgid4 = 400

      if( iam .eq. 1) then
      do 2 kkkazu= 1,kazu*npe
           read(1,9000)n1,xxx,yyy,vxxx,vyyy
           wrk1g(kkkazu)=xxx
           wrk2g(kkkazu)=yyy
           wrk3g(kkkazu)=vxxx
           wrk4g(kkkazu)=vyyy
    2 continue

      call csend(msgid1,wrk1g,len,-1,0)
      call csend(msgid2,wrk2g,len,-1,0)
      call csend(msgid3,wrk3g,len,-1,0)
      call csend(msgid4,wrk4g,len,-1,0)
      else
      call crecv(msgid1,wrk1g,len)
      call crecv(msgid2,wrk2g,len)
      call crecv(msgid3,wrk3g,len)
      call crecv(msgid4,wrk4g,len)
      end if

         do 740 ii = 1,kazu*npe
            pppx(ii)=wrk1g(ii)
            pppy(ii)=wrk2g(ii)
            vvvx(ii)=wrk3g(ii)
            vvvy(ii)=wrk4g(ii)
  740 continue

      return
      end
```

Fig.4.6 Modified subroutine allset.

```
      subroutine gid22
           :
           :
      do 50 i200=1,kazu
      do 40 j1=1,nyygmax
      do 30 i1=1,nxxgmax
         read(37,999)iiix,iiiy,vrexp,sss
         vwgr(i1,j1,i200) = vrexp
         vclo(i1,j1,i200) = vrexp*vrexp + sss*sss
         usovwgr(i1,j1,i200)= sss
         uvwbe(i1,j1,i200)= 0.0000D0
30    continue
40    continue
50    continue
           :
      return
      end
```

Fig.4.7 Original subroutine gid22.

```
      subroutine gid22
      include'fnx.h'
      common /paralell/ iam,nodes
      dimension wrk1g(nxxgmax,nyygmax,kazu*npe)
     &         ,wrk2g(nxxgmax,nyygmax,kazu*npe)
     &         ,wrk3g(nxxgmax,nyygmax,kazu*npe)
     &         ,wrk4g(nxxgmax,nyygmax,kazu*npe)
           :
           :
      len=nxxgmax*nyygmax*kazu*npe*isized
      msgid1=100
      msgid2=200
      msgid3=300
      msgid4=400

      if( iam .eq. 1) then
      do 50 i200=1,kazu*npe
      do 40 j1=1,nyygmax
      do 30 i1=1,nxxgmax
         read(37,999)iiix,iiiy,vrexp,sss
         wrk1g(i1,j1,i200)=vrexp
         wrk2g(i1,j1,i200)=vrexp*vrexp + sss*sss
         wrk3g(i1,j1,i200)=sss
         wrk4g(i1,j1,i200)=0.0000D0
30    continue
40    continue
50    continue

      call csend(msgid1,wrk1g,len,-1,0)
      call csend(msgid2,wrk2g,len,-1,0)
      call csend(msgid3,wrk3g,len,-1,0)
      call csend(msgid4,wrk4g,len,-1,0)
      else
      call crecv(msgid1,wrk1g,len)
      call crecv(msgid2,wrk2g,len)
      call crecv(msgid3,wrk3g,len)
      call crecv(msgid4,wrk4g,len)
      end if

      do 51 i200=1,kazu
         is=kazu*(iam-1)+i200
      do 41 j1=1,nyygmax
      do 31 i1=1,nxxgmax
         vwgr(i1,j1,i200)    = wrk1g(i1,j1,is)
         vclo(i1,j1,i200)    = wrk2g(i1,j1,is)
         usovwgr(i1,j1,i200)= wrk3g(i1,j1,is)
         uvwbe(i1,j1,i200)   = wrk4g(i1,j1,is)
31    continue
41    continue
51    continue

      return
      end
```

Fig.4.8 Modified subroutine gid22.

```
      subroutine try
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

        do 100 i=1,maxtepe
        nwstepe=i
9000    format(1i7)

          call nlconsp
          call elecden
          call data

           do 45 i45 = 1,kazu
                :
           call schredi(i45)
                :
 45       continue
100     continue

        return
        end
```

Fig.4.9 Original subroutine try.

```
      subroutine try
                  :
      include'fnx.h'
      common /paralell/ iam,nodes
      common /schredicom/
     &        nstatet(npe*kazu)
     &       ,vclot(nxxgmax,nyygmax,npe*kazu)
      dimension
     &        tmp1(npe*kazu)
     &       ,tmp2(nxxgmax,nyygmax,npe*kazu)

        do 100 i=1,maxtepe

        nwstepe=i
        call nlconsp
        call elecden
        call data

        do 440 ii=1,kazu*npe
          nstatet(ii)=0
         do 460 jjpo= 1,nyygmax
         do 480 iipo= 1,nxxgmax
          vclot(iipo,jjpo,ii)=0.0d0
480   continue
460   continue
440   continue

      do 540 ii=1,kazu
         is=(iam-1)*kazu+ii
         nstatet(is)=nstatet(is)+nstate(ii)
        do 560 jjpo= 1,nyygmax
        do 580 iipo= 1,nxxgmax
         vclot(iipo,jjpo,is)
        =vclot(iipo,jjpo,is)+vclo(iipo,jjpo,ii)
580   continue
560   continue
540   continue

      nwd1=kazu*npe
      nwd2=nxxgmax*nyygmax*kazu*npe
      call gdsum(nstatet,nwd1,tmp1)
      call gdsum(vclot,nwd2,tmp2)

          do 45 i45 = 1,kazu
                  :
          call schredi(i45)
                  :
 45    continue
100    continue

      return
      end
```

Fig.4.10 Modified subroutine try.

```
      subroutine schredi(il00)
      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'
            :
            :
        do 142 ibetue= 1,kazu
        if (ibetue .eq. il00) goto 142
        kt2em= nelep(nstate(il00),nstate(ibetue))
        do 140 nnpo= 1,nyygmax
        do 130 mmpo= 1,nxxgmax
           const=vclo(mmpo,nnpo,ibetue)
            :
130   continue
140   continue
142   continue
            :
         if (mod(nwstepe,20) .eq. 1) then
     do 989 i989= 1,kazu
            :
            :
989  continue
            :
         end if
            :
         xppt   = pppx(il00)
            :
         do 120 jjpo= 1,nyygmax
            :
            :
120     continue
            :
      return
      end
```

Fig.4.11 Original subroutine schredi.

```
      subroutine schredi(il00)
      implicit double precision(a-h,o-z)
      include'fnx.h'
      include'prame4'
      include'QQQ.h'
      common /paralell/ mynode,nodes
      common /schredicom/
     &         nstatet(npe*kazu)
     &        ,vclot(nxxgmax,nyygmax,npe*kazu)

        is=kazu*(mynode-1)+il00

                :
        do 142 ibetue= 1,kazu*npe
        if (ibetue .eq. is) goto 142

        kt2em= nelep(nstatet(is),nstatet(ibetue))
        do 140 nnpo= 1,nyygmax
        do 130 mmpo= 1,nxxgmax
           const=vclot(mmpo,nnpo,ibetue)
                :
130   continue
140   continue
142   continue
                :
         if (mod(nwstepe,20) .eq. 1) then
                :
     do 989 i989= 1,kazu*npe
                :
989  continue
                :
       endif
                :
                :
        xppt  = pppx(is)
                :
        do 120 jjpo= 1,nyygmax
                :
                :
120    continue

     return
     end
```

Fig.4.12 Modified subroutine schredi.

```
      subroutine calcu
                :
      do 9090 ionum= 1,kazu
                :
         xmolpo= pppx(ionum)
         ymolpo= pppy(ionum)

         do 1000 imol1= 1,kazu
         if (imol1 .eq. ionum) goto 1000
                :
                t126= dipox(ionum)+dipox(imol1)
                :
1000    continue
9090  continue

      do 1101 i101=1,kazu
        call interpo(i101)
1101  continue

      do 1100 i11=1,kazu

      call ionpo4(i11)

      akx= vvvx(il1)
      aky= vvvy(il1)
      vvvx(il1)=vvvx(il1)+aaax(il1)*deltt
      vvvy(il1)=vvvy(il1)+aaay(il1)*deltt
      vvpx= akx+aaax(il1)*deltt*0.500D0
      vvpy= aky+aaay(il1)*deltt*0.500D0
      subpox=pppx(il1)+vvvx(il1)*deltt
      subpoy=pppy(il1)+vvvy(il1)*deltt
      pppx(il1)=subpox
      pppy(il1)=subpoy

1100  continue

      return
      end
```

Fig.4.13 Original subroutine calcu.

```
      subroutine calcu
      include'fnx.h'
            :
c.para <<
      common /paralell/ iam,nodes
      common /ionpo4com/ vaftmp(npe*kazu,nfine,nfine)
     &,vafpxtmp(npe*kazu,nfine,nfine),vafpytmp(npe*kazu,nfine,nfine)

      dimension tmp1(npe*kazu,nfine,nfine)
     &,tmp2(npe*kazu,nfine,nfine),tmp3(npe*kazu,nfine,nfine)

      dimension
     & aaaxt(kazu*npe),aaayt(kazu*npe),vvvxt(kazu*npe)
     &,vvvyt(kazu*npe),pppxt(kazu*npe),pppyt(kazu*npe)
     &,dipoxtmp(kazu*npe),enep2(kazu) ,tmp(kazu*npe)
c.para >>

         nwd =kazu*npe
         do 7750 ii= 1,kazu*npe
            dipoxtmp(ii)=0.0d0
 7750  continue
         do 7710 ii = 1,kazu
            iis=kazu*(iam-1)+ii
            dipoxtmp(iis)=dipox(ii)
 7710  continue
      call gdsum(dipoxtmp,nwd,tmp)



      do 9090 ionum= 1,kazu
        is=kazu*(iam-1)+ionum
            :
        xmolpo = pppx(is)
        ymolpo = pppy(is)
            :
         do 1000 imol1= 1,kazu*npe
         if (imol1 .eq. is) goto 1000
            :
             t126= dipoxtmp(is)+dipoxtmp(imol1)
            :
 1000  continue
 9090  continue

      do 1101 i101=1,kazu
        call interpo(i101)
 1101  continue
            :
            :
      return
      end
```

Fig.4.14 Modified subroutine calcu.(1/3)

```
      subroutine calcu
      include'fnx.h'
              :
              :
c-------------------------------------
         nwdvaf = nfine*nfine*kazu*npe
         do 750 ii= 1,kazu*npe
         do 760 j  = 1,nfine
         do 770 i  = 1,nfine
            vaftmp(ii,i,j)  =0.0d0
            vafpxtmp(ii,i,j)=0.0d0
            vafpytmp(ii,i,j)=0.0d0
  770  continue
  760  continue
  750  continue

         do 710 ii = 1,kazu
            iis=kazu*(iam-1)+ii
         do 720 j  = 1,nfine
         do 730 i  = 1,nfine
            vaftmp(iis,i,j)  =vaf(ii,i,j)
            vafpxtmp(iis,i,j)=vafpx(ii,i,j)
            vafpytmp(iis,i,j)=vafpy(ii,i,j)
  730  continue
  720  continue
  710  continue
      call gdsum(vaftmp,nwdvaf,tmp1)
      call gdsum(vafpxtmp,nwdvaf,tmp2)
      call gdsum(vafpytmp,nwdvaf,tmp3)
c-------------------------------------
       do 1091 ii= 1,kazu
              :
 1091   continue

      do 1100 ill=1,kazu
       is=kazu*(iam-1)+ill
       call ionpo4(ill)

      aaax(is)= (xforce(ill)+subefx+ffqqq(ill)+fffcha)/situk
      aaay(is)= (yforce(ill)+subefy )/situk
      akx= vvvx(is)
      aky= vvvy(is)
      vvvx(is)=vvvx(is)+aaax(is)*deltt
      vvvy(is)=vvvy(is)+aaay(is)*deltt
      vvpx= akx+aaax(is)*deltt*0.500D0
      vvpy= aky+aaay(is)*deltt*0.500D0
      subpox=pppx(is)+vvvx(is)*deltt
      subpoy=pppy(is)+vvvy(is)*deltt
      pppx(is)=subpox
      pppy(is)=subpoy

