ベクトル並列型スーパーコンピューター SR8000 利用ガイド
～ FORTRAN によるコンパイル、実行入門（2）

システム運用掛　宮 嵜　　洋
佐々木　一孝

コンパイルオプションを中心に記述した前号では、単一ノード内のプログラムを並列化する要素並列化オプションを紹介しましたが、複数ノードを使用して本格的に並列化するためには通信ライブラリーを用いてプログラミングする必要があります。本稿では簡単なサンプルプログラムと実行例をもとに SR8000 で利用できる各種並列アプリケーションと、数値計算ライブラリーについて紹介します。

4. 並列アプリケーション

分散メモリー型並列計算機である SR8000 でノード間並列実行する場合には各ノードに分散されているメモリー空間を相互にアクセスするためにノード間通信を行なう必要があります。このためメッセージ通信ライブラリーとしてリモート DMA 転送、MPI、PVM、PARALLELWARE が提供されており、これらを利用したプログラムの並列化、並列実行を支援する各種コマンドが用意されています。

prun（並列実行コマンド）

プログラムを複数のノードで並列に実行するコマンドです。定義ファイルを用いるとデーターのファイル名にノード番号を付加することができるので複数の異なる入出力データーに対してプログラムを並列に実行することができます。また、通信ライブラリーを用いた並列プログラムの実行にも使用します。

使用例

```
% cat a.f                                    サンプルプログラム
        read(5,*) a,b
        write(6,*) a+b,a-b
        stop
        end

% f77  a.f  -o  a.out                        ≪コンパイル
f77: compile start : a.f

*OFORT77 V01-01-A entered.
*program name = MAIN
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics                          【続く】
```

```
generated.
% cat  sample.def                                定義ファイル
*2  a.out < data.%n > result.%n                    2 ノードを使用して 2 種類のデーター
% ls                                              data.? をノード毎に入力し、結果を
a.f         a.out        data.1         data.2     result.? に出す。
sample.def
% prun  -f  sample.def                           ≪実行*
% ls                                               prun コマンドを使用（定義ファイル指定）
a.f       data.1       result.1     sample.def
a.out     data.2       result.2                   result.1, result.2 が作成される
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

定義ファイル記述用の記号

リダイレクト出力指定			ファイル名修飾	
>	上書き	標準出力	%n	1～並列プロセス数の数値
>&	〃	標準エラー出力	%r	各プロセスが動作している相対ノード番号
>>	追記	標準出力	%a	各プロセスが動作している絶対ノード座標
>>&	〃	標準エラー出力	%t	prun が起動した時刻
			%d	prun が起動した日付
			%p	起動した並列プロセスのプロセス ID

コマンド、オプション、定義ファイルの詳細はオンラインコマンド “man prun” を参照して下さい。

リモート DMA 転送ライブラリー

リモート DMA（Direct Memory Access） 転送ライブラリーはリモート DMA 転送機能を用いてノード間のメッセージ通信を行なうインターフェースです。リモート DMA 転送は通常のノード間通信と異なり OS を介さず、直接ユーザー空間のデーターを転送するため高速な通信ができます。通信性能を最大限に引き出す場合にはこの通信を使用します。

使用例

```
% cat rdma_sample.f                              サンプルプログラム
      program sample

      integer rank, size
      integer node(2)

      call $rgetnod(node)

      rank=node(1)
      size=node(2)

      write(*,*)  'node=',rank,'size=',size
      stop
      end

% f77  -rdma  rdma_sample.f                       ≪コンパイル
f77: compile start : rdma_sample.f                 静的データー自動リモート DMA 機能を使用
                                                   （プログラム中で *COPTION パラメーターを
*OFORT77 V01-01-A entered.                         定義しているときには-rdma オプションは不
*program name = SAMPLE                             要）
*end of compilation : SAMPLE
*program  units  =  0001,  no  diagnostics                                  【続く】
```

