Ejercicios sobre el uso de MPI en el Laboratorio Nacional de Supercómputo del Sureste de México

Luis M. Villaseñor Cendejas
Benemérita Universidad Autónoma de Puebla, Puebla, Mexico
E-mail: lvillasen@gmail.com

2 de diciembre de 2016

1 Introducción

Este documento contiene una lista de ejercicios sobre MPI [1]. Estos ejercicios se proponen para complementar el tutorial sobre MPI que está disponible en [2, 3]. De acuerdo con este tutorial, que explica el uso básico de MPI en C, Fortran, Julia y Python, el lenguaje de programación para hacer estos ejercicios puede ser cualquiera de estos cuatro.

2 Ejercicios

- Elaborar un programa usando MPI para hacer el producto escalar de dos vectores de forma paralela. El producto escalar de dos vectores se obtiene haciendo la suma de los productos entre los elementos de los vectores.
- Elaborar un programa usando MPI para hacer la multiplicacin paralela de una matriz (NxN) con un vector (Nx1) para obtener un vector (Nx1).
- Elaborar un programa usando MPI para hacer la multiplicacin paralela de una matriz (LxM) por una matriz (MxN) para obtener una matriz (LxN).
- Elaborar un programa usando MPI que implemente un "ping-pong", en el que el proceso raíz envía un número a un proceso y este último lo devuelve inmediatamente al proceso raíz aumentado en uno.
- Elaborar un programa usando MPI que implemente el juego de "papa caliente", en el que un proceso aleatorio envía un número a otro proceso escogido al azar, y este último a su vez lo manda a otro proceso, aumentado en uno y también escogido al azar. Cuente el número de envíos antes de que el número regrese al proceso inicial.
- Elaborar un programa usando MPI que determine si el número de unos en una secuencia binaria grande, almacenada en un arreglo, es par o impar.
- Elaborar un programa usando MPI para contar los números primos menores que un número dado. Como guía, en el Listado 1 se muestra un programa secuencial en Python, y en el Listado 2 se muestra otro programa secuencial en Julia, para este propósito.
- Elaborar un programa usando MPI para paralelizar el fractal de Mandelbrot [4]. En el Listado 3 se muestra un programa secuencial escrito en Julia para este propósito. En la Figura 1 se muestra el fractal de Mandelbrot obtenido con este programa. Puede usar el lenguaje de su preferencia.
- Elaborar un programa usando MPI para implementar el "Juego de la Vida" [5]. Como guía, el Listado 4 muestra un programa secuencial escrito en Julia para este propósito. Este código produce la Figura 2. Puede usar el lenguaje de su preferencia.
- Elaborar un programa usando MPI para procesar una malla bidimensional X de 1000×1000 puntos, actualizándola de acuerdo al siguiente esquema iterativo:

Listado 1: Programa secuencial para calcular números primos en Python.

```
import numpy as np
N1 =800000
N2 = 850000
if (N1%2 == 0):
    N1+=1
if (N2%2 == 0):
    N2-=1
n= N1
Num_primos=0
while (n<=N2):
    n1=3
    p=1
    while (n1<np.sqrt(n) and p ==1):
        if (n%n1 == 0):
            p=0
        n1+=2
    if (p==1):
        Num_primos+=1
    n+=2
print "Num_primos=",Num_primos
```

$$X_{i,j}^{t+1} = (X_{i+1,j+1}^{t} + X_{i-1,j-1}^{t} + X_{i+1,j-1}^{t} + X_{i-1,j+1}^{t})/4$$

Para resolver el problema en los bordes de la malla considere condiciones periódicas. Suponga que los valores iniciales son $X_i = 10$ en los bordes de la malla y $X_i = 0$ en los puntos interiores. Establecer un criterio de convergencia.

Bibliografía

[1] http://dl.acm.org/citation.cfm?id=898758
[2] https://github.com/lvillasen/TutorialMPI
[3] http://www.lns.org.mx/node/131
[4] https://en.wikipedia.org/wiki/Mandelbrot_set
[5] https://en.wikipedia.org/wiki/Conway's_Game_of_Life

Listado 2: Programa secuencial para calcular números primos en Julia.

```
N1 =8000000
N2 = 8500000
if N1%2 == 0; N1+=1;end
if N2%2 == 0; N2-=1;end
n= N1
Num_primos=0
while n<=N2
    n1=3
    p=1
    while n1<sqrt(n) && p ==1
        if n%n1 == 0;p=0; end
        n1+=2
    end
    if p==1;Num_primos+=1 ;end
    n+=2
end
print("Num_primos=$Num_primos \t")
```

Listado 3: Programa secuencial para graficar el fractal de Mandelbrot en Julia.

```
 using PyCall
@pyimport matplotlib.pyplot as plt
const N=800
function mandelbrot(x0,y0,side,L=30,R=3.)
    m = zeros(N,N)
    delta = side/N
    for i=1:N, j=1:N
        c = x0+delta*i+(y0+delta*j)*im
        z, h = 0+0*im, 0
        while (h<L) && (abs(z)<R)
            z = z^2+c
            h+=1
        end
        m[j,i]=h
    end
    return m
end
n=2.6
m=mandelbrot(-n/1.5,-n/1.9, n)
plt.imshow(m)
plt.show()
```

