RS/6000 SP の使用法

下坂久司

1　RS/6000 SP の構成

1.1　分散メモリ型並列計算機

RS/6000 SP(以下 SP) は「分散メモリ型並列計算機」です．しかしながら，実際には「複数のワークステーション RS/6000(クラスタ) がたまたま一つの箱に入った，RS/6000 の集合体」であると言えます．従って，SP を実行するには，必ず並列化する必要はなく，単体で実行しても一向に構いません．

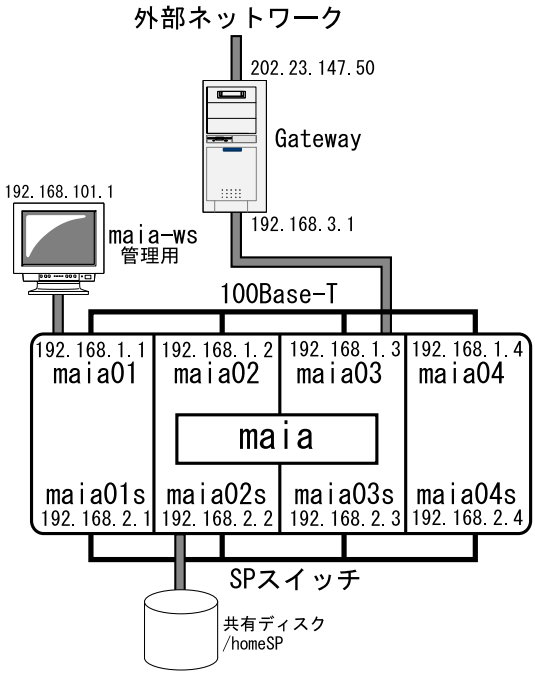

Fig. 1　RS/6000SP の構成 1

SP の全般的な構成は，Fig. 1 のようになっています．今回導入された SP の名前は maia です．４ノードのクラスタシステムです．全般の構成として，外部ネットワークとの出入口としての Gateway，４ノードのクラスタシステムである maia，管理用ワークステーションである maia-ws，共有ディスクで構成されています．

maia は OS に UNIX(IBM 機では AIX と呼ぶ) 採用しています．また標準のシェルは k シェルです．

maia は４ノードのクラスタシステムですが，各ノードは４つの CPU と４ GB の共有メモリを保持しています．よって，並列プログラムを書くには，４プロセッサを使用するスレッドで書くこともできますし，４ノードを使用するプロセスで書くこともできます．また，各ノードに４つずつプロセスを立ち上げ，１６プロセスで並列プログラムを書くこともできますし，４プロセスで各プロセスに４つのスレッドで分割する方法で実行することもできます．

ハードディスクに関しては，maia では各ノードにローカルディスクを保持していますが，全ノードに共通して /homeSP に共有ディスクをマウントしています．よって，並列プログラムは /homeS に置くか各ノードのローカルディスクにそれぞれ同じものを置くことによって，実行することができます．

分散メモリ型システムで並列ジョブを実行する場合，プロセッサ間でデータのやりとりが必要になります．SP では，通信方式に関して，イーサネットと高速な SP スイッチを用意しています．それぞれの使い分けは，並列プログラムを実行する際の，.rhosts ファイルと通信プロトコル等で指定します．その際，イーサネットと SP スイッチに割り当てられているホスト名が異なるので注意してください．