 1100 continue
              :
              :
      return
      end
```

Fig.4.14 Modified subroutine calcu.(2/3)

```
      subroutine calcu
      include'fnx.h'
              :
              :
              :
        nwd = kazu*npe
        do 960 ii=1,kazu*npe
           aaaxtg(ii)=0.0d0
           aaaytg(ii)=0.0d0
           vvvxtg(ii)=0.0d0
           vvvytg(ii)=0.0d0
           pppxtg(ii)=0.0d0
           pppytg(ii)=0.0d0
960     continue

        do 975 ii=1,kazu
           i=kazu*(iam-1)+ii
           aaaxtg(i)=aaax(i)
           aaaytg(i)=aaay(i)
           vvvxtg(i)=vvvx(i)
           vvvytg(i)=vvvy(i)
           pppxtg(i)=pppx(i)
           pppytg(i)=pppy(i)
975  continue

     call gdsum(aaaxtg,nwd,aaaxt)
     call gdsum(aaaytg,nwd,aaayt)
     call gdsum(vvvxtg,nwd,vvvxt)
     call gdsum(vvvytg,nwd,vvvyt)
     call gdsum(pppxtg,nwd,pppxt)
     call gdsum(pppytg,nwd,pppyt)

        do 987 ii=1,kazu*npe
           aaax(ii)=aaaxtg(ii)
           aaay(ii)=aaaytg(ii)
           vvvx(ii)=vvvxtg(ii)
           vvvy(ii)=vvvytg(ii)
           pppx(ii)=pppxtg(ii)
           pppy(ii)=pppytg(ii)
987     continue

      return
      end
```

Fig.4.14 Modified subroutine calcu.(3/3)

```
      subroutine ionpo4(il00)
                :
            xmolpo= pppx(il00)
            ymolpo= pppy(il00)

        do 300 iother=1,kazu
                :
            xgenpo= vafpx(iother,i,j)
            ygenpo= vafpy(iother,i,j)
            totvaf= vaf(iother,i,j)
                :
               xemc= xemc + xsubme/rkyo3*totvaf
                :
300     continue
                :
      return
      end
```

Fig.4.15 Original subroutine ionpo4.

```
      subroutine ionpo4(il00)
                :
      include'fnx.h'
      common /paralell/ iam,nodes
      common /ionpo4com/ vaftmp(npe*kazu,nfine,nfine)
     &,vafpxtmp(npe*kazu,nfine,nfine),vafpytmp(npe*kazu,nfine,nfine)

      is=kazu*(iam-1)+il00
      xmolpo= pppx(is)
      ymolpo= pppy(is)

        do 300 iother=1,kazu*npe
                :
            xgenpo= vafpxtmp(iother,i,j)
            ygenpo= vafpytmp(iother,i,j)
            totvaf= vaftmp(iother,i,j)
                :
             zxemc= zxemc + xsubme/rkyo3*totvaf
                :
300     continue

         xemc= -zxemc*econst

      return
      end
```

Fig.4.16 Modified subroutine ionpo4.

```
      subroutine skaku

      implicit double precision(a-h,o-z)
      include'prame4'
      include'QQQ.h'

     do 2000 maa=1,maxtep
     do 1000 i11=1,kazu
       write(3,9199) maa,dpppx(i11,maa),dpppy(i11,maa),
    &              dvvvx(i11,maa),dvvvy(i11,maa),
    &              deeex(i11,maa),deeey(i11,maa),
    &              dpxre(i11,maa),dpyre(i11,maa)
       write(39+i11,9099)  dundou(i11,maa),
    &                      denepii(i11,maa),
    &                      denepei(i11,maa)
1000  continue
2000  continue

     do 3000 i30=1,nttsp
      do 1001 i11=1,kazu
       write(38,9299) i30,denergy(i11,i30),denekin(i11,i30),
    &                    denepot(i11,i30)
       write(34,9399) ddnti(i30),dsoukan(i30),dusok(i30)
1001    continue
3000    continue

     do 4004 mn=1,kazu
     do 4003 k=1,nvti
     do 4002 j=1,nyygmax
     do 4001 i=1,nxxgmax
        nxygm= nyygmax*(j-1)+i
      write(15,9533)i,j,dvclo(k,mn,nxygm)
      write(14,9433)i,j,dvwgr(k,mn,nxygm),dusovwgr(k,mn,nxygm)
4001   continue
4002   continue
4003   continue
4004   continue

      return
      end
```

Fig.4.17 Original subroutine skaku.

```
      subroutine skaku
            :
      include'fnx.h'
      common /paralell/ iam,nodes
      dimension
     &  dpppxg(kazu*npe,maxtep)    ,dpppyg(kazu*npe,maxtep)
     & ,dvvvxg(kazu*npe,maxtep)    ,dvvvyg(kazu*npe,maxtep)
     & ,deeexg(kazu*npe,maxtep)    ,deeeyg(kazu*npe,maxtep)
     & ,dpxreg(kazu*npe,maxtep)    ,dpyreg(kazu*npe,maxtep)
     & ,dundoug(kazu*npe,maxtep)   ,denepiig(kazu*npe,maxtep)
     & ,denepeig(kazu*npe,maxtep) ,denergyg(kazu*npe,nttsp)
     & ,deneking(kazu*npe,nttsp)   ,denepotg(kazu*npe,nttsp)
     & ,dvclog(nvti,kazu*npe,nsku),dvwgrg(nvti,kazu*npe,nsku)
     & ,dusovwgrg(nvti,kazu*npe,nsku)
     & ,coefljg(kazu*npe,kazu*npe,maxtep)
     & ,coefclg(kazu*npe,kazu*npe,maxtep)
     & ,dpppxt(kazu*npe,maxtep)    ,dpppyt(kazu*npe,maxtep)
     & ,dvvvxt(kazu*npe,maxtep)    ,dvvvyt(kazu*npe,maxtep)
     & ,deeext(kazu*npe,maxtep)    ,deeeyt(kazu*npe,maxtep)
     & ,dpxret(kazu*npe,maxtep)    ,dpyret(kazu*npe,maxtep)
     & ,dundout(kazu*npe,maxtep)   ,denepiit(kazu*npe,maxtep)
     & ,denepeit(kazu*npe,maxtep) ,denergyt(kazu*npe,nttsp)
     & ,denekint(kazu*npe,nttsp)   ,denepott(kazu*npe,nttsp)
     & ,dvclot(nvti,kazu*npe,nsku),dvwgrg(nvti,kazu*npe,nsku)
     & ,dusovwgrt(nvti,kazu*npe,nsku)
     & ,coefljt(kazu*npe,kazu*npe,maxtep)
     & ,coefclt(kazu*npe,kazu*npe,maxtep)

       do 1500 i11=1,kazu*npe
       do 2500 maa=1,maxtep
         dpppxg(i11,maa)  =0.0d0
         dpppyg(i11,maa)  =0.0d0
         dvvvxg(i11,maa)  =0.0d0
         dvvvyg(i11,maa)  =0.0d0
         deeexg(i11,maa)  =0.0d0
         deeeyg(i11,maa)  =0.0d0
         dpxreg(i11,maa)  =0.0d0
         dpyreg(i11,maa)  =0.0d0
         dundoug(i11,maa) =0.0d0
         denepiig(i11,maa)=0.0d0
         denepeig(i11,maa)=0.0d0
2500   continue
       do 3500 i30=1,nttsp
          denergyg(i11,i30)=0.0d0
          deneking(i11,i30)=0.0d0
          denepotg(i11,i30)=0.0d0
3500   continue
       do 4504 jj=1,nsku
       do 4503 k=1,nvti
          dvclog(k,i11,jj)   =0.0d0
          dvwgrg(k,i11,jj)   =0.0d0
          dusovwgrg(k,i11,jj)=0.0d0
4503   continue
4504   continue
       do 5504 istep=1,maxtep
       do 5503 imoll=1,kazu*npe
          coefljg(imoll,i11,istep)=0.0d0
          coefclg(imoll,i11,istep)=0.0d0
5503   continue
5504   continue
1500   continue
```

Fig.4.18 Modified subroutine skaku.(1/3)

```
                :
      do 1550 ill=1,kazu
      iis=kazu*(iam-1)+ill
      do 2550 maa=1,maxtep
        dpppxg(iis,maa)  =dpppx(ill,maa)
        dpppyg(iis,maa)  =dpppy(ill,maa)
        dvvvxg(iis,maa)  =dvvvx(ill,maa)
        dvvvyg(iis,maa)  =dvvvy(ill,maa)
        deeexg(iis,maa)  =deeex(ill,maa)
        deeeyg(iis,maa)  =deeey(ill,maa)
        dpxreg(iis,maa)  =dpxre(ill,maa)
        dpyreg(iis,maa)  =dpyre(ill,maa)
        dundoug(iis,maa) =dundou(ill,maa)
        denepiig(iis,maa)=denepii(ill,maa)
        denepeig(iis,maa)=denepei(ill,maa)
2550  continue

      do 3550 i30=1,nttsp
         denergyg(iis,i30)=denergy(ill,i30)
         deneking(iis,i30)=denekin(ill,i30)
         denepotg(iis,i30)=denepot(ill,i30)
3550  continue

      do 4554 jj=1,nsku
      do 4553 k=1,nvti
         dvclog(k,iis,jj)   =dvclo(k,ill,jj)
         dvwgrg(k,iis,jj)   =dvwgr(k,ill,jj)
         dusovwgrg(k,iis,jj)=dusovwgr(k,ill,jj)
4553  continue
4554  continue

      do 5554 istep=1,maxtep
      do 5553 imoll=1,kazu*npe
         coefljg(imoll,iis,istep)=coeflj(imoll,ill,istep)
         coefclg(imoll,iis,istep)=coefcl(imoll,ill,istep)
5553  continue
5554  continue
1550  continue

        call gdsum(dpppxg,nwdmaa,dpppxt)
        call gdsum(dpppyg,nwdmaa,dpppyt)
        call gdsum(dvvvxg,nwdmaa,dvvvxt)
        call gdsum(dvvvyg,nwdmaa,dvvvyt)
        call gdsum(deeexg,nwdmaa,deeext)
        call gdsum(deeeyg,nwdmaa,deeeyt)
        call gdsum(dpxreg,nwdmaa,dpxret)
        call gdsum(dpyreg,nwdmaa,dpyret)
        call gdsum(dundoug,nwdmaa,dundout)
        call gdsum(denepiig,nwdmaa,denepiit)
        call gdsum(denepeig,nwdmaa,denepeit)
        call gdsum(denergyg,nwdnttsp,denergyt)
        call gdsum(deneking,nwdnttsp,denekint)
        call gdsum(denepotg,nwdnttsp,denepott)
        call gdsum(dvclog,nwdnsku,dvclot)
        call gdsum(dvwgrg,nwdnsku,dvwgrt)
        call gdsum(dusovwgrg,nwdnsku,dusovwgrt)
        call gdsum(coefljg,nwdistp,coefljt)
        call gdsum(coefclg,nwdistp,coefclt)
                :
```

Fig.4.18 Modified subroutine skaku.(2/3)