Listado 4: Programa secuencial del "Juego de la Vida" en Julia.

```
#Usage julia GameOfLife.py n n_iter m
# n is the board size
# n_iter the initial number of iterations
# m is 1 to plot the board or 0 for no graphical output
using PyCall
@pyimport matplotlib.pyplot as plt
const n = int(ARGS[1])
n_iter = int(ARGS[2])
const m = int(ARGS[3])
function life_game(n,M,n_iter)
    for k=1:n_iter
        C = copy(M) # copy current life grid
        for i=1:n, j=1:n
            if (i==1) i_left=n else i_left=i-1 end
            if(i==n) i_right=1;
            else i_right=i+1;end
            if(j==1) j_down=n; else j_down=j-1;end
            if(j==n) j_up=1;
            else j_up=j+1;end
            count = C[i_left,j] + C[i_left,j_up] + C[i_left,j_down] +
             C[i,j_down] + C[i,j_up] + C[i_right,j] +
             C[i_right,j_up] + C[i_right,j_down]
            if C[i, j]==1
                if count < 2 || count > 3
            M[i, j] = 0 # living cells with <2 or >3 neighbors die
                end
            elseif count == 3
                M[i, j] = 1 # dead cells with 3 neighbors are born
            end; end; end
    return M
end
M0=rand(0:1,n,n) # Initialize with a random pattern
M=life_game(n,M0,n_iter)
print(sum(M)," live cells after ",n_iter," iterations\n")
if m != 0
    fig = plt.figure(figsize=(12,12))
    plt.gray()
    while m != 0
        M1=abs(1-M)
        plt.imshow(M1, interpolation="none")
        plt.title("Conway's Game of Life, $n_iter Iterations")
        plt.pause(.01)
        #print(" k=",n_iter," live cells=",sum(M),"\n")
        M=life_game(n,M,1)
        n_iter=n_iter+1; end; end
```