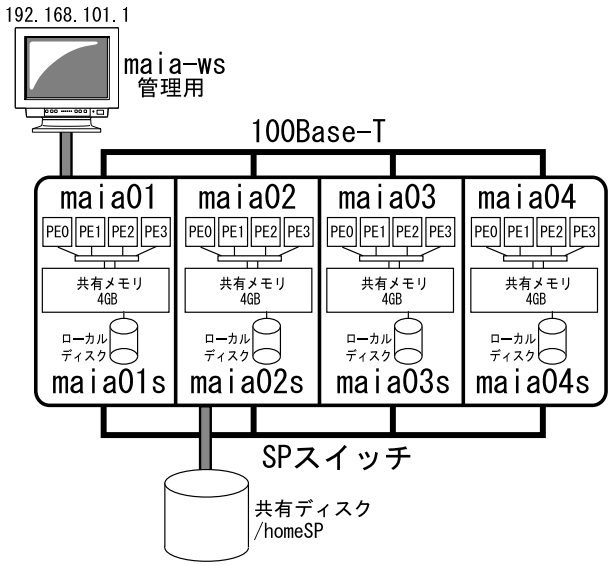

Fig. 2　RS/6000SP の構成 2

2　RS/6000 のソフトウェア

3　プログラムのコンパイルと実行方法

3.1　プログラムの送信

maia の Gateway では，ssh しか接続を許可していないため，ログインする場合は ssh，ファイルを転送する場合は scp などのセキュアな方法しかとることができません．例えば，自分のマシン (例：192.168.6.136) のファイル (例：/home/hisashi/test.c) を Gateway の/homeSP/hisashi に転送したい場合は次のようにコマンドを打ちます．

```
scp /home/hisashi/test.c 202.23.147.50:/homeSP/hisashi
```

また，デフォルトでは，/homeSP/ユーザ名 が，そのユーザのホームディレクトリとなっているので，次のようにすることもできます．

```
scp /home/hisashi/test.c 202.23.147.50:~/
```

ディレクトリ (/home/hisashi/test) をまるまる送りたい場合は，次のようにします．

```
scp -r /home/hisashi/test 202.23.147.50:~/
```

デフォルトでは，ユーザのホームディレクトリは/homeSP/ユーザ名 となっているため，ホームディレクトリにファイルを送信すれば，そのファイルは全てのノードで使用することができます．

3.2　ログイン

外部から maia にログオンするには，まず Gateway にログオンする必要があります．ログオンには ssh を使用します．Gateway にログオンできれば，実際にプログラムをコンパイル，実行するために，子ノードにログインする必要があります．それには telnet を使用します．例えば，子ノードの maia03 にログインするには，次のようにコマンドを打ちます．その後，ユーザ名とパスワードを入力します．

```
telnet maia03
```

3.3　単体プログラムのコンパイルと実行

並列プログラムをコンパイルする前に，その前提として単体プログラム (test.c) をコンパイルする方法を示します．AIX のコンパイラは，C 言語は xlc(C ＋＋はファイルの拡張子を.C にすることによって,C ＋＋言語としてコンパイルしてくれる)，Fortran 言語は xlf です．コンパイルするコマンドは C 言語では xlc，Fortran 言語では xlf です．

```
xlc test.c
```

上の例では，実行ファイルは a.out となる．任意の実行ファイルを作成したい場合は-o オプションを使用する．実行するには，次のように実行ファイルを実行します．

```
./a.out
```

最適化オプションを付けると，次のようになる．

```
xlc -O3 -qstrict -qarch=pwr3 test.c
```

それぞれの最適化オプションの意味は
-qarch=pwr3：POWER3 ハードウェアプラットフォーム (SP のプラットフォーム) 用に高速化された命令を含むオブジェクトを作成する
-O3：最大限の最適化を行なう
-qstrict：-O3 のときのみ有効．このオプションは，ユーザープログラムの意味を変える可能性のある先進的な最適化をオフにする．
その他の最適化オプションについては，man ページ等を参考にしてください．

4 並列プログラムのコンパイルと実行

4.1 mpi を用いた並列プログラム

並列ジョブを実行するには前に，いくつかの環境変数等を指定する必要があります．

4.1.1 .rhosts の指定

並列ジョブを実行する前に，各ノードへのアクセス権限が必要となるので，ホームディレクトリに.rhosts ファイルを作成します．このファイルには，maia の各ノードのホスト名を記述します．

```
maia01s
maia02s
maia03s
maia04s
```