```
                              :
      if (iam .eq. 1) then
      do 2000 maa=1,maxtep
      do 1000 i11=1,kazu*npe
        write(3,9199) maa,dpppxg(i11,maa),dpppyg(i11,maa),
     &               dvvvxg(i11,maa),dvvvyg(i11,maa),
     &               deeexg(i11,maa),deeeyg(i11,maa),
     &               dpxreg(i11,maa),dpyreg(i11,maa)

        write(39+i11,9099)  dundoug(i11,maa),
     &                      denepiig(i11,maa),
     &                      denepeig(i11,maa)
1000  continue
2000  continue

      do 3000 i30=1,nttsp
      do 1001 i11=1,kazu*npe
        write(38,9299) i30,denergyg(i11,i30),deneking(i11,i30),
     &                     denepotg(i11,i30)
        write(34,9399) ddnti(i30),dsoukan(i30),dusok(i30)
1001  continue
3000  continue

      do 4004 mn=1,kazu*npe
      do 4003 k=1,nvti
      do 4002 j=1,nyygmax
      do 4001 i=1,nxxgmax
         nxygm= nyygmax*(j-1)+i
       write(15,9533)i,j,dvclog(k,mn,nxygm)
       write(14,9433)i,j,dvwgrg(k,mn,nxygm),dusovwgrg(k,mn,nxygm)
4001   continue
4002   continue
4003   continue
4004   continue

      do 1900 istep=1,maxtep
      do 1950 ionum=1,kazu*npe
      do 1970 imol1=1,kazu*npe
         if (imol1 .eq. ionum) goto 1970
              write(52,9696)coefljg(imol1,ionum,istep)
     &                      ,coefclg(imol1,ionum,istep)
1970  continue
1950  continue
1900  continue
      end if

       return
       end
```

Fig.4.18 Modified subroutine skaku.(3/3)

```
c     parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=500,
c    &          maxtepe=20,kazu=1,nttsp=10000,nsku=1024,nvti=1,
c    &          intgra=1,  ndatas=20,ndata1=500, akousi=5.333D-10,
     &          npex=2,    npey=7,npe=7,
c    &          npex=4,    npey=7,npe=7,
c    &          npex=8,    npey=7,npe=7,
c    &          npex=16,   npey=7,npe=7,
c    &          npex=32,   npey=7,npe=7,
c test run: timestep & number of node
      parameter(nxxmax=8,nyymax=8,nxxm11=24,nyym11=24,maxtep=10,
     &          maxtepe=20,kazu=1,nttsp=10000,nsku=1024,nvti=1,
     &          intgra=1, ndatas=20,ndata1=10, akousi=5.333D-10,
     &          isizei =4,isizer=4,isized=8,
     &          firtmp=0.02000D0,ncutof=5,deltt=2.00000D-16,
     &          intev1=1,intev2=1,numene=1,
     &          nbunp=10000, nbunp1=10000,
     &          nxxgmax=32,nyygmax=32, nsisuu=5,gbaip=6.00D0,
     &          deltat = 0.1000000D-16,  gridlen= 32.32959D-10,
     &          eweight= 0.9110000D-30,  hbar= 1.055D-34,
     &          epusi0 = 0.88541878D-11, esoryo=0.16021892D-18,
     &          kaida  = 1024,           naida= 10,
     &          kstart = 1, nspec= 32768,mspec= 15,
     &          situk  = 6.4931916D-26,  situxe=6.4931916D-26,
     &          situka = 6.4931916D-26,  sigkk=4.00D-10,
     &          epukk  = 1.625D-21*100.00D0,
     &          sigke  = 1.000D0*sigkk,  epuke=epukk/500.0D0,
     &          cutke  = 1.0000D0,       boltu= 1.381D-23,
     &          nfine  = 311,            nggai=10,
     &          constdh = deltat/hbar
     &           )
```

Fig.4.19 Modified include file prame4.

```
      program QQQ
            :
      include'fnx.h'
      common /paralell/ iam(2),nodes
      dimension iclstr(2)

        iclstr(1)= npex
        iclstr(2)= npey
        iam(1)   = mod(mynode(),iclstr(1))+1
        iam(2)   = int(mynode()/iclstr(1))+1
        nodes=numnodes()

      if( mynode() .eq. 0) then
         read(*,nam)
      call csend(msgid1,enepara,len,-1,0)
      else
      call crecv(msgid1,enepara,len)
      end if
            :
         call clear
         call allset
         nwstep=1
         call lasdat
         call gid22
         call edatas2

       do 100 i=1,maxtep
         nwstep=i
         call try
         call calcu
888      call datass
100    continue

       call skaku
            :
       stop
       end
```

Fig.4.20 Modified main program.

```
      subroutine try
                :
      include'fnx.h'
      common /paralell/ iam(2),nodes
      common /schredicom/
     &        nstatet(npe*kazu)
     &       ,vclot(nxxgmax,nyygmax,npe*kazu)
      dimension
     &        dnstatet(npe*kazu)
     &       ,tmp1(npe*kazu)
     &       ,tmp2(nxxgmax,nyygmax,npe*kazu)

        do 100 i=1,maxtepe
        nwstepe=i

        call nlconsp
        call elecden
        call data

        do 440 ii=1,kazu*npe
         do 460 jjpo= 1,nyygmax
         do 480 iipo= 1,nxxgmax
          vclot(iipo,jjpo,ii)=0.0d0
  480   continue
  460   continue
  440   continue

        do 540 ii=1,kazu
           is=(iam(2)-1)*kazu+ii
          do 560 jjpo= 1,nyygmax
          do 580 iipo= 1,nxxgmax
           vclot(iipo,jjpo,is)=vclot(iipo,jjpo,is)+vclo(iipo,jjpo,ii)
  580   continue
  560   continue
  540   continue
        nwd2=nxxgmax*nyygmax*kazu*npe
        call test(vclot,tmp2,nwd2,npex,npey,2)

            do 45 i45 = 1,kazu
                  :
            call schredi(i45)
                  :
   45   continue
  100   continue

        return
        end
```

Fig.4.21 Modified subroutine try.

```
      subroutine schredi(i100)
                :
      do 2 kpoyy = 1,nyygmax
      do 1 kpoxx = 1,nxxgmax
        vwcons(kpoxx,kpoyy,i100) = 0.0000D0
    1 continue
    2 continue

      do 142 ib= 1,kazu*npe
      if (ib .eq. is) goto 142
      kt= nelep(nstatet(is),nstatet(ib))

      do 140 n= 1,nyygmax
      do 130 m= 1,nxxgmax
      do 122 j= 1,nyygmax
      do 112 i= 1,nxxgmax

          nx= i - m + 32
          ny= j - n + 32

        vwcons(i,j,i100) = vwcons(i,j,i100)
     & + vclot(m,n,ib)*eeclo2(nx,ny,kt)
  112 continue
  122 continue
  130 continue
  140 continue

  142 continue
                :
      return
      end
```

Fig.4.22 Subroutine schredi do 142.

```
      subroutine schredi(i100)
                :
      do 2 kpoyy = 1,nyygmax
      do 1 kpoxx = 1,nxxgmax
        vwcons(kpoxx,kpoyy,i100) = 0.0000D0
    1 continue
    2 continue

      do 142 ib= 1,kazu*npe
      if (ib .eq. is) goto 142
      kt= nelep(nstatet(is),nstatet(ib))

      do 122 j= 1,nyygmax
      do 112 i= 1,nxxgmax
         const= 0.0d0
     do 140 n= 1,nyygmax
     do 130 m= 1,nxxgmax
        const= const + vclot(m,n,ib)
&                * eeclo2(nx(i,m),ny(j,n),kt)
  130       continue
  140       continue
           vwcons(i,j,i100)=vwcons(i,j,i100)+const
  112 continue
  122 continue

  142 continue
                :
      return
      end
```

Fig.4.23 Modified subroutine schredi do 142.

```
      subroutine schredi(il00)
          :
      include'fnx.h'
      common /paralell/ iam(2),nodes
      common /schredicom/ nstatet(npe*kazu)
     &                   ,vclot(nxxgmax,nyygmax,npe*kazu)
      common /indxcom/ nxa(nxxgmax,nxxgmax),nya(nyygmax,nyygmax)
     &                ,is,istart,iend
      dimension tmp(nxxgmax,nyygmax,nxxgmax,nyygmax)
     &,wrk(nxxgmax,nyygmax) ,wrkbf(nxxgmax,nyygmax)
          :
        if(nwstepe.eq.1 .and. nwstep.eq.1) then
        is=kazu*(iam(2)-1)+il00
        istart=nyygmax/npex*(iam(1)-1)+1
        iend  =nyygmax/npex*iam(1)
          :
        end if
          :
        do 142 ib= 1,kazu*npe
        if (ib .eq. is) goto 142
        kt= nelep(nstatet(is),nstatet(ib))

        do 122 jp= istart,iend
        do 112 ip= 1,nxxgmax
           const=0.0d0
        do 140 np= 1,nyygmax
        do 130 mp= 1,nxxgmax
           const = const + vclot(mp,np,ib)
     &                  * eeclo2(nxa(ip,mp),nya(jp,np),kt)
  130   continue
  140   continue
        vwcons(ip,jp,il00)=vwcons(ip,jp,il00)+const
  112   continue
  122   continue
  142   continue

        ndim=nxxgmax*nyygmax
        call test(vwcons,wrkbf,ndim,npex,npey,1)
          :
          :
      return
      end
```

Fig.4.24 Modified subroutine schredi do 142.

```
      subroutine calcu
                        :
      dimension  pppgl(kazu*npe,6) ,pppglbf(kazu*npe,6)
     &,dipoxtmp(kazu*npe),tmp(kazu*npe)
                        :
         do 7750 ii= 1,kazu*npe
7750  continue
         do 7710 ii = 1,kazu
            iis=kazu*(iam(2)-1)+ii
7710  continue
      call test(dipoxtmp,tmp,nwdfl,npex,npey,2)

      do 9090 ionum= 1,kazu
         is=kazu*(iam(2)-1)+ionum
                        :
9090  continue
                        :
      do 1100 i11=1,kazu
       is=kazu*(iam(2)-1)+i11
        call ionpo4(i11)
1100  continue

        do 975 ii=1,kazu
           i=kazu*(iam(2)-1)+ii
           pppgl(i,1)=aaax(i)
           pppgl(i,2)=aaay(i)
                        :
 975 continue
     ndim=kazu*npe*6
     call test(pppgl,pppglbf,ndim,npex,npey,2)

        do 987 ii=1,kazu*npe
           aaax(ii)=pppgl(ii,1)
           aaay(ii)=pppgl(ii,2)
                        :
 987    continue
                        :
      return
      end
```

Fig.4.25 Modified subroutine calcu.

```
      subroutine ionpo4(il00)
                :
      include'fnx.h'
      common /paralell/ iam,nodes
      common /ionpo4com/ vaftmp(npe*kazu,nfine,nfine)
     &,vafpxtmp(npe*kazu,nfine,nfine),vafpytmp(npe*kazu,nfine,nfine)

      is=kazu*(iam-1)+il00
      xmolpo= pppx(is)
      ymolpo= pppy(is)
      zxemc = 0.000D0
        do 300 iother=1,kazu*npe
                :
            xgenpo= vafpxtmp(iother,i,j)
            ygenpo= vafpytmp(iother,i,j)
            totvaf= vaftmp(iother,i,j)
                :
             zxemc= zxemc + xsubme/rkyo3*totvaf
                :
300     continue

         xemc= -zxemc*econst

      return
      end
```

Fig.4.26 Subroutine ionpo4.