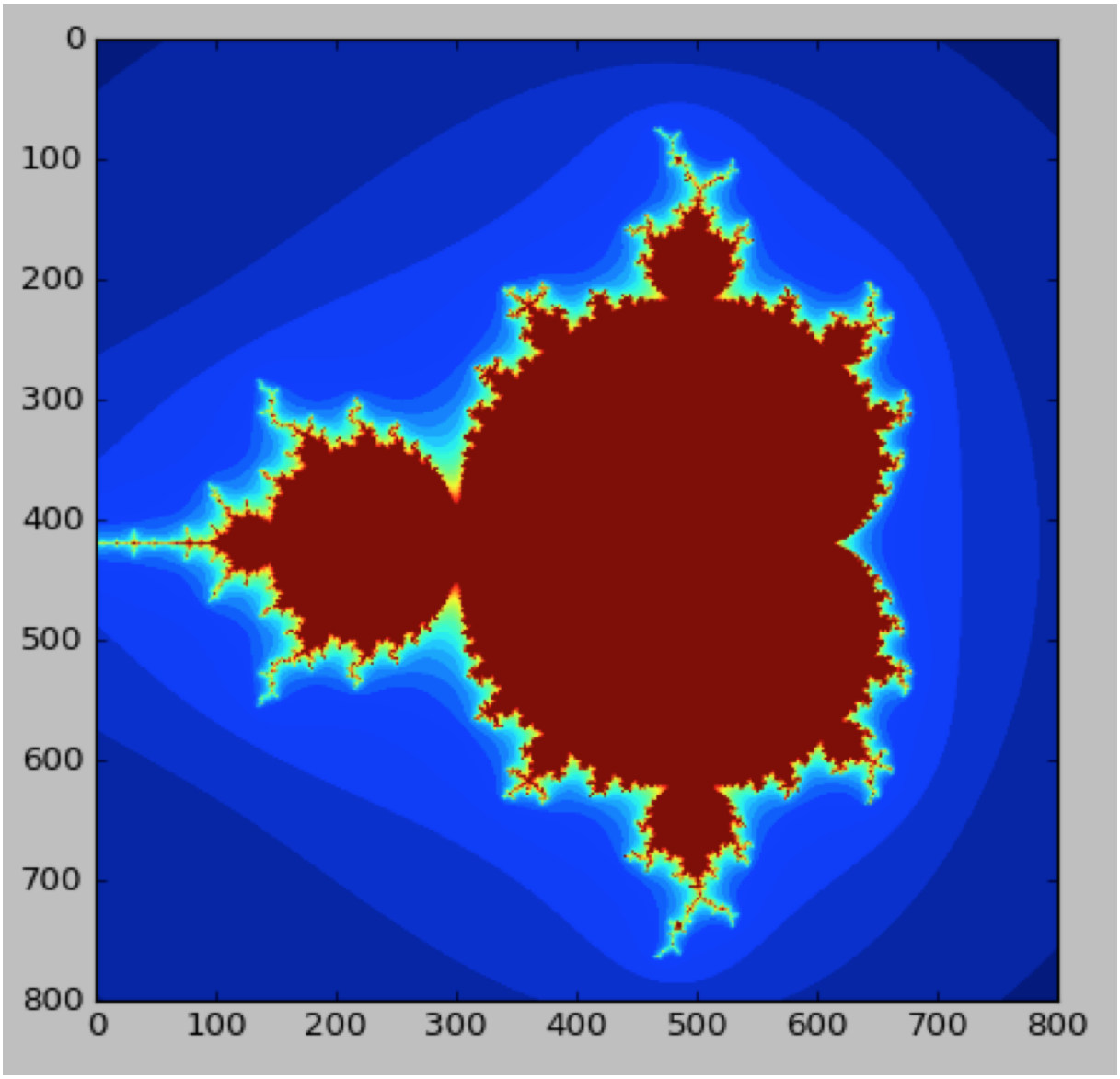

Figure 1: Gráfica del fractal de Mandelbrot obtenida con el programa mostrado en el Listado 3.

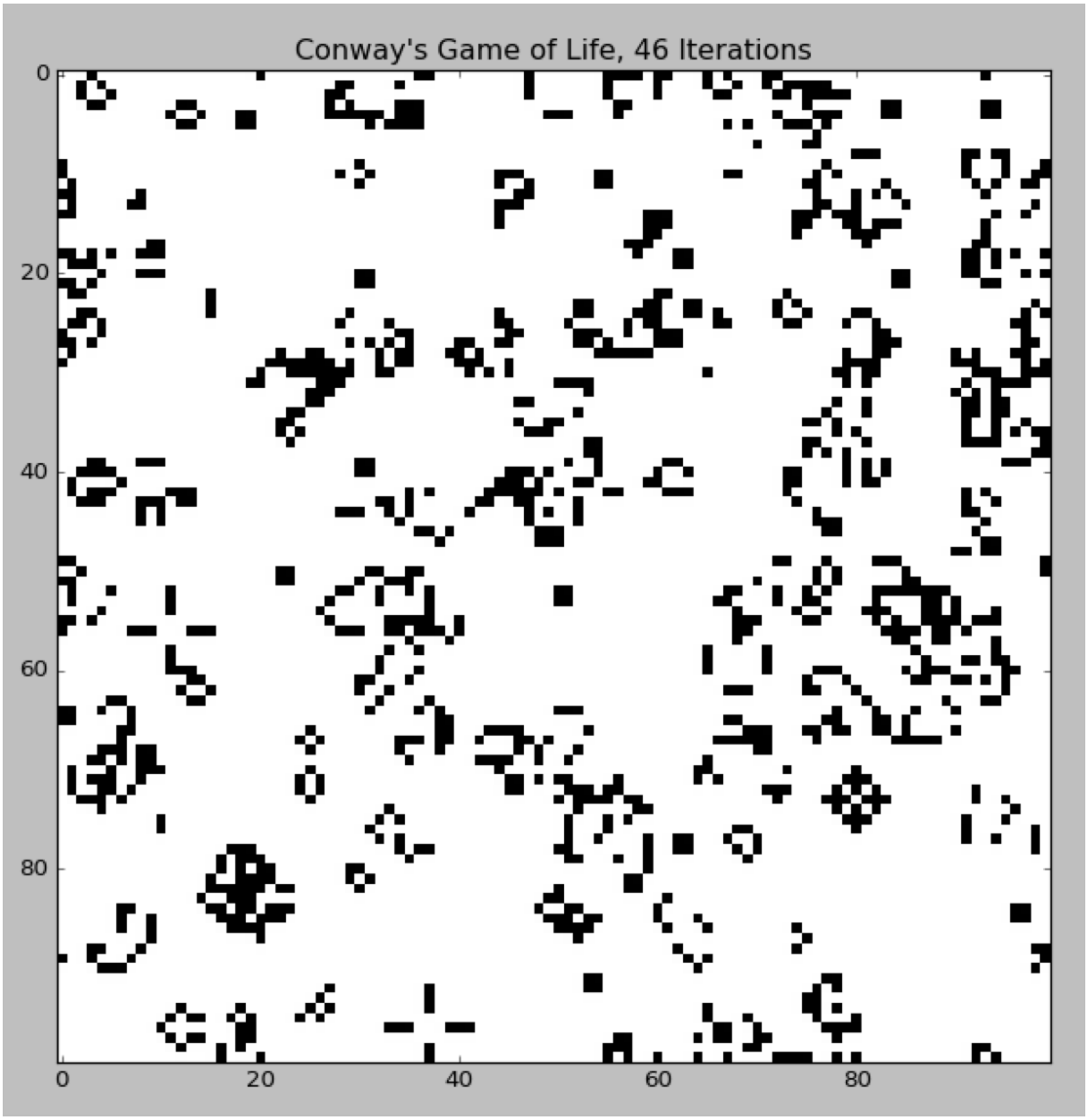

Figure 2: Gráfica del "Juego de la Vida" obtenida con el programa mostrado en el Listado 4.