4.1.2 マシンのリスト

mpi でプロセスを発生させて並列プログラムを実行するには，プロセスを発生させるマシンのリストが必要になります．ここでは，例として list というファイルにそのリストを記述します．list には次のように記述します．並列プログラムを実行する際，この list ファイルに記述されているノードを上から順に，使用します．

```
maia01s
maia01s
maia01s
maia01s
maia02s
maia02s
maia02s
maia02s
maia03s
maia03s
maia03s
maia03s
maia04s
maia04s
maia04s
maia04s
```

そして，環境変数 MP_HOSTFILE にその list を指定します．

```
export MP_HOSTFILE=list
```

4.1.3 コンパイルと実行

mpi プログラムをコンパイルするコマンドは C 言語の場合 mpcc，C ＋＋言語なら mpCC，fortran 言語なら mpxlf です．次のように実行します．

```
mpcc -O3 -qstrict -qarch=pwr3 test.c
```

上のようにコンパイルを行った場合，実行ファイルは a.out になります．この並列プログラムを実行するには，a.out に続いて，発生させたいプロセス数を指定します． 4プロセスで実行する方法を次に示します．

```
./a.out -procs 4
```

実行ファイルは各ノードの同じ場所に存在する必要があります．/homeSP に実行ファイルを置いておけば，各ノードがマウントしているので，並列プログラムを実行することができます．しかしながらその場合，最初にネットワークを通して，実行ファイルが各ノードのメモリ上に読み込まれます．これを回避するには，各ノードの同じ場所 (例えば/home 以下) に実行ファイルをコピーしておく必要があります．このことは，ファイルへの出力等でも同じことが言えます．

4.1.4 通信媒体と通信プロトコルの指定

前節までの説明で，並列プログラムを実行することはできますが，並列プログラムを実行する上でもう少し細かい設定を行うことができます．通常，mpi を用いた並列プログラムではメッセージ交換を行うため，通信媒体と通信プロトコルの指定をする必要があります．指定しない場合はデフォルト値がとられますが，変更などを行いたい際は，以下を参考にしてください．

```
export MP_EUIDEVICE=css0 ⋯ (1)
export MP_FUIDEVICE=en0 ⋯ (2)
export MP_EUILIB=us ⋯ (3)
export MP_EUILIB=ip ⋯ (4)
```

通信媒体として，SP スイッチを使用する場合は (1) を，イーサネットを使用する場合は (2) を指定します．通信プロトコルとして，SP スイッチ用に開発された高速な US(ユーザスペース) プロトコルを使用する場合は (3)，IP プロトコルを使用する場合は (4) を指定します．デフォルトでは (1) と (3) の組合せです．またイーサネットと US プロトコルの組合せを指定することはできません．

注意点として，USプロトコルは一つのノード (listに記述されているノード) に対して，一つのプロセスしか実行することはできません．

4.1.5 ノード内プロセス間の通信の高速化

ノード間にまたがるプロセス間の通信はSPスイッチのアダプタを経由しますが，同じノード内のプロセス間の通信はアダプタを経由せずに，メモリ内で直接コピーを行った方が高速になります．それには，以下のように環境変数を指定する必要があります．

```
export MP_WAIT_MODE=poll
export MP_SHARED_MEMORY=yes
```

4.1.6 POEコマンド

並列ジョブの実行は並列操作環境 (POE) がサポートしています．POEコマンドは各ノードで同じAIXコマンドを実行することができます．

```
./a.out -procs 16 ··· (1)
poe a.out -procs 16 ··· (2)
poe AIX コマンド -procs 16 ··· (3)
poe mkdir shimosaka -procs 16 ··· (4)
```

通常，並列プログラムを実行するには (1) のように行いますが，実際は (2) の省略形です．a.out部分をAIXコマンドにすることによって，listと-prcsで指定したノードに対して，同じコマンドを実行することができます．例えば，(4) では全ノードで，shimosakaというフォルダを作成します．

4.2 SMPでの並列プログラムのコンパイルと実行

SPの各ノードは1ノードあたり4つのCPUと4 GBの共有メモリを持つSMPマシンです．よって，1ノードに4つのプロセスを立ち上げて，並列プログラムを実行しても良いのですが，それでは4つのCPUが4 GBの共有メモリを有効に活用しているとは言い切れません．よって，1ノードに4つのスレッドを発生させて並列プログラムを実行する方法について解説します．