```
      subroutine ionpo4(i100)
      include'fnx.h'
            :
      common /paralell/ iam(2),nodes
      dimension dsmsg(3),drmsg(3)
     &,zxemc(npey),zyemc(npey),zpoteki(npey),iclstr(2),ndtop(2)
c.....................................................
      id=2
      if( id .eq. 1 ) irv=2
      if( id .eq. 2 ) irv=1
      nodes=numnodes()
      iclstr(1)= npex
      iclstr(2)= npey
      ndtop(1) = (mynode()/npex)*npex
      ndtop(2) = mynode()-(iam(2)-1)*npex
      len      = 3*8
c.....................................................
      do 1000 istep=1,npey
      zxemc(istep)  = 0.000D0
      xmolpo        = pppx(istep)
      ymolpo        = pppy(istep)

        do 300 iother=1,kazu
        do 200 j= 1,nfine
        do 100 i= 1,nfine

            totvaf= vaf(iother,i,j)
            xgenpo= vafpx(iother,i,j)
            ygenpo= vafpy(iother,i,j)
                  :
            zxemc(istep)= zxemc(istep) + xsubme*rkyo3*totvaf
                  :
  100   continue
  200   continue
  300   continue
 1000   continue
c.....................................................
c ---- Send message to target-node [ndsel]
      do 2000 istep=1,npey
         dsmsg(1)=zxemc(istep)
         ndsel=ndtop(id)+(istep-1)*iclstr(irv)
         msgid=mynode()
         call csend(msgid,dsmsg,len,ndsel,0)
 2000   continue

c ---- Receive & sum message
      xemc  = 0.000D0
      msgid=iam(1)
      do 4000 inode=0,npey-1
         ndsel=ndtop(id)+inode*iclstr(irv)
         msgid=ndsel
         call crecv(msgid,drmsg,len)
         xemc   = xemc  +drmsg(1)
 4000   continue
                  :
      return
      end
```

Fig.4.27 Modified subroutine ionpo4.

```
      subroutine ionpo4(i100)
            :
c.....................................................
            :
c.....................................................
      do 1000 istep=1,npey
            :
        do 300 iother=1,kazu
        do 200 j= iam(1),nfine,npex
        do 100 i= 1,nfine
            :
            :
  100   continue
  200   continue
  300   continue

      xemc  = zxemc(istep)
      call test(xemc,tmp1,1,npex,npey,1)
            :
 1000 continue
c.....................................................
c ---- Send message to target-node [ndsel]
            :
c ---- Receive & sum message
            :

      return
      end
```

Fig.4.28 Modified subroutine ionpo4.

```
      call test(dpppxg,dpppxt,nwdmaa,npex,npey,2)
      call test(dpppyg,dpppyt,nwdmaa,npex,npey,2)
      call test(dvvvxg,dvvvxt,nwdmaa,npex,npey,2)
      call test(dvvvyg,dvvvyt,nwdmaa,npex,npey,2)
      call test(deeexg,deeext,nwdmaa,npex,npey,2)
      call test(deeeyg,deeeyt,nwdmaa,npex,npey,2)
      call test(dpxreg,dpxret,nwdmaa,npex,npey,2)
      call test(dpyreg,dpyret,nwdmaa,npex,npey,2)
      call test(dundoug,dundout,nwdmaa,npex,npey,2)
      call test(denepiig,denepiit,nwdmaa,npex,npey,2)
      call test(denepeig,denepeit,nwdmaa,npex,npey,2)
      call test(dvclog,dvclot,nwdnsku,npex,npey,2)
      call test(dvwgrg,dvwgrt,nwdnsku,npex,npey,2)
      call test(dusovwgrg,dusovwgrt,nwdnsku,npex,npey,2)
      call test(coefljg,coefljt,nwdistp,npex,npey,2)
      call test(coefclg,coefclt,nwdistp,npex,npey,2)
      call test(denergyg,denergyt,nwdnttsp,npex,npey,2)
      call test(deneking,denekint,nwdnttsp,npex,npey,2)
      call test(denepotg,denepott,nwdnttsp,npex,npey,2)
```

Fig.4.29 Modified subroutine skaku.

```
QQQF.vpp_p/------ *.f, includefile,makefile
    |---- SH/
    |---- DATA/
```

```
QQQF.vpp_p/
Makefile   並列実行用 Makefile 記述例

QQQ.h      インクルードファイル (配列宣言など)
prame4     インクルードファイル (パラメータファイル)

*.f     以下, ソースファイル一覧
abspec.f  after.f   after1.f   allset.f  before.f  beforel.f
calcu.f   clear.f   conju.f    convr.f   data.f    datass.f
edatas2.f elecden.f eneaf.f    enegbe.f  fcnjxy.f  fcnvrx.f
fcnvry.f  fconjx.f  fconjy.f   fft.f     fftho.f   frcnvx.f
frcnvy.f  gid22.f   grids.f    ibitr.f   ifftenx.f iffteny.f
interpo.f invxfft.f invyfft.f  ionbou.f  ionpo4.f  ionpre.f
lasdat.f  nlconho.f nlconsp.f  qqq.f     rconv.f   rxfft.f
ryfft.f   schdiho.f schredi.f  skaku.f   test.f    try.f
tranini.f
```

```
SH/      nqs 投入用シェルスクリプト記述例
go_sh.014p              14 parallel
go_sh.028p              28 parallel
go_sh.056p              56 parallel
go_sh.112p             112 parallel
go_sh.224p             224 parallel
org.sh
make_nqs_sh
```

```
DATA/
stakxe.data   unit01 よりの入力データ
stawv.data    unit37 よりの入力データ
namlist       unit05 ネームリストファイル
```

Fig.4.30 File lists of the QQQF.paragonII_p.tar.Z.

```
#####################################################
# Makefile on paragon
#          Write Day : Wed May  1 09:03:44 1996
# % make [RET]
#####################################################
F77      = if77
MAKE     = Makefile
OPT      = -nx -O2 -Mconcur -Minfo=loop -Mneginfo=concur
LIBES    =
INCLUDES = prame4 QQQ.h
TERGET   = a.out.p

OBJS = \
      qqq.o  abspec.o  after.o  after1.o  allset.o\
      before.o  before1.o  calcu.o  clear.o  conju.o\
      convr.o  data.o  datass.o  edatas2.o  elecden.o\
      eneaf.o  enegbe.o  fcnjxy.o  fcnvrx.o  fcnvry.o\
      fconjx.o  fconjy.o  fft.o  fftho.o  gid22.o\
      frcnvx.o  frcnvy.o  grids.o  ifftenx.o  iffteny.o\
      interpo.o  invxfft.o  invyfft.o  ionbou.o  ionpo4.o\
      ionpre.o  lasdat.o  nlconho.o  nlconsp.o  rconv.o\
      rxfft.o  ryfft.o  schredi.o  skaku.o  isum2d.o dsum2d.o\
      dxytran.o isum2dnew.o dsum2dnew.o try.o  ibitr.o

.f.o     :
         $(F77) $(OPT) $(INCLUDES) -c $<

all      :$(OBJS)
         $(F77) -o $(TERGET) $(OPT) $(OBJS) $(LIBES)

clean    :
         rm -f $(OBJS)
```

Fig.4.31 Makefile for making load-module.

```
#!/bin/csh -f
#####################################################
# spp16-8 -eo
#@$-q spps16-i -eo
#####################################################
cd $HOME/QQQF

# read.data.test.14p
setenv FOR001 $HOME/QQQF/wkvfl/stakxe.data
setenv FOR037 $HOME/QQQF/wkvfl/stawv.data

# write.data.test.14p
#  check wave.data.test.14p last step
setenv FOR014 $HOME/QQQF/wkvfl/sscc_vec2/wave.data.test.14p
setenv FOR015 $HOME/QQQF/wkvfl/sscc_vec2/tmitu.data.test.14p
setenv FOR016 $HOME/QQQF/wkvfl/sscc_vec2/mitu.data.test.14p
setenv FOR034 $HOME/QQQF/wkvfl/sscc_vec2/spec.data.test.14p

#  check enevw.data.test.14p every step
setenv FOR002 $HOME/QQQF/wkvfl/sscc_vec2/tempa.data.test.14p
setenv FOR003 $HOME/QQQF/wkvfl/sscc_vec2/molpv.data.test.14p
setenv FOR038 $HOME/QQQF/wkvfl/sscc_vec2/enevw.data.test.14p
setenv FOR039 $HOME/QQQF/wkvfl/sscc_vec2/jyusin.data.test.14p
setenv FOR040 $HOME/QQQF/wkvfl/sscc_vec2/enegg1.data.test.14p
setenv FOR041 $HOME/QQQF/wkvfl/sscc_vec2/enegg2.data.test.14p
setenv FOR042 $HOME/QQQF/wkvfl/sscc_vec2/enegg3.data.test.14p
setenv FOR043 $HOME/QQQF/wkvfl/sscc_vec2/enegg4.data.test.14p
setenv FOR044 $HOME/QQQF/wkvfl/sscc_vec2/enegg5.data.test.14p
setenv FOR045 $HOME/QQQF/wkvfl/sscc_vec2/enegg6.data.test.14p
setenv FOR046 $HOME/QQQF/wkvfl/sscc_vec2/enegg7.data.test.14p
setenv FOR047 $HOME/QQQF/wkvfl/sscc_vec2/disou.data.test.14p
setenv FOR048 $HOME/QQQF/wkvfl/sscc_vec2/avewv.data.test.14p
setenv FOR049 $HOME/QQQF/wkvfl/sscc_vec2/ene14p.data.test.14p
setenv FOR050 $HOME/QQQF/wkvfl/sscc_vec2/force.data.test.14p
setenv FOR052 $HOME/QQQF/wkvfl/sscc_vec2/coef.dddd.test.14p