```
generated.
% prun  -n  2  a.out                          ≪実行*
node=          0 size=          2              prun コマンドを使用
node=          1 size=          2
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

参考マニュアル　「リモート DMA 転送使用の手引 -FORTRAN-」(6A30-3-312)
「リモート DMA 転送使用の手引 -FORTRAN77-」(6A30-3-315)

MPI

MPI（Message-Passing Interface）は MPI Forum によって標準化されたメッセージ通信ライブラリーのインターフェース規約です。ノード間およびノード内のプロセス間通信に MPI 通信ライブラリーを用いたメッセージ通信ができます。多くの計算機に実装されており移植性の高いインターフェースです。

使用例

```
% cat mpi_sample.f                          サンプルプログラム
      program sample
      include 'mpif.h'

      integer size, rank, ierr

      call MPI_INIT(ierr)
      call MPI_COMM_SIZE(MPI_COMM_
     &         WORLD, size, ierr)
      call MPI_COMM_RANK(MPI_COMM_
     &         WORLD, rank, ierr)

      write(*,*) 'node=',rank, 'size=',size

      call MPI_FINALIZE(ierr)
      stop
      end

% mpif77  mpi_sample.f                      ≪コンパイル*
f77: compile start : mpi_sample.f            MPI コンパイルコマンドを使用
                                             ※MPI 機能で提供されているコマンドを使用
*OFORT77 V01-01-A entered.                     する場合には環境変数 path の設定が必要
*program name = SAMPLE                        % set path=($path  /usr/mpi/bin)
*end of compilation : SAMPLE
*program units = 0001, no diagnostics generated

% mpirun -n 2 -np 4 a.out                   ≪実行**
mpirun: (I) 4 processes will be created on 2 nodes
                                             MPI 実行コマンドを使用
                                             （2 ノード 4 プロセスで実行）
                                             ※MPI は 1 ノード複数プロセスで実行可
node=          0 size=          4
node=          2 size=          4
node=          3 size=          4
node=          1 size=          4
```

*mpif77 コマンドの代わりに f77 コマンドでも同等のコンパイルが可能です。

```
% f77  -i,U  -nosymnchk  -I/usr/mpi/include  -L/usr/mpi/lib  -lfmpi  -lmpi  mpi_sample.f
```

**sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

参考マニュアル　「MPI・PVM 使用の手引」(6A30-3-026)

PVM

PVM（Parallel Virtual Machine）は複数の計算機による仮想的な並列計算機のメッセージ通信ライブラリーとして発展してきたインターフェースです。ここではノード間通信として PVM 通信ライブラリーを用いたメッセージ通信ができます。

使用例

```
% cat host.f                                  ホストプログラムの例
      program host
      include 'fpvm3.h'
      integer mytid, tids(4), numt
      call pvmfmytid(mytid)
     call pvmfspawn('node', PVMDEFAULT, '*',   （2 ノードを使用する例）
    &               2, tids, numt)
      write(*,*) 'parent =', mytid
      write(*,*) numt, 'tasks:', tids
      call pvmfexit(into)
      stop
      end

% cat node.f                                  ノードプログラムの例
      program node
      include 'fpvm3.h'
      call pvmfmytid(mytid)
      write(*,*) 'task id=',mytid
      stop
      end

% f77  -i,U  -nosymnchk  host.f               ≪コンパイル
       -I/usr/pvm3/include                     ホストプログラム host を作成
        -L/usr/pvm3/lib/$PVM_ARCH
         -lfpvm3 -lpvm3  -o host  -parallel
f77: compile start : host.f

*OFORT77 V01-01-A entered.
*program name = HOST
*end of compilation : HOST
*program  units  =  0001,  no  diagnostics
generated.

% f77  -i,U  -nosymnchk  node.f               ≪コンパイル
       -I/usr/pvm3/include                     ノードプログラム  node を作成
        -L/usr/pvm3/lib/$PVM_ARCH
         -lfpvm3 -lpvm3  -o node  -parallel
f77: compile start : node.f

*OFORT77 V01-01-A entered.
*program name = NODE
*end of compilation : NODE
*program  units  =  0001,  no  diagnostics
generated.

% pvm -batch start                            PVM デーモンの起動
Console ended because pvmd already running.
% host                                        ≪実行*
parent =      267266                          ホストプログラムを実行する（ノードプログ
         2 tasks:       332801        365569  ラムはホストプログラムが生成する）
0           0
% pvm -batch end                              PVM デーモンの終了
pvmd already running.
```

