MPI ゼミ

ゼミ担当者 ： 里村 宏章，王 路易
開催日 ： 2009 年 6 月 5 日

1 はじめに

近年，CPU のマルチコア化，メニーコア化が進むにつれ，並列プログラミングが注目されている．一般的に，分散メモリ型並列計算機上の並列プログラミングでは，各ノード上でそれぞれプロセスを走らせる．しかし，各プロセスは独立なアドレス空間を持つため，変数を共有することが出来ない．そこで，適宜，各プロセスが各々のデータを他のプロセスに送信，他のプロセスから受信することにより，各プロセスを協調させる仕組みが考えられている．このデータの送受信をメッセージパッシングという．

分散メモリ型並列計算機におけるメッセージパッシングを用いた並列プログラミングのための Application Programming Interface（API）を提供するライブラリとして，Message Passing Interface（MPI）が注目されている．従来，各社が独自のメッセージパッシングライブラリを用いていたため規格が散在していたが，1990 年代前半に Parallel Virtual Machine（PVM）が登場し，そしてその後 MPI へと移行して，MPI が標準的なメッセージパッシングライブラリとなった．本稿では，MPI の仕様に準じたフリーの実装ライブラリである MPICH を，文法が簡易で可読性に優れるプログラミング言語 Python から利用する方法について説明する．

2 pyMPI を用いたプログラミング

Python で記述したプログラムから MPICH のライブラリを利用するモジュールに pyMPI がある．pyMPI は http://pympi.sourceforge.net/から取得可能である [1]．pyMPI をインストールした環境では，ソースコードに“ import mpi ”を記述することにより，pyMPI のメソッドを利用することができる．本章では，pyMPI の各種メソッドを用いたソースコードの記述方法を述べる．

2.1 属性

MPI を用いたプログラムの実行は，全てのプロセスが同一のソースコードを読み込むことにより行われる．従って，どの処理をどのプロセスに振り分けるかを考慮しながら，ソースコードを記述する必要があるため，各プロセスをソースコード上で識別しなければならない．その識別するための属性が rank である．同様に，使用されるプロセスの総数もソースコード上で把握する必要があり，その属性が size である．pyMPI を用いた場合，“ mpi.rank ” mpi.size ”と記述することにより，rank と size を呼び出すことができる．pyMPI を用いたプログラミングにおける，rank と size の使用例を Fig. 1, Fig. 2 に示す．

```
import mpi
:
if mpi.rank == 0:
    :
else:
    :
:
```

Fig. 1 “ mpi.rank ”の使用例

```
import mpi
:
if mpi.rank == 0:
    for i in range(1,mpi.size):
        :
else:
    :
:
```

Fig. 2 “ mpi.size ”の使用例

2.2 メソッド

pyMPI のメッセージ通信を行うメソッドの中で，使用頻度の高い“ mpi.send() ”と“ mpi.recv() ”について説明する．メッセージ通信の方法は，1 対多と 1 対 1，ブロッキングとノンブロンキングなどに分類することができるが，“ mpi.send() ”および“ mpi.recv() ”は 1 対 1 のブロッキング通信である．

2.2.1 メッセージ送信

“ mpi.send() ”は，ある 1 つのプロセスが，他の 1 つのプロセスにメッセージを送信する際に用いられるメソッドである．送信する内容および送り先のプロセスを指定する必要があるので，以下のように，引数として“ メッセージ内容 ” rank ”を記述しなければならない．

mpi.send(msg,rank)

- msg：送信するメッセージ (型は問わない)
- rank：送信先プロセスの rank

また，“ メッセージ内容 ” rank ”の他に“ tag ”を付与して送信することが可能である．tag は以下のように，3 番目の引数で指定する．

mpi.send(msg,rank,tag)

- tag：付与するタグ（int 型）

“ mpi.send() ”を利用したソースコードの例を Fig. 3 に示す．

```
import mpi
:
if mpi.rank == 0:
      for i in range(1,mpi.size):
           msg = [100*(i-1), 100*i]
           mpi.send(msg,i,tag=22)
else:
     :
:
```

Fig. 3 “ mpi.send() ”の使用例

2.2.2 メッセージ受信

“ mpi.recv() ”は，“ mpi.send() ”と対をなすメソッドであり，受信側プロセスがメッセージを受け取る際に用いられる．以下のように，“ 受信したメッセージ ”と“ 状態 ”の 2 つの変数を返り値として受け取る．また，“ 状態 ”を表す変数は，1 番目の要素に rank，2 番目の要素に tag が格納されている配列である．

msg, status= mpi.recv()

- msg：メッセージを格納する変数
- status：送信元プロセスの rank や状態を格納する変数
  （status[0] に rank, status[1] に tag が格納される）

上記の記述方法は，あらゆるプロセス，あらゆる tag からのメッセージを受け付ける場合を想定している．“ mpi.recv() ”では，以下のように引数に値を記述することにより，受け取ることが可能な送信元プロセス，tag を限定することが可能である．

- 受信可能な送信元プロセスのみを指定（どの様な tag でも受信）
  msg, status= mpi.recv([受信可能な rank])
- 受信可能な tag のみを指定（どの rank からも受信）
  msg, status= mpi.recv(tag == [受信可能な tag])
- 送信元プロセス, tag ともに指定
  msg, status= mpi.recv([受信可能な rank], [受信可能な tag])

“ mpi.recv() ”を利用したソースコードの例を Fig. 4 に示す．

3 pyMPI を用いたプログラムの実行方法

本章では，pyMPI を用いたプログラムの実行方法について述べる．ただし，python, pyMPI，MPICH がインストールされている UNIX ベースのマシン上での作業を前提にしている．

```
import mpi
:
if mpi.rank == 0:
      for i in range(1,mpi.size):
           msg = [100*(i-1),100*i]
           mpi.send(msg,i,tag=22)]
else:
      msg,status = mpi.recv(0,22)
      print str(mpi.rank)+":"+str(msg[0])+"〜"+str(msg[1])
:
```

Fig. 4 “ mpi.recv() ”の使用例

プログラムを実行する際には，“ mpirun ”コマンドを用いる．ただし，オプションや引数により，ノード数，使用言語，使用ノード名を記載したファイルを指定しなければならない．また，使用ノード名を記載したファイルは実行する前に作成する必要がある．ノード数を指定するためのオプションは“ -np ”であり，直後にノード数を記載する．同様に，使用ノード名を記載したファイルを指定するためのオプションは“ -machinefile ”であり，直後にファイルを指定する．実行コマンドの記述例を以下に示す．

```
mpirun -np 10 -machinefile machinename /usr/bin/pyMPI Main.py
```

上記の例では，pyMPI は/usr/bin/以下にインストールされており，使用ノード名を記載したファイル名は“ machinename ”，ソースコードを記述したファイル名は“ Main.py ”であると仮定している．また，“ machinename ”（ファイル名は何でもよい）の記述例を Fig. 5 に示す．

```
isdl043
isdl044
isdl045
isdl046
isdl047
isdl048
isdl049
isdl050
isdl051
isdl052
```

Fig. 5 使用ノード名の記述例

4 おわりに

本稿では，pyMPI を用いたプログラミング方法について述べた．ただし，pyMPI を利用する上での必要最低限の機能にしか触れていない．従って，詳しい機能を知りたい場合，http://pympi.sourceforge.net/から取得可能な pyMPI マニュアル [1] を参照されたい．

参考文献

1) pyMPIsourceforge.net: Putting the py in MPI.
http://pympi.sourceforge.net/