a.out.test -plk -sz 14 -mbf 5242880 < namlist
```

Fig.4.32 Shell script for parallel execution.

Table.4.1 Cpu time of original version and modified version.

手続き名	計算所要時間　単位：秒	時間発展計算部分に対する割合
時間発展計算部分	0.5100E+03	100.00 %
サブルーチン try [サブルーチン SCHREDI]	0.3821E+03 [0.3637E+03]	74.94 % [71.33 %]
サブルーチン calcu [サブルーチン IONPO4]	0.1278E+03 [0.1050E+03]	25.06 % [20.59 %]

Table.4.2 Execution time on paragon.

npex	使用ノード数	schredi sec (通信所要時間)	ionpo4 sec	時間発展 sec	全体 sec
1	7	0.1808E+05	0.5288E+04	0.2539E+05	0.2559E+05
2	14	0.5439E+04 (0.3397E+02)	0.1960E+04	0.8714E+04	0.8816E+04
4	28	0.2829E+04 (0.6892E+02)	0.9961E+03	0.5144E+04	0.5246E+04
8	56	0.1510E+04 (0.8168E+02)	0.5272E+03	0.3423E+04	0.3527E+04
16	112	0.9014E+03 (0.1247E+03)	0.3073E+03	0.2726E+04	0.2831E+04
32	224	0.6057E+03 (0.1717E+03)	0.2024E+03	0.2358E+04	0.2463E+04

Table.4.3 Speedup ratio

npex	schredi	ionpo4	時間発展	全体
2	3.3	2.7	2.9	2.9
4	6.4	5.3	4.9	4.9
8	12.0	10.0	7.4	7.3
16	20.1	17.2	9.3	9.0
32	29.8	26.1	10.8	10.4

Table.4.4 memory size.

並列化前	41.5
並列化後	11.6

5. Paragon における MONTEV 並列化

VPP 上で実行されていた，光モンテカルロ－分子動力学ハイブリッドコード MONTEV を Paragon へ移植し，複数台のノードを用いて，並列実行できるよう変更した．

5．1 コード概要

MONTEV は分子動力学コードisis[9]を継承し，レーザー光連続照射下での原子系の時間発展を求めるコードである[10] ．MONTEV はモンテカルロ部分(MC部)と分子動力学部分(MD部)に大別される．前回の計算結果を読み込み，前回の計算と時間的に連続したプログラム実行を行うリスタート機能を持っている．Fig.5.1 にリスタート実行用データの入出力の概要を示す．

5．2 VPP500 から Paragon への移植

全てのソースファイルへのインクルードファイル fnx.h のインクルード，全てのファンクション名の変更（例：ファンクション名 fname から real*8 function fname），インクルードファイル para.h における implicit real*8(a-h,o-z) の宣言追加，インクルードファイル para.h のインクルード方法の変更，変数 iwrite の変数名の iwriteorg への変更を行い，Paragon へのプログラム移植を行った．

5．3 プログラム解析

移植後のオリジナルプログラム各部分について，1ノード上での計算所要時間・計算コストを Table.5.1 に示す．担体数6000，原子数7000，時間ステップ1000，MC部呼び出し回数:MD部5回に対し1回の場合に関して clockd() を使用し，時間測定を行った．サブルーチン monte,force の計算コストが高く，この2つで全体の98%を占めている．本並列化では，monte，force の順に並列化を行う．

5．4 並列化

本並列化では，（エネルギー担体数／使用ノード数 nodes ）に関する計算を行うノードを nodes 台並列に動作させるように並列化を行った．使用ノード数は，プログラムの実行開始後システムコールから取得する．また，必要に応じてノード間通信を行う．
ただし，任意の時間ステップでのリスタート実行が可能であるという，本プログラムの動作を損なわないようにした．

（1） インクルードファイルの整備

オリジナルの各ソースプログラム中のコモン配列・変数を削除し，新規作成したインクルードファイル 'inc.h' に記述した．また各ソースプログラムでは 'inc.h' をインクルードするよう変更を行った．Fig.5.2 に inc.h の内容を示す．

（2） メインプログラム main

メインプログラムにおける装置番号50からの入力は，ノード識別番号０ノードがデータを入力した後，他のノードとデータの送受信をすることにした．このために，メインプログラムにおける①②の変更，インクルードファイル inc.h へのコモン領域 /rdatai/, /rdatad/ の追加を行った．

コモン領域 /rdatai/,/rdatad/ が作業配列 nval,dval と結合したため，ノード識別番号０ノードにおいて，整数型の入力は作業配列 nval に，実数型の入力は作業配列 dval に代入される．２つ作業配列の送受信は，入力データの送受信となる．なお，変数名 iwrite は，Paragon ではシステムコールとしての予約文であるため，③の変更を行った．Fig.5.3に変更前の main 概要，Fig.5.4 に変更後の概要を示す．

①作業配列 nval,dval の追加

②equivalence文による，配列nvalとコモン領域/rdatai/，
配列 dval とコモン領域 /rdatad/ のメモリ領域の結合

③変数iwriteの変数名のiwriteorgへの変更

（3） サブルーチン monte

サブルーチン monte の並列化については，goto 文による文番号 200 への繰り返し処理を使用ノード数で均等分割した．このために，文番号 200 への繰り返し処理の終了判定を行う判別式の変更!"を行った． Fig.5.5 に変更前のサブルーチン monte 概要，Fig.5.6 に変更後のサブルーチン monte 概要を示す．

サブルーチン monte の並列化については，goto 文による文番号 200 への繰り返し処理の終了判定を行う判別式の変更①②を行った．Fig.5.5 に変更前のサブルーチン monte 概要，Fig.5.6 に変更後のサブルーチン monte 概要を示す．

①判別式の (i.gt.np) から (i.gt.np/numnodes()) への変更

②判別式が真の場合，変数nabs,ntrs,nsct,配列asiのgdsumによる総和演算の実行

（4） サブルーチン user

サブルーチン user における変更は，entry inlet の入力部分，entry outpi, entry output の出力部分に関するものである．entry inlet , entry outpi, entry output 順に説明する．

サブルーチン user の entry inlet における装置番号45からの入力は，ノード識別番号０ノードがデータを入力した後，他のノードとデータの送受信をすることにした．このために，サブルーチン user における①②の変更を行った．

①作業配列 nval,dval,darry,randarry の宣言

②equivalence文による作業配列darryとコモン領域/xyz/のメモリ領域の結合

コモン領域 /xyz/ が作業配列 darry と結合したので，ノード識別番号０ノードにおいて，装置番号45からの実数型の入力は作業配列 darry に代入され，整数型の入力変数は作業配列 nval に代入される．２つ作業配列の送受信は入力データの送受信となる．なお，コモン領域 /xyz/ はインクルードファイル inc.h に記述されている．

ここで，乱数発生に関して行ったサブルーチン user の変更を説明する．初回時間ステップからの計算は，判別式 (nval(4).eq.0) が真の場合である．この場合，入力データに記録されている乱数の種は１つである．各ノードに異なる乱数の種を与えるため，③④の変更を行った．リスタート実行であるのは，判別式 (nval(4).eq.0) が偽の場合である．この場合，前回計算出力には乱数の種が nodes 個記録されている．各ノードに前回実行時と連続した乱数を与えるため，⑤⑥の変更を行った．

③識別番号０のノードが配列 randarry(1) に種を１つ入力し，各ノードに送信する

④受信した種に自己のノード識別番号を足し，各ノードごとに異なる乱数の種とする

例：rand= 5249347.d0+dble(mynode())

⑤識別番号０のノードが配列要素 (randarry(i),i=1,nodes) に

86nodes 個の種を入力し，各ノードに送信する

⑥各ノードは配列 ranndarry から使用する種 rand を抽出する．

例：rand=randarry(mynode()+1)

サブルーチン user の entry outpi,entry output における出力データは，乱数の種以外は各ノード上に同じものが配置されている．乱数の種に関しては，各ノード上の乱数の種を識別番号の０ノードに送信し，この０ノードにおいて各種を配列 randarry に代入し，これ以外の出力データをあわせて識別番号０のノードから出力する．

Fig.5.7 に変更前のサブルーチン user entry inlet 概要，Fig.5.8 に変更後の entry inlet 概要を示す．Fig.5.8に変更後の entry outpi 概要，Fig.5.8 に変更後の entry outpi 概要を示す．

（５） サブルーチン tabel

本サブルーチンにおける変更は，次の①②である．Fig.5.9 に変更前の table を，Fig.5.10 に変更後の table を示す．

①do 10 のループカウンタ変数 icnt を配列 ndx のインデックスをとした

②do 10 を，各ノードに循環的に割り当てた

（６） サブルーチン force

サブルーチン force の並列化については，do 10 のループカウンタ変数 icnt を配列 ndx のインデックスをとし，do 10 を各ノードに循環的に割り当てた．この場合，配列 pot,biral,fx0,fy0,fz0,near の総和計算が必要である．

総和計算には gdsum を使用した．gdsum の使用に関して，呼び出し回数を最小にするため，また１回に通信するデータ量を大きく（500KB以上）して通信効率を向上させるため，本サブルーチンにおいて次の①～③の変更を行った．この変更により gdsum の呼び出しは，作業配列 fxyz0,pot_biral,near に関する３回となった．Fig.5.11 に変更前の force を，Fig.5.12 に変更後の force を示す．

①作業配列 fxyz0,pot_biral,wfxyz0,wpot_biral,nwnear の宣言

②equivalence 文による，作業配列fxyz0とコモン領域/xyz/のメモリ領域を結合

③同様に作業配列 pot_biral に，２つの配列 pot,biral の格納

5．5　並列化の効果

Table.5.2 に示すプログラム動作パラメータでの実行時における並列化効果の評価を行う．並列実行時の使用ノード数は nodes=10,50,100,200,300,500 とした.Table.5.3 (1)にプログラム各部分の計算所用時間，Table.5.3(2)に速度向上=nodesノードの計算所要時間／nodesノードの計算所要時間を示す．また．Table.5.4 に必要メモリ量についてまとめる．本並列化によって，作業配列の使用により必要メモリ量の縮小は達成されなかった．また使用ノード数が50程度までは，ほぼ台数分の効果が得られた．

5．6　実行方法

MD_paragon.tar.Z を解凍後，ロードモジュールの作成，ＮＱＳへの投入の順に行う．MD_paragon.tar.Z には，paragon 上でのプログラム実行に必要な，ファイル一式(ソースプログラム・インクルードファイル・Makefile 記述例・nqs 投入用シェルスクリプト記述例・入力データファイル）を，アーカイブ形式にまとめ圧縮して格納してある．Fig.5.13 に MD_paragon.tar.Z 解凍時のファイルリストを示す．

（1）　アーカイブ形式ファイルについて

・圧縮ファイルの ftp 等による転送は，バイナリモードで行う．

転送例　**>bin** [RET]
　　>put *MD_paragon.tar.Z* [RET]

・ファイルの解凍には，uncompress，tar コマンドを使用する．

解凍例　1%**uncompress** *MD_paragon.tar.Z* [RET]
　　2%**tar -xvf** *MD_paragon.tar* [RET]

（2） ロードモジュールの作成（ソースプログラムのコンパイル・リンク）

ロードモジュールの作成には，make コマンドを実行する．Fig.5.14 に Makefile の記述例を示す．

・ロードモジュールの作成
例 **%make** [RET]

（3） プログラムの実行

プログラムの実行は，ＮＱＳ投入用シェルスクリプトを qsub コマンドによってＮＱＳへ投入して行う．Fig.5.15 にＮＱＳ投入用シェルスクリプトの記述例を示す．Fig.5.15 のようなシェルスクリプトを，ジョブクラス，ディレクトリ名，ファイル名等を実行時の環境に適したように変更し使用する．なお，使用ノード数は Fig.5.15 の例での ***nodes*** のように実行時オプション -sz で指定する．ただし，エネルギー担体数の素因数でなければならない．また，リスタート実行時は前回実行時と同じノード数を指定しなければならない．

・ＮＱＳへの投入
例 **%qsub** *montevP010.sh* [RET]

・テスト計算に用いたプログラム動作環境（*NODES*=使用ノード数）

計算機 paragon(OS UNIX)	変更前	変更後
言語	fortran77	fortran77
並列化ライブラリ	nx	nx
コンパイラ	if77	if77
最適化オプション	-nx -O2	-nx -O2
実行時オプション	-plk -sz 1	-plk -sz *NODES*

5．7 まとめ

本並列化において，通信用システムコールのオーバーヘッドを極力小さくするため，また１回における通信データ量を大きくして高速通信を行うため，作業配列とコモン領域のメモリ領域の結合をequivalence 文を用いて行った．

並列化後プログラムの計算結果については，完全な一致は得られなかった．これは，演算順序の変更・乱数列の変更などによるものと考えられるが，特に問題ないと思われる．

ノード数の増加に従って速度向上が下がっており，性能は50ノード程度でほぼ飽和した．これは，ノード数の増加に比例して，負荷バランスの低下・通信負荷の増加が起こっているためと考えられる．しかし，追跡するエネルギー担体数・原子数の増加に対しては，負荷バランスのばらつきの相殺が起こり，通信負荷も相対的に小さくなる．つまり，追跡するエネルギー担体数・原子数の増加に対しては，十分な並列化の効果があり，したがって，本並列化による並列化によるプログラムの高速化は，十分に達成できたと考えられる．

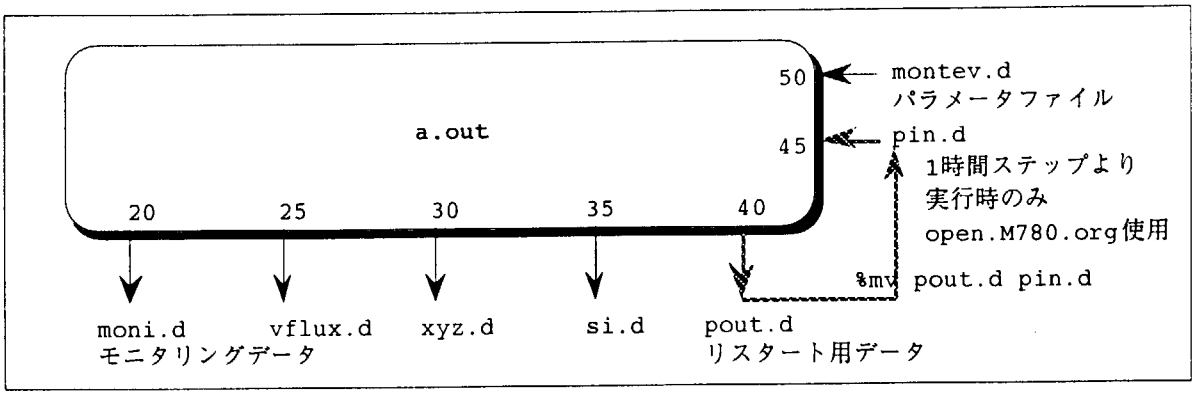

Fig.5.1 Input and output data for restart execution.