【続く】

```
% ls /tmp                                    実行結果は /tmp に出力される
pvml.10391                  pvml.10391.t51401  （出力先は環境変数 PVM_LOGDIR の設定で変
pvml.10391.t59401                              更できる）
% cat pvml.10391.t51401
 task id=       332801
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

参考マニュアル 「MPI・PVM 使用の手引」(6A30-3-026)

PARALLELWARE

PARALLELWARE（Express）は ParaSoft 社によって開発された並列処理プログラム開発環境ソフトウェアであり、メッセージ通信ライブラリーとデバッガー等の開発支援ツールから構成されています。

使用例（ホスト-ノードモデル）

```
% cat host.f                                 ホストプログラムの例
      program host

      call kxinit()
      pgid=kxrun(2,'node')                   （2 ノードを使用する例）
      call kxclos(pgid)
      stop
      end
% cat node.f                                 ノードプログラムの例
      program node

      integer rank, size
      integer env(4)

      call kxinit()
      call kxpara(env)

      rank=env(1)
      size=env(2)

      write(6,*) 'node=',rank, 'size=',size
      stop
      end

% srf77 -kXH  host.f  -o host               ≪コンパイル
f77: compile start : host.f                  ホストプログラム host を作成

*OFORT77 V01-02 entered.
*program name = HOST
*end of compilation : HOST
*program  units  =  0001,  no  diagnostics
generated.
% srf77 -kXN node.f -o node                 ≪コンパイル
f77: compile start : node.f                  ノードプログラム node を作成

*OFORT77 V01-02 entered.
*program name = NODE
*end of compilation : NODE
*program  units  =  0001,  no  diagnostics
generated.
% ls
host      host.f   host.o   node      node.f
node.o                                                          【続く】
```

```
% host                                          ≪実行*
Using partition: "ALL"                            ホストプログラムを実行する（ノードプログ
Allocated 2 nodes, origin at 0, process id        ラムはホストプログラムが生成する）
378483.
Loading "node" into all processors ...
Loaded, starting ....
node=           1 size=           2
node=           0 size=           2
%
```

使用例（Cubix モデル）

```
% cat cubix.f                                   Cubix プログラムの例
      program node

      integer rank, size
      integer env(4)

      call kxinit()
      call kxpara(env)

      rank=env(1)
      size=env(2)

      call kmulti(6)
      write(6,*) 'node=',rank, 'size=',size
      stop
      end

% srf77 -kcubix cubix.f -o cubix-node          ≪コンパイル
f77: compile start : cubix_cbx.f                  Cubix モデルとしてコンパイル

*OFORT77 V01-02 entered.
   KCHF233C  00                 TIORET
             the variable is defined but never
referred.
*program name = CBXMAIN
*end of compilation : CBXMAIN
*program units = 0001, 0001   diagnostics
generated, highest severity code is  00

% cubix -n 2 cubix-node                        ≪実行*
Cubix Version 3.2.5 -- Copyright (C) 1988-1996    cubix コマンドを使用（2 ノードで実行）
ParaSoft Corp.
All Rights Reserved Copyright (C) 1998,1999,
Hitachi Ltd.
HI-UX/MPP for SR8000 PARALLELWARE 03-00
Using partition: "ALL"
Allocated 2 nodes, origin at 0, process id
378501.
Loading "cubix-node" into all processors ...
Loaded, starting ....
Execution Terminated:
  node=  0  size=  2
  node=  1  size=  2
System 0:1   User 0:0
CUBIX: exit status 0
%
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

