

Homework #1

Due April 15, 2005, 5pm

Problem #1:

Consider a code in which a Laplacian smoothing is iteratively performed on the $a(i, j)$ array with a smoothing coefficient $\epsilon = 0.2$. The value of $b(i, j)$ is computed from all neighbors including the four diagonals using the stencil described in the following code.

```
program main
.....
dimension a(n1,n2),x(n1),y(n2)
c initialize array x and y
  do i=1,n1
    x(i)= 1./float(n1)*(.5+(i-1))
  end do
  do j=1,n2
    y(j)= 1./float(n2)*(.5+(j-1))
  end do
c initialize array a
  do j=1,n2
    do i=1,n1
      if (x(i).lt.0.5) then
        a(i,j)= cos( x(i)+y(j))
      else
        a(i,j)= sin( x(i)+y(j))
      end if
      b(i,j)=a(i,j)
    end do
  end do
c perform Laplacian smoothing in interior points
  do n=1,iter
    do j=2,n2-1
      do i=2,n1-1
        b(i,j)= b(i,j) + epsilon * (
&          b(i-1,j+1)+  b(i,j+1)+b(i+1,j+1)
&          +b(i-1,j )-8.*b(i,j )+b(i+1,j )
&          +b(i-1,j-1)+  b(i,j-1)+b(i+1,j-1)
&          )
      end do
    end do
  end do
end
```

Write an MPI program that distributes the domain $[0, 1] \times [0, 1]$ using p_1 processors in the x direction and p_2 in the y direction, using the following MPI calls for sends and receives:

- i) Blocking send and recv: `MPI_SEND`, `MPI_RECV`
- ii) Send-recv: `MPI_SENDRECV`
- iii) Buffered send: `MPI_BSEND`, `MPI_RECV`
- iv) Nonblocking send and recv: `MPI_ISEND`, `MPI_IRECV`

Test the code for the following parameters:

1. $n_1 = 90, n_2 = 60, p_1 = 3, p_2 = 3$
2. $n_1 = 125, n_2 = 90, p_1 = 4, p_2 = 4$

For the last set of parameters, plot the initial values a and the solution b after 10 iterations in the whole domain and along the lines $x = 0.5$ and $y = 0.45$.

Run the four different versions of the code on junior with the parameters $n_1 = 1024, n_2 = 1024, p_1 = 4, p_2 = 4, iter = 100$ and compare the execution times. Also provide the source code and line plots of the final solution along row and column 512 for all four communication strategies. Explain the differences in run time.

Extra Credit (10%): Answer the following questions and provide any evidence you may be able to obtain.

1. If you were only allowed to distribute the domain in either horizontal ($p_1 = 1$) or vertical ($p_2 = 1$) slices, would you get better scalability and performance than with the approach you were asked to take (p_1 and p_2 different from 1, for the same number of processors)? Why?
2. Do you think it is possible to obtain parallel speedups that are higher than the actual number of processors you are using in a calculation (superlinear scalability)? Can you show this on an actual parallel computer such as *junior.stanford.edu*? For what problem sizes and number of processors can you do this?

Problem #2: Write a paragraph describing your area of interest in parallel and high performance computing