```
      common
c.para<<
c.main.intger...<<
     &/all/n,ahalf
     &/step/kstep,nstep
     &/rdatai/job,mstepa,mstepb,mcor,iswp,mtime,mx,my,mz,mcc
c.main.intger...>>
c.main.double...<<
     &/lasr/vi,si,spp,spv,ref
     &/prtcl/sig11,r11,dd,pi,pv,pm,pot(mol)
     &/scle/g,dzdt,zfac
     &/tabl/ipair(mol/2*mpair),ndx(0:mol-1)
     &     ,nup,ncnt,rcs,rcs2,rsp2
     &/rdatad/gs,ge,dr,rsp,rpct,rm1,sig,d,rho,g0
     &     ,alf,r0,dt,rc,v0,um
c.main.double...>>
c.user.double...<<
     &/xyz/x0(mol),y0(mol),z0(mol),xc(mol),yc(mol),zc(mol)
     &    ,vx(mol),vy(mol),vz(mol),dxc(mol),dyc(mol),dzc(mol)
     &    ,fx0(mol),fy0(mol),fz0(mol),fx1(mol),fy1(mol),fz1(mol)
     &    ,fx2(mol),fy2(mol),fz2(mol),fx3(mol),fy3(mol),fz3(mol)
     &/xxx/x(mol),xp(mol),dxp(mol),ox(mol),sidex,sid2x
     &/yyy/y(mol),yp(mol),dyp(mol),oy(mol),sidey,sid2y
     &/zzz/z(mol),zp(mol),dzp(mol),oz(mol),sidez,sid2z
c.user.double...>>
     &/timecheck/tmontetrn,tforcetrn,ttbltrn
c.para>>
     &/bolt/btemp(mol),tb(50),nk(50)
     &/comp/cx,cy,cz
     &/momp/px,py,pz,ek
     &/mont/asi(mol),rand
     &/sursl/asf,asfb,area,nv,nvb,no
     &/surlg/asfg,asfgb,nvg,nvgb,near(mol)
     &/unit/unitl,unitt
     &/vekz/zek(31),tempz(31),rmstz(31),ezmax,ezmin,numz(31)
     &/vflx/npu,npd,eku,ekd
```

Fig.5.2 New include file inc.h

```
include'para.h'
common
&/all/.....................
&/prtcl/.....................
&/step/.....................
&/unit/.....................
&/xxx/.....................
&/yyy/.....................
&/zzz/.....................
&/varx/.....................
&/vary/.....................
&/varz/.....................
&/frcx/.....................
&/frcy/.....................
&/frcz/.....................
common
&/scle/.....................
&/tabl/.....................
&/momp/.....................
&/comp/.....................
&/sursl/.....................
&/surlg/.....................
&/lasr/.....................
&/mont/.....................
  read(50,*)  job,mstepa,mstepb,mcor
  read(50,*)  iswp,mtime,iswr,iswv,iswo
  read(50,*)  iavp,iavm,iavr,iavv
  read(50,*)  gs,ge,dr,rsp,rpct
  read(50,*)  rm1,sig,d,rho,g0,alf,r0
  read(50,*)  mx,my,mz,mcc,dt,rc,x1,rs,rm
  read(50,*)  v0,um

  call set
  call inlet
  call outpi
  call table
  iwrite=0
  do 100 istep=1,mstepb
     if(job.eq.1.and.mod(istep,mtime).eq.1)
&        call monte(absr,trns,sctt,istep,ncal)
     call force(ew,prs,nvap)
     if(job.eq.1.and.mod(istep,iswo).eq.1) iwrite=iwrite+1
     call stepve(temp,tempb,num,k,zmin,zmax
&               ,px,py,pz,amelf,istep,iswo,iwrite)
     call check
     call table
     call surfsl(amelf)

     if(mod(istep,iswr).eq.1) then
        call vflux(25,istep,prs,zmax,temp,vprs,nvap,amelf)
     end if
     if(mod(istep,iswp).eq.0) then
        write(6,*)' '
                        :
     end if
                       :
  stop
  end
```

Fig.5.3 Original main program.

```
       include'para.h'
       include'inc.h'
       include'fnx.h'

      dimension nval(17),dval(19)
      equivalence (nval(1),job)
      equivalence (dval(1),gs)

      if( mynode().eq.0) then
      read(50,*)  nval(1),nval(2),nval(3),nval(4)
      read(50,*)  nval(5),nval(6),nval(11),nval(12),nval(13)
      read(50,*)  nval(14),nval(15),nval(16),nval(17)
      read(50,*)  dval(1),dval(2),dval(3),dval(4),dval(5)
      read(50,*)  dval(6),dval(7),dval(8),dval(9)
     +           ,dval(10),dval(11),dval(12)
      read(50,*)  nval(7),nval(8),nval(9),nval(10)
     +           ,dval(13),dval(14),dval(17),dval(18),dval(19)
      read(50,*)  dval(15),dval(16)

      call csend(100,nval,17*4,-1,0)
      call csend(200,dval,19*8,-1,0)
      else
      call crecv(100,nval,17*4)
      call crecv(200,dval,19*8)
      end if

      iswr  = nval(11)
      iswv  = nval(12)
      iswo  = nval(13)
      iavp  = nval(14)
      iavm  = nval(15)
      iavr  = nval(16)
      iavv  = nval(17)
      xl    = dval(17)
      rs    = dval(18)
      rm    = dval(19)
c
      call set
      call inlet
      call outpi
c --- make tblae of praticle pair ---
      call table
                :
```

Fig.5.4 Modified main program.(1/2)

```
                  :
                  :
      iwriteorg=0
      do 100 istep=1,mstepb
        if(job.eq.1.and.mod(istep,mtime).eq.1) then
        call monte(absr,trns,sctt,istep)
        end if
        call force(ew,prs,nvap)
        if(job.eq.1.and.mod(istep,iswo).eq.1) iwriteorg=iwriteorg+1
        call stepve(temp,tempb,num,k,zmin,zmax
     &             ,amelf,istep,iswo,iwriteorg)
        call surfsl(amelf)

c --- monitering ---
         if(mod(istep,iswr).eq.1) then
           call vflux(25,istep,prs,zmax,temp,vprs,nvap,amelf)
         end if
         if(mod(istep,iswp).eq.0) then
           if(mynode().eq.0.) then
           write(6,*)' '
           write(6,*)'istep=',istep
                     :
           write(6,121)ew*bcnst*g0/1.6022d-19
           write(6,122)absr*100.0d0,trns*100.0d0,sctt*100.0d0
           end if
         end if
         nall=nall+ncal
  100 continue
                  :
      stop
      end
```

Fig.5.4 Modified main program.(2/2)

```
      subroutine monte(absr,trns,sctt,istep,ncal)
      include'para.h'
      include'fnx.h'
      common
     &/all/.....
     &/prtcl/.....
     &/step/.....
     &/unit/.....
      common
     &/xxx/.....
     &/yyy/.....
     &/zzz/.....
     &/sursl/.....
     &/surlg/.....
     &/lasr/.....
     &/mont/.....
            :
      si=0.0d0
      do 100 i=1,n
        asi(i)=0.0d0
  100 continue
c --- initial position & velocity ---
            :
      nabs=0
      ntrs=0
      nsct=0
      ncal=0
      i=0
  200 i=i+1
      if(i.gt.np) then
            :
          return
      end if

  210 continue
c --- find collision pair ---
            :
      if(pnz(i).lt.zmin)then
         ntrs=ntrs+1
         goto 200
      end if
      if(pnz(i).gt.zmax) then
         nsct=nsct+1
         goto 200
      end if
            :
c --- judgement of scattering or absorption ---
      re=ran(rand)
      if(re.lt.(1.0d0-ref)) then
        asi(jpnt)=asi(jpnt)+sia
        if(z(jpnt).le.asfg+sigll) si=si+sia
        nabs=nabs+1
        goto 200
      else
c --- isotropic scattering ---
            :
      end if

      end
```

Fig.5.5 Original subroutine monte.

```
      subroutine monte(absr,trns,sctt,istep)
      include'para.h'
      include'inc.h'
      include'fnx.h'
            :
      dimension wasi(mol),nptcl(3),nptclw(3)
      equivalence(nptcl(1),ntrs),(nptcl(2),nsct),(nptcl(3),nabs)
            :
      si=0.0d0
      do 100 i=1,n
        asi(i)=0.0d0
  100 continue
c --- initial position & velocity ---
            :
      nabs=0
      ntrs=0
      nsct=0
      ncal=0
      i=0
  200 i=i+1
      if(i.gt.np/numnodes()) then
      call gisum(nptcl,3,nptclw)
      call gdsum(asi,mol,wasi)
      call gdsum(si,1,wsi)
            :
      return
      end if

c --- find collision pair ---
            :
      if(pnz(i).lt.zmin)then
         ntrs=ntrs+1
         goto 200
      end if
      if(pnz(i).gt.zmax) then
         nsct=nsct+1
         goto 200
      end if
            :
c --- judgement of scattering or absorption ---
      re=ran(rand)
      if(re.lt.(1.0d0-ref)) then
        asi(jpnt)=asi(jpnt)+sia
        if(z(jpnt).le.asfg+sig11) si=si+sia
        nabs=nabs+1
        goto 200
      else
c --- isotropic scattering ---
        theta=2.0d0*pi*ran(rand)
        eta=dacos(1.0d0-2.0d0*ran(rand))
            :
      end if

      end
```

Fig.5.6 Modified subroutine monte.