参考マニュアル　「PARALLELWARE ユーザーズガイド -FORTRAN-」(6A30-3-400)
「PARALLELWARE リファレンス –FORTRAN-」(6A30-3-401)

Parallel FORTRAN

Parallel FORTRAN は並列処理機能を拡張した FORTRAN 言語である HPF（High Performance Fortran）の指示文を含むプログラムを Fortran90 ソースプログラムに変換するトランスレーターです。プログラム中にコメントで並列化指示文を記述することによって通信ライブラリーを使用したプログラムに変換できます。なお、使用する通信ライブラリーはオプションで選択します。

使用例

```
% cat sample.f                                  サンプルプログラム
       integer a(100)
!hpf$ processors p(2)                            HPF 指示文（2 ノードを使用する例）
!hpf$ distribute a(block) onto p
       do 10 i=1,100
        a(i)=i
 10    continue
       write(*,*)  (a(i),i=1,100)
       end

% pf90 -Wt,'comlib(rdma),pfmsg(1)' sample.f     ≪トランスレート
pf90: translate start : sample.f                 指示文を含むプログラムを並列化ソースプロ
                                                 グラムに変換（ここでは通信ライブラリーと
*Parallel FORTRAN V01-01 entered.                してリモート DMA 関数を使用する）
*program name = MAIN
                                                 ※通信ライブラリーを MPI にする場合
*Parallel FORTRAN V01-01   diagnosis loop        % pf90 -Wt,'comlib(mpi),pfmsg(1)' sample.f
distribution
   KCPF3301T 00  Loop at line 4 is distributed.
*program name=MAIN
*program units = 0001, no diagnostics
generated.
% ls                                             pf_node ディレクトリー下に変換後のソースプ
pf_node    sample.f                              ログラムが置かれる
% ls pf_node
sample.f

% f90 pf_node/sample.f -lhpf                     ≪コンパイル
f90: compile start : pf_node/sample.f            f90 コマンドを使用（HPF ライブラリーを指
                                                 定する）
*OFORT90 V01-01-A entered.
KCHF656K  -I the following file is               ※通信ライブラリーが MPI の場合
referred.--/usr/include/hpf_var_info.f90           % mpif90 pf_node/sample.f -lhpf
KCHF656K  -I the following file is
referred.--/usr/include/hpf_intrinsics.f90       ※MPI 機能で提供されているコマンドを使用す
KCHF656K -I the following file is                    る場合には環境変数 path の設定が必要
             :                                     % set path=($path  /usr/mpi/bin)

*program name = MAIN
*end of compilation : MAIN
*program units = 0001, 0002  diagnostics
generated, highest severity code is  00

% prun -n 2 a.out                                ≪実行*
         1              2              3          prun コマンドを使用
4          5          6                         ※通信ライブラリーが MPI の場合
                                                   % mpiprun -n 2 a.out
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

参考マニュアル　「Parallel FORTRAN 言語」(6A30-3-320)
「Parallel FORTRAN 使用の手引」(6A30-3-321)

5. 数値計算ライブラリー

SR8000 には数値計算ライブラリーとして MATRIX/MPP、MSL2、BLAS、LAPACK、ScaLAPACK があります。これらのライブラリーを利用するには f77、f90 コマンドのオプションとして

–L ライブラリー検索パス名

–l ライブラリー名

を指定します。 –L オプションは –l オプションより前に指定しなければなりませんが、センター提供の数値計算ライブラリーの検索パスは標準で設定されていますので、–L オプションは省略できます。–l オプションは通常プログラムファイル名の後に指定します。このオプションは左から順に処理されるのでライブラリー名を指定する順序には注意して下さい。

MATRIX/MPP

MATRIX/MPP/SSS

基本配列演算、連立 1 次方程式、逆行列、固有値・固有ベクトル、高速フーリエ変換、擬似乱数等に関する副プログラムライブラリーです。並列処理用インターフェースを用いることにより、データーを各ノードに分散して配置、並列に実行することができます。MATRIX/MPP を使用する場合にはコンパイル時にオプションとして以下のライブラリーを指定します。

MATRIX/MPP	（要素並列処理用）	–l***matmpp***
	（スカラー処理用）	–l***matmpp_sc***
MATRIX/MPP/SSS （スカイライン法）	（要素並列処理用）	–l***matmpps***
	（スカラー処理用）	–l***matmpps_sc***

参考マニュアル 「行列計算副プログラムライブラリ MATRIX/MPP」(6A30-7-600)
「行列計算副プログラムライブラリ -スカイライン法- MATRIX/M/SSS」(6A30-7-601)

使用例（単一ノードの例）