4.2.1 並列化の方法

SMPマシンで，並列プログラムをコンパイルする方法には次の2つが存在します．

- 自動並列化
- 指示行による強制並列化

自動並列化では，コンパイラが並列化可能と判断したループに対して，並列化を試みます．また，並列性があるにもかかわらず，何らかの理由によって，コンパイラが自動的に並列化を行わなかったループに対して指示行 (directive) と呼ばれる呼ばれる命令を挿入して，強制的に並列化することをコンパイラに指示することもできます．指示行の仕様はメーカによってばらばらでしたが，指示行を標準化するために，OpenMPというフォーラムが設立され，現在ではOpenMPの仕様が業界標準になっています．OpenMPに関しては，第3回並列ゼミ資料 (http://mikilab.doshisha.ac.jp/dia/seminar/2001/pdf/openmp1.pdf) やOpenMPの公式サイト (http://www.openmp.org/) を参考にしてください．また，IBMの拡張機能などに関しては，参考文献等を参考にしてください．なお，自動並列化の機能に関しては，メーカによって異なるので注意が必要です．

4.2.2 並列プログラムのコンパイル

SMP並列の場合のコンパイルコマンドはC言語では`xlc_r`，C＋＋言語では`xlC_r`，Fortran言語では`xlf_r`を使用します．-qsmpオプションを指定すると，コンパイラがループに対して，自動並列化を試みます．

```
xlc -O3 -qstrict -qarch=pwr3 -qsmp test.c
```

また次のように，-qsourceと-qreport=smplistというオプションを指定すると，test.lstという並列化の報告書が作成されます．

```
xlc -O3 -qstrict -qarch=pwr3 -qsmp -qsource -qreport=smplist test.c
```

並列ジョブを実行する前に環境変数を設定することもできます．

- export YIELDLOOPTIME=500
- export SPINLOOPTIME=500
- OMP_NUM_THREADS=4

環境変数 YIELDLOOPTIME と SPINLOOPTIME は，SMP 並列のパフォーマンスを向上させるためのものです．OMP_NUM_THREADS は並列ジョブで生成したいスレッド数を指定します．これは，プログラム中で指定することもできます．

4.2.3 -qsmp オプション

コンパイラオプションによって，自動並列化と強制並列化を指示できます．

- -qsmp : 自動並列化を行い，指示行が指定されているループを強制的に並列化します．最適化の-qhot オプションを自動で付加します．
- -qsmp=noauto : 自動並列化を行わず，指示行で指定されているループに対してのみ，強制並列化を行います．最適化の-qhot オプションを自動で付加します．
- -qnosmp : 自動並列化も強制並列化も行わない．OpenMP の指示行でしてされたループも逐次で実行します．

4.3 MPI と SMP の混合

MPI と SMP を混合した並列プログラムをコンパイルし，実行することも可能です．コンパイルのコマンドは C 言語では，mpcc_r，C ＋＋言語では mpCC_r，Fortran 言語では mpxlf_r です．maia では，４ノードのクラスタシステムなので，mpi で４プロセスを発生させ，各プロセスを４つのスレッドに分割することで，１ノードあたりの４ CPU を有効に活用することができます．

```
mpcc_r -qsmp -O3 -qstrict -qarch=psw3 sample.c
export MP_HOSTFILE=list
export YIEDLOOPTIME=500
export SPINLOOPTIME=500
export XLSMPORTS=parthds=4 … １プロセスから４スレッド発生
./a.out -procs 4
```

環境変数 XLSMPORTS は，OMP のオプションを使用することも可能です．

参考文献

1) 並列プログラミング虎の巻 MPI 版 日本アイ・ビー・エム株式会社 青山幸也

2) 並列プログラミング虎の巻 SMP 版 日本アイ・ビー・エム株式会社 青山幸也

3) チューニング技法虎の巻 日本アイ・ビー・エム株式会社 青山幸也

文責：下坂久司（hisashi@mikilab.doshisha.ac.jp）