```
      subroutine user(istep)
      include'para.h'
              :
      entry inlet
              :
      do 100 i=1,n
         ox(i)=0.0d0
         oy(i)=0.0d0
         oz(i)=0.0d0
         asi(i)=0.0d0
  100 continue
      dzdt=0.0d0
      ncnt=0
      nup=1
      rewind(45)
      read(45,*) nstep
     +        ,(x0(i),y0(i),z0(i),i=1,n)
     +        ,(xc(i),yc(i),zc(i),i=1,n)
     +        ,(vx(i),vy(i),vz(i),i=1,n)
     +        ,(dxc(i),dyc(i),dzc(i),i=1,n)
     +        ,(fx0(i),fy0(i),fz0(i),i=1,n)
     +        ,(fx1(i),fy1(i),fz1(i),i=1,n)
     +        ,(fx2(i),fy2(i),fz2(i),i=1,n)
     +        ,(fx3(i),fy3(i),fz3(i),i=1,n)
     +         ,asfb,nvb
     +         ,nvgb,kstep
     +         ,asfgb
     +         ,si,rand
      asf=asfb
      asfg=asfgb
      if(kstep.eq.0) rand=5249347.d0
                    :
      return
c
      entry outpi
         write(20,111)
         write(20,333) asys(job),n,gs,ge
     +         ,g0,rho
                    :
         write(20,*)'v0=',v0
                    :
      return
c
      entry output(istep)
                    :
      write(20,444) ncnt,mstepb
      write(40,*) nstep
     +        ,(x0(i),y0(i),z0(i),i=1,n)
                    :
     +        ,si,rand

      call boltz(20)
      do 42 i=1,n
                    :
   42 continue

      call ekz(20,istep)
      return
      end
```

Fig.5.7 Original subroutine user.

```
      subroutine user(istep)
      include'para.h'
      include'inc.h'
      include'fnx.h'
      dimension nval(4),darry(mol*24),dval(3),randarry(800)
      equivalence (darry(0*mol+1),x0(1))
           :
      entry inlet
           :
      nseed=numnodes()
      if(mynode() .eq. 0) then
      read(45,*) nval(1)
     +  ,(darry(0*mol+i),darry(1*mol+i),darry(2*mol+i),i=1,n)
     +  ,(darry(3*mol+i),darry(4*mol+i),darry(5*mol+i),i=1,n)
     +  ,(darry(6*mol+i),darry(7*mol+i),darry(8*mol+i),i=1,n)
     +  ,(darry(9*mol+i),darry(10*mol+i),darry(11*mol+i),i=1,n)
     +  ,(darry(12*mol+i),darry(13*mol+i),darry(14*mol+i),i=1,n)
     +  ,(darry(15*mol+i),darry(16*mol+i),darry(17*mol+i),i=1,n)
     +  ,(darry(18*mol+i),darry(19*mol+i),darry(20*mol+i),i=1,n)
     +  ,(darry(21*mol+i),darry(22*mol+i),darry(23*mol+i),i=1,n)
     +  ,dval(1),nval(2),nval(3),nval(4),dval(2),dval(3)

         if(nval(4).eq.0) nseed=1
         read(45,*)(randarry(i),i=1,nseed)
      end if

      if(mynode() .eq. 0) then
         call csend(350,nseed,1*4,-1,0)
      else
         call crecv(350,nseed,1*4)
      end if

      if(mynode() .eq. 0) then
      call csend(100,nval,4*4,-1,0)
      call csend(200,darry,mol*8*24,-1,0)
      call csend(300,dval,3*8,-1,0)
      call csend(400,randarry,nseed*8,-1,0)
      else
      call crecv(100,nval,4*4)
      call crecv(200,darry,mol*8*24)
      call crecv(300,dval,3*8)
      call crecv(400,randarry,nseed*8)
      end if

      nstep=nval(1)
      nvb  =nval(2)
      nvgb =nval(3)
      kstep=nval(4)
      asfb =dval(1)
      asfgb=dval(2)
      si   =dval(3)
      if(kstep .eq. 0) then
         rand =randarry(1)
      else
         rand =randarry(mynode()+1)
      end if
      if(kstep.eq.0)
           :
      return          rand=5249347.d0+dble(mynode())
```

Fig.5.8 Modified subroutine user.(1/2)

```
      subroutine user(istep)
           :
           :
      entry outpi
         if( mynode().eq.0 ) then
         write(20,111)
         write(20,333) asys(job),n,gs,ge
     +         ,g0,rho
           :
         write(20,*)'v0=',v0
         end if
           :
      return
c
      entry output(istep)
c --- output for results ---
c.para<<
      if( mynode().ne.0 ) then
          msgid=mynode()
          call csend(msgid,rand,1*8,0,0)
      else
          randarry(1)=rand
          do 700 inode=1,numnodes()-1
          call crecv(inode,rand,1*8)
          randarry(inode+1)=rand
 700  continue
      end if
c.para>>
      if( mynode().eq.0 ) then
      write(20,444) ncnt,mstepb
      write(40,*) nstep
     +        ,(x0(i),y0(i),z0(i),i=1,n)
           :
           :
     +        ,si
     +        ,(randarry(inode),inode=1,numnodes())
c.para>>
      end if

      call boltz(20)

      if( mynode().eq.0 ) then
      do 42 i=1,n
           :
   42 continue
      end if
      call ekz(20,istep)
      return
      end
```

Fig.5.8 Modified subroutine user.(2/2)

```
      subroutine table
      include'para.h'
      include'fnx.h'
     common
    &/all/.................
    &/prtcl/...............
    &/step/................
    &/unit/................
    &/tabl/................
     common
    &/xxx/.................
    &/yyy/.................
    &/zzz/.................

     ndx(0)=0
     k=0
     do 10 i=1,n-1
        do 20 j=i+1,n
              :
              k=k+1
              ipair(k)=j
           end if
 20     continue
        ndx(i)=k
 10  continue

     return
     end
```

Fig.5.9 Original subroutine table.

```
      subroutine table
      include'para.h'
      include'inc.h'
      include'fnx.h'
c
      ndx(0)=0
      k=0
      icunt=0

      do 10 i=mynode()+1,n-1,numnodes()
      icunt=icunt+1
         do 20 j=i+1,n
               :
               k=k+1
               ipair(k)=j

 20     continue
        ndx(icunt)=k
 10  continue
     return
     end
```

Fig.5.10 Modified subroutine table.

```
      subroutine force(ew,prs,nvap)
      include'para.h'
      include'fnx.h'
     common
    &/all/.....................
    &/prtcl/.....................
    &/step/.....................
    &/unit/.....................
    &/tabl/.....................
     common
    &/xxx/.....................
    &/yyy/.....................
    &/zzz/.....................
    &/varx/.....................
    &/vary/.....................
    &/varz/.....................
    &/frcx/.....................
    &/frcy/.....................
    &/frcz/.....................
     common
    &/scle/.....................
    &/sursl/.....................
    &/surlg/.....................
    &/lasr/.....................
      dimension ipnt(mol),biral(mol),ivap(mol)

      do 5 i=1,n
         fx0(i)=0.0d0
         fy0(i)=0.0d0
         fz0(i)=0.0d0
         pot(i)=0.0d0
         biral(i)=0.0d0
         near(i)=0
 5 continue

      do 10 i=1,n-1
          do 20 jdx=ndx(i-1)+1,ndx(i)
            j=ipair(jdx)

              pot(i)=pot(i)+ppp
              biral(i)=biral(i)+bbb
              fx0(i)=fx0(i)+dum*xij
              fy0(i)=fy0(i)+dum*yij
              fz0(i)=fz0(i)+dum*zij
              near(j)=near(j)+1
              near(i)=near(i)+1
20      continue
10 continue

      return
      end
```

Fig.5.11 Original subroutine force.

```
      subroutine force(ew,prs,nvap)
      include'para.h'
      include'inc.h'
      include'fnx.h'
      dimension ipnt(mol),biral(mol),ivap(mol)
      dimension fxyz0(3*mol),pot_biral(2*mol)
      dimension wfxyz0(3*mol),wpot_biral(2*mol),nwnear(mol)
      equivalence (fxyz0(1),fx0(1))
     & ,(pot_biral(1),pot(1)),(pot_biral(mol+1),biral(1))
c
      do 5 i=1,n
         fx0(i)=0.0d0
         fy0(i)=0.0d0
         fz0(i)=0.0d0
         pot(i)=0.0d0
         biral(i)=0.0d0
         near(i)=0
    5 continue
c
      icunt=0
      do 10 i=mynode()+1,n-1,numnodes()
      icunt=icunt+1
                    :
          do 20 jdx=ndx(icunt-1)+1,ndx(icunt)
            j=ipair(jdx)
                    :
            pot(i)=pot(i)+ppp
            biral(i)=biral(i)+bbb
            fx0(i)=fx0(i)+dum*xij
            fy0(i)=fy0(i)+dum*yij
            fz0(i)=fz0(i)+dum*zij

            near(j)=near(j)+1
            near(i)=near(i)+1
   20     continue
   10 continue
      call gdsum(fxyz0,3*mol,wfxyz0)
      call gdsum(pot_biral,2*mol,wpot_biral)
      call gisum(near,mol,nwnear)

      return
      end
```

Fig.5.12 Modified subroutine force.

```
MD_paragon/
    ├── SH/
    ├── MDdata/
    ├── data/
    └── src_pgn_p/
```

```
src_pgn_p/
Makefile  コンパイル・リンク用 Makefile 用記述例
inc.h     インクルードファイル（配列宣言など）
para.h    インクルードファイル（パラメータ記述）

*.f       以下並列実行用ソースファイル一式
boltz.f          boltz_p.f        cbrt.f           check.f
com.f            ekz.f            force_p.f        main_p.f
mom.f            monte_p.f        ran.f            ranu.f
stepve.f         surf.f           table.f          table_p.f
user_p.f         vflux.f          force.f
```

```
SH/     nqs 投入用シェルスクリプト記述例
montevP010.sh    montevP100.sh    montevP300.sh
montevP050.sh    montevP200.sh    montevP500.sh
```

```
MDdat/open/open.M780 unit45 よりの入力データ．初回実行時のみ使用
data/montev.d        unit50 よりの入力データ．動作制御用パラメータ
```

Fig.5.13 File lists of the MD_paragon.tar.Z.

```
#
# Center Sample Makefile
#               Write Day : Mon Jan  8 13:10:14 1996
#

F77     = if77
MAKE    = Makefile
OPT     = -nx -O2
INCLUDES =
TERGET  = a.out

OBJS = \
      boltz_p.o  cbrt.o  check.o  com.o  ekz.o\
      force_p.o  main_p.o  mom.o  monte_p.o  ran.o\
      ranu.o  stepve.o  surf.o  table_p.o  user_p.o\
      vflux.o

.f.o :
 $(F77) $(OPT) $(INCLUDES) -c $<

all :$(OBJS)
 $(F77) $(OPT) -o $(TERGET) $(OBJS) $(LIBES)

clean :
 rm -f $(OBJS)
```

Fig.5.14 Makefile for making load-module.