```
% cat hsru1m.f                          サンプルプログラム
      parameter ( n=10 )                 逐次処理用インターフェースの例
      implicit real*4(a-h,o-z)
      dimension x(n)
      ix=0
      call hsru1m(n,ix,x,ier)
      do 10 i=1,n
        write(6,*) i,x(i)
   10 continue
      end

% f77 hsru1m.f -lmatmpp_sc              ≪コンパイル
f77: compile start : hsru1m.f            スカラー処理用ライブラリーを使用
                                         （スカラージョブクラス又は sr8000-s で実行でき
*OFORT77 V01-02 entered.                 ます。）
*program name = MAIN                                          【続く】
```

```
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.
% a.out
          1  0.148270369                    ≪実行*
          2  0.158839539                      a.out を実行する
          3  0.645628750
                :
         10  0.629399896
% f77 hsrulm.f -lmatmpp -parallel
f77: compile start : hsrulm.f               ≪コンパイル
                                              要素並列処理用ライブラリーを使用
*OFORT77 V01-02 entered.                     （並列ジョブクラス又は sr8000-p で実行でき
*program name = MAIN                          ます。）
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.
% a.out
          1  0.148270369                    ≪実行**
          2  0.158839539                      a.out を実行
          3  0.645628750
                :
         10  0.629399896
```

* **sr8000-p を使用したインタラクティブ実行例です。（*は sr8000-s でも実行可）

使用例（複数ノード実行の例）

```
% cat hdru3mdp.f                            サンプルプログラム
      parameter ( n=20,npu=2 )
      implicit real*8(a-h,o-z)
      dimension x(n),lstpu(0:npu-1),
     & iopt2(2),ienv(4)
      do 10 k=0,npu-1
         lstpu(k)=k
  10  continue
      iwksize=max(128,(npu+1)*8)
      call hmatinit(iwksize,lstpu,npu,ier)
      call hkxpara(ienv)
      me=ienv(1)
      ix=0
      iopt1=1
      iopt2(1)=1
      iopt2(2)=1
      call  hdru3mdp(n,ix,lstpu,npu,iopt1,     並列処理用インターフェース(2 ノードを使用
     & iopt2,x,ier)                             する例)
      write(6,*) me, n, ier
      do 20 i=1,n
        write(6,*) i,x(i)
  20  continue
      end
% f77 hdru3mdp.f -lmatmpp -parallel          ≪コンパイル
f77: compile start : hdru3mdp.f               要素並列処理用ライブラリーを使用

*OFORT77 V01-02 entered.
*program name = MAIN
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.
% prun -n 2 a.out                           ≪実行*
          0          20           0           prun コマンドを使用
                      :
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大 16 ノードまでの並列化が可能です。

MSL2

行列計算（連立 1 次方程式、逆行列、固有値・固有ベクトル等）、関数計算（非線形方程式、常微分方程式、数値積分等）、統計計算（分布関数、回帰分析、多変量解析等）に関する副プログラムライブラリーです。MSL2 **を使用するためにはコンパイル時にオプションとして以下のライブラリーを指定します。**

MSL 2	**（要素並列処理用）**	**–l*MSL2P***
	（スカラー処理用）	**–l*MSL2***

参考マニュアル 「**数値計算副プログラムライブラリ** MSL2 **操作**」(6A30-7-613)
「**数値計算副プログラムライブラリ** MSL2 **行列計算**」(6A30-7-610)
「**数値計算副プログラムライブラリ** MSL2 **関数計算**」(6A30-7-611)
「**数値計算副プログラムライブラリ** MSL2 **統計計算**」(6A30-7-612)

使用例（スカラー処理の例）

```
% cat msgu1m.f                                  サンプルプログラム
      parameter ( n=10 )
      implicit real*4(a-h,o-z)
      dimension x(n)
      ix=0
      call msgu1m(n,ix,x,ier)
      do 10 i=1,n
        write(6,*) i,x(i)
   10 continue
      end

% f77 msgu1m.f -lMSL2                          ≪コンパイル
f77: compile start : msgu1m.f

*OFORT77 V01-02 entered.
*program name = MAIN
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.

% a.out                                        ≪実行*
          1  0.148270369                        a.out を実行する
          2  0.158839539
          3  0.645628750
             :
         10  0.629399896
```

*sr8000-s **でインタラクティブ実行できます。。**

使用例（要素並列処理の例）

```
% cat msvafm.f                                  サンプルプログラム
       parameter (n=2,m=2)
       real a(n,m),b(n,m),r(n,m)
       iopt=2
       data ((a(i,j),i=1,n),j=1,m)/1,2,3,4/
       data ((b(i,j),i=1,n),j=1,m)/1,2,3,4/
       call msvafm(a,n,m,n,b,n,iopt,r,n,ier)
       write(*,*) ((r(i,j),i=1,n),j=1,m)
       stop                                                    【続く】
```

```
      end

% f77 msvafm.f -lMSL2P -parallel                ≪コンパイル
f77: compile start : msvafm.f

*OFORT77 V01-02 entered.
*program name = MAIN
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.

% a.out                                         ≪実行*
  2.00000000                    4.00000000      a.out を実行する
6.00000000       8.00000000
```

*sr8000-p を使用したインタラクティブ実行例です。なお、以下の副プログラムに要素並列化版があります。

MCLF1M	MDLB1M	MDVAFM	MDVF2M	MSLF1M	MSVB2M	MSVMTM
MDFB2M	MDLF1M	MDVAHM	MDVFHM	MSLK1M	MSVF1M	MZFB2M
MDFC2M	MDLF3M	MDVB1M	MDVMFM	MSVAFM	MSVF2M	MZLF1M
MDFI2M	MDLF5M	MDVB2M	MDVMTM	MSVAHM	MSVFHM	
MDFS2M	MDLK1M	MDVF1M	MSLB1M	MSVB1M	MSVMFM	

BLAS

LAPACK

ScaLAPACK

BLAS（Basic Linear Algebra Subprogram）はベクトル、行列に関する基本演算ライブラリー、LAPACK（Linear Algebra PACKage）は連立一次方程式、固有値、固有ベクトルなどの線形計算ライブラリー、ScaLAPACK（Scalable Linear Algebra PACKage）は並列版の行列計算ライブラリーです。これらのライブラリーの機能の詳細は以下の URL を御覧下さい。

http://www.netlib.org/blas

http://www.netlib.org/lapack

http://www.netlib.org/scalapack

これらのライブラリーを使用する場合にはコンパイル時にオプションとして以下のライブラリーを指定します。（本ライブラリーは要素並列化版のため、要素並列化オプション –parallel も同時に指定して下さい。）