```
#!/bin/csh -f
#---------------- SET NQS OPTIONS ----------------
#@$-q  spp128-2 # set queue class
#@$-eo
#---------------- SET NQS OPTIONS ----------------

set MDdata=$HOME/MD/MDdata/montev

#--- read
setenv FOR045 $MDdata/pin.d
#setenv FOR050 $HOME/MD/data/montev.d.moni
setenv FOR050 $HOME/MD/data/montev.d

#--- write
setenv FOR020 $HOME/MD/data/moni.d.500
setenv FOR025 $MDdata/vflux.d.500
setenv FOR030 $MDdata/xyz.d.500
setenv FOR035 $MDdata/si.d.500
setenv FOR040 $MDdata/pout.d.500

#cd $MDdata
#   cp pout.d pin.d

cd $HOME/MD/
a.out.p -plk -sz nodes
```

Fig.5.15 Shell script for parallel execution.

Table.5.1 Cpu time of original version.

手続き名	計算所要時間	計算コスト
monte	0.1322E+05	0.4373E+00
force	0.1502E+04	0.4629E+00
stepve	0.1180E+03	0.3955E-02
check	0.1843E+02	0.6177E-03
Table	0.1748E+04	0.5858E-01
surfsl	0.1767E+01	0.5924E-04
vflux	0.1765E+01	0.5915E-04
total	0.2984E+05	0.1000E+01

Table.5.2 parameter for program action.

変数名	内容	値	インクルードファイル名
np	エネルギー担体数	6000	para.h
mol	原子数	7000	para.h
istep	時間ステップ	10000	monte.d
iswr	モニタリング	100ステップに1回	monte.d
mtime	monte 呼び出し回数	MD部5回に1回	monte.d

Table.5.3 Execution time and Speedup ratio on Paragon.

使用ノード数	(1)MC 部 sec	(1)MD 部 sec	(1)全体 sec	(2)速度向上
1	0.1322E+05	0.1419E+04	0.2984E+05	1
10	0.1473E+05	0.1502E+04 (0.3526E+02)	0.3244E+04	9.20
50	0.3383E+03	0.3383E+03 (0.4579E+02)	0.8675E+03	34.40
100	0.1999E+03	0.2012E+03 (0.3785E+02)	0.5590E+03	53.38
200	0.1274E+03	0.1514E+03 (0.6138E+02)	0.4307E+03	69.28
300	0.1032E+03	0.1507E+03 (0.8705E+02)	0.4049E+03	73.70
500	0.8242E+02	0.1689E+03 (0.1259E+03)	0.4087E+03	73.01

Table.5.4 memory size.

並列化前	16.4 MB
並列化後	16.8 MB

6. 終わりに

今回扱ったコードのうち並列化を施したコードについては，従来よりも一層大きな系や長時間のシミュレーションが可能となった．これらのコードは，関西研究所で頻繁に使用されているものであり，かつ光量子シミュレーションコードの典型的なタイプであるため，本報告書の並列化，ベクトル化等の成果が，他のコードの改良によい影響を与えることを期待したい．

謝　　辞

本業務を行う上での，vpp500，vpp300，Paragon のハードウエア・ソフトウエアに関する相談について，計算科学推進センター藤井　実氏，（株）富士通　渡辺　秀雄氏，鈴木氏，（株）インテル　池井氏　清水氏，プログラム相談室，並びに関西研究所計算機管理課の方々にご協力いただきました．ここに感謝の意を表します．

最後に，本報告書を書く機会を与えて下さいました日本原子力研究所（現在　宇宙開発事業団地球シミュレータ研究開発センター）谷　啓二氏に感謝いたします．

参 考 文 献

[1] 芝原正彦，光熱変換機構の量子分子動力学的研究，博士論文，(1996)東京大学．
[2]「UXP/M VPP アナライザ 使用手引書 V10用」，富士通（株），1994年1月．
[3]「原研 VPP500/42システム利用手引 第２版」
日本原子力研究所　計算科学推進センター情報システム管理室，1995年，7月．
[4]「UXP/M VPP FORTRAN77 EX/vpp 使用手引書 V12用」
，富士通（株），1994年1月．
[5]「UXP/M VPP FORTRAN77 EX/vpp 使用手引書 V12用」，
富士通（株），1994年1月．
[6]「関西研究所 並列計算機(Paragon)利用手引 第１版」
日本原子力研究所　計算科学推進センター情報システム管理室，1996年，5月
[7] PARAGON^{TM} Fortran System Calls Refernce Manual.
[8] PARAGON^{TM} Fortran Compiler User's Guide.
[9] 藤井実，私信．
[10] 功刀資彰他，機論B，61，1826（1995）．

付録1　自動並列化機能（マルチプロセッサ最適化機能）の適用

本コード QQQF に関しては，自動並列化機能の適用によって，サブルーチン scredi do 140 が約2倍高速化された．この結果プログラム全体で約1.7倍の高速化が実現された．よって，本作業ではこの機能を利用することにした．

-Mconcur の利用によるサブルーチン scredi do 140 高速化の理由は，この手続きでのキャッシュヒット率の向上であると考えられる．これは，do 140 のループ分割がなされていないこと，使用キャッシュサイズを半分に指定した場合の実行時間が，そうでない場合より遅いことから推測される．

以下に左から7並列実行用プログラムにおいて，-Mconcur 指定時，-Mconcur 指定時使用・キャッシュサイズデフォルトの半分，-Mconcur 指定なし，での各サブルーチンの1通過の所要時間の測定結果，テスト実行時におけるプログラム動作環境を示す．

（1）各サブルーチン1通過の所要時間　単位：秒

サブルーチン名	2CPU (1) キャッシュサイズ8KB	2CPU(2) キャッシュサイズ4KB	1CPU(3) キャッシュサイズ8KB
AFTER1	0.1047E-02	0.6584E-03	0.9566E-03
BEFOR1	0.1020E-02	0.2058E-02	0.2060E-02
CALCU	0.2054E-02	0.2370E-02	0.3537E-02
DATA	0.1211E-02	0.8600E-05	0.9345E-03
ELECDEN	0.1442E-02	0.1447E-02	0.1532E-02
ENEAF	0.1704E-02	0.1926E-02	0.2100E-02
ENEGBE	0.1659E-01	0.2230E-01	0.2282E-01
FFT	0.1445E-01	0.3349E-02	0.7584E-02
INTERPO	0.2763E+00	0.3089E+00	0.4767E+00
INVFFT	0.3127E-01	0.4328E-01	0.4341E-01
IONPO4	0.1048E+02	0.1049E+02	0.1058E+02
IONPRE	0.2337E-02	0.2461E-02	0.4542E-02
NLCONSP	0.3824E-03	0.4473E-03	0.5936E-03
RFFT	0.1373E-01	0.1405E-01	0.2757E-01
SCHREDI	0.1808E+01	0.2413E+01	0.3672E+01

（2）テスト実行時におけるプログラム動作環境

計算機 paragon	(1)	(2)	(3)
言語	fortran77	fortran77	fortran77
並列化ライブラリ	nx	nx	nx
コンパイラ	if77	if77	if77
最適化オプション	-O2	-O2 -Mconcur -Mvect cachesize=8192	-O2 -Mconcur
実行時オプション	-plk -sz 7	-plk -sz 7	-plk -sz 7

付録2 サブルーチン testプログラムリスト


```
c*****************************************************
c 1996.08.01     kaori@love.tokai.jaeri.go.jp
      subroutine test
     &(dsenddata,dbuffer,ndim,npex,npey,id)
c*****************************************************
      implicit real*8 (a-h,o-z)
      include 'fnx.h'
      dimension dsenddata(ndim),dbuffer(ndim)
      common /nowstep/ nwstep
      common/testcomm/
     &        iam(2)
     &       ,iswch(2),nfmly(2,800)
     &       ,ndsel(2,800),ndsel0(2,2,800)

      call gsync()
      if(nwstep.eq.1) then
      call traninit(npex,npey)
      end if

      len = ndim*8
      k=0
 100  continue
      k=k+1
c ---------- buffer clear
      do 20 i=1,ndim
         dbuffer(i)=0.0d0
  20  continue

c ---------- set break condition
      if(iswch(id).eq.0 .and. nfmly(id,k).eq.1) goto 999
      if(iswch(id).eq.1 .and. nfmly(id,k).eq.0) goto 999

c ---------- spacial node action [iclstr(id)=7&iam(id)=0]
      if( iswch(id).eq.1 .and. iam(id).eq.0 ) then
      if( k.eq.1) goto 188

c      ------------------ iii=1:dummy action (iii=0:standard action)
      mid             = mynode()
      do 250 iii=1,2
      ndsl            = ndsel0(iii,id,k)
      call csend(mid,dsenddata,len,ndsl,0)
 250  continue
      mid1=ndsel0(1,id,k)
      mid2=ndsel0(2,id,k)
      call crecv(mid1,dbuffer,len)
      call crecv(mid2,dbuffer,len)
 188  continue

c ---------- standard node action
      else
      ndsl        = ndsel(id,k)
      mid         = mynode()
c     write(*,*) mynode(),ndsl
      call csend(mid,dsenddata,len,ndsl,0)
      call crecv(ndsl,dbuffer,len)
      end if
```



```
c ---------- summention
      do 30 i=1,ndim
      dsenddata(i)=dsenddata(i)+dbuffer(i)
  30  continue

      call gsync()
      goto 100
 999  continue

      call gsync()

      return
      end

c*****************************************************
c 1996.08.01     kaori@love.tokai.jaeri.go.jp
      subroutine traninit(npex,npey)
c*****************************************************
      implicit real*8 (a-h,o-z)
      include 'fnx.h'
      dimension iclstr(2),nhead(2),nn(2)
      common/testcomm/
     &        iam(2)
     &       ,iswch(2),nfmly(2,800)
     &       ,ndsel(2,800),ndsel0(2,2,800)

c ---------- topology
      nodes    = numnodes()
      nn(1)    = 1
      nn(2)    = npex
      iclstr(1)= npex
      iclstr(2)= npey
      iam(1)   = mod(mynode(),npex)
      iam(2)   = int(mynode()/npex)
      nhead(1) = iam(2)*npex
      nhead(2) = iam(1)
      iswch(1)= mod(npex,2)
      iswch(2)= mod(npey,2)


      id=0
 2000 continue
      id=id+1

c ---------- id change & topology reset
      iam(1)   = mod(mynode(),npex)
      iam(2)   = int(mynode()/npex)
      if(iswch(id).eq.0 .and. (iam(id)+1).eq.iclstr(id)) iam(id)=-1

      k=0
 1000 continue
      k=k+1
      kk=2**(k-1)
      nfmly(id,k)=iclstr(id)/kk
```

```
c ---------- Set break condition
      if(iswch(id).eq.0 .and. nfmly(id,k).eq.1   .or.
     &   iswch(id).eq.1 .and. nfmly(id,k).eq.0) goto 1999

c ---------- Am I spacial node ? [iclstr(id)=7&iam(id)=0]
      if( iswch(id).eq.1 .and. iam(id).eq.0 )

c           <--------- Yes, I am. iii=1:dummy (iii=0:standard)
     & then
           iiam = nhead(id)-nn(id)
           do 1500 iii=1,2
                     myfid = (iam(id)+1)/kk
                     iflag = mod(myfid ,2)
           intvl           = nn(id)*kk*(-1)**iflag
           ndsel0(iii,id,k) = iiam + intvl
                     iiam  = iiam + nn(id)
 1500      continue

c           <--------- NO, I am not. standard
      else
           if(id.eq.1)iiam = nhead(1)+iam(1)
           if(id.eq.2)iiam = iam(2)*nn(2)+iam(1)
                myfid = (iam(id)+1)/kk
                iflag = mod(myfid ,2)
           intvl       = nn(id)*kk*(-1)**iflag
           ndsel(id,k) = iiam + intvl

c                                        ---------- standard to standard
           if(iswch(id).eq.0) then
              if( ndsel(id,k).lt.nhead(id)) then
                  if(id.eq.1) ndsel(id,k)=ndsel(id,k)+iclstr(id)
                  if(id.eq.2) ndsel(id,k)=ndsel(id,k)+nodes
              end  if
           end  if
c                                        ---------- standard to dummy
           if(iswch(id).eq.1) then
               if(ndsel(id,k).lt.nhead(id)) then
                 if(iswch(id).eq.1) ndsel(id,k)=nhead(id)
               end  if
           end  if


      end if

      goto 1000
 1999 continue
      if(id.eq.1) goto 2000

      return
      end
```

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^{5}Pa

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^{5}dyn)	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^{4}St(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）

光量子分子動力学コード［QQQF，MONTEV］のベクトル化および並列化