BLAS	–l***blas***
LAPACK	–l***lapack*** –l***blas***
ScaLAPACK*	–l***scalapack*** (ScaLAPACK) –l***pblas*** (PBLAS) –l***tools*** (tools) –l***redist*** (redistribution) –l***blacsBASE*** (BLACS Base) –l***blacsF77*** (BLACS Fortran) –l***blas*** (BLAS)

*ライブラリーオプションの指定順序は使用例を参照して下さい。

なお、ScaLAPACK の通信関数には MPI を使用していますのでコンパイルには MPI ライブラリーの指定も必要です。（mpif77、mpif90 コマンド使用の場合は MPI ライブラリーの指定は省略できます。）

使用例（BLAS）

```
% cat blas.f                                  サンプルプログラム
      program blas
      parameter ( n=4 )
      dimension x(n)
      do 10 i=1,n
        x(i)=1d0*i
   10 continue
      alpha=2d0
      incx=1
      call dscal(n,alpha,x,incx)
      do 20 i=1,n
        write(*,*) x(i)
   20 continue
      stop
      end
% f77 blas.f -lblas -parallel                 ≪コンパイル
f77: compile start : blas.f                    要素並列化オプション–parallel が必要

*OFORT77 V01-02 entered.
*program name = BLAS
*end of compilation : BLAS
*program  units  =  0001,  no  diagnostics
generated.
% a.out                                       ≪実行*
  1.12500000                                     a.out を実行する
  2.00000000
  3.25000000
  4.00000000
```

*sr8000-p を使用したインタラクティブ実行例です。

使用例（LAPACK）

```
cat lapack.f                                 サンプルプログラム
       program lapack
      parameter (n = 2, lda = n+1, ldb = n+1,
     &  nrhs = 1)
      double precision a(lda,n), b(ldb,nrhs)
       integer ipiv(n), info
       data ((a(i,j),j=1,n),i=1,n)/2d0,4d0,
     &  1d0,1d0/
       data (b(i,1),i=1,n)/14d0,5d0/
       call dgesv( n, nrhs, a, lda, ipiv,
     &  b, ldb, info)
       do 10 i=1,n
       write(*,*) b(i,nrhs)
 10    continue
       stop
       end
% f77 lapack.f -llapack -lblas -parallel     ≪コンパイル
f77: compile start : lapack.f                要素並列化オプション–parallel が必要

*OFORT77 V01-02 entered.
*program name = LAPACK
*end of compilation : LAPACK
*program  units  =  0001,  no  diagnostics                              【続く】
```

```
generated.
% a.out
  3.00000000000000000                    ≪実行*
  2.00000000000000000                      a.out を実行する
```

*sr8000-p を使用したインタラクティブ実行例です。

使用例（ScaLAPACK）

```
% mpif77 fexample.f -parallel -lscalapack      ≪コンパイル
     -lpblas -ltools -lredist -lblacsBASE        mpif77 コマンドを使用
    -lblacsF77 -lblacsBASE -lblas
f77: compile start : fexample.f                 MPI 機能で提供されているコマンドを使用す
                                              る場合には環境変数 path の設定が必要
*OFORT77 V01-02 entered.                          % set path=($path  /usr/mpi/bin)
*program name = DLU
*program name = AINITBLK                      サンプルプログラムは
          :                                    /usr/examples/ScaLAPACK/fexamples.f
*end of compilation : PRTMTX                  をコピーしたもの
*end of compilation : AINITMTX
*program units = 0007, 0001   diagnostics
generated, highest severity code is  00
% mpirun -n 2 a.out                           ≪実行*
The input values of matrix A on (0, 0) are :    mpirun コマンドを使用
  0.0  1.00000  1.0000   1.00000    1.00000
          :
The Solutions on (0,  0) are :
   5.60000    4.60000    3.60000    2.60000
1.60000      0.60000
  -0.40000   -1.40000   -2.40000   -3.40000
-4.40000
%
```

*sr8000-p を使用したインタラクティブ実行例です。バッチジョブ（並列ジョブクラス）を使用することにより最大16 ノードまでの並列化が可能です。