Homework #1

Due March 9, 2011

Problem 0 – Setting up your parallel computing environment

Setup your own MPI environment. A Unix/Linux/Mac box is probably best, although Windows works too. Feel free to use any compiler you want (C/C++/Fortran). The GNU compilers (gcc, g++, gfortran) are freely available and perform very well. Similarly, feel free to use your favorite MPI implementation. We advise you to use either MPICH (`http://www.mcs.anl.gov/research/projects/mpich2/`) or OPENMPI (`http://www.open-mpi.org/software/ompi/v1.4/`).

Problem 1 – Laplacian in parallel

Consider a code in which a Laplacian smoothing is iteratively performed on the $a(i,j)$ array with a smoothing coefficient $\varepsilon = 0.1$. 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),b(n1,n2)
  dimension x(n1),y(n2)
  ! 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
  ! 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
  ! Perform Laplacian smoothing in interior points
  do n=1,iter
     do j=2,n2-1
        do i=2,n1-1
           b(i,j)=a(i,j)+epsilon*(&
                  a(i-1,j+1)+   a(i,j+1)+a(i+1,j+1)&
                 +a(i-1,j  )-8.*a(i,j  )+a(i+1,j  )&
                 +a(i-1,j-1)+   a(i,j-1)+a(i+1,j-1)&
                )
        end do
     end do
     a=b
  end do
end program main
```

Figure 1: Serial pseudo-code for Laplacian smoothing.

Write an MPI program that distributes the domain $[0,1] \times [0,1]$ using $p1$ processors in the x direction and $p2$ processors in the y direction, using the following MPI calls for send and receive:

i) Blocking send and recv: MPI_SEND, MPI_RECV

ii) Send-recv: MPI_SENDRECV

iii) Buffered send: MPI_BSEND, MPI_RECV

iv) Non-blocking send and recv: MPI_ISEND, MPI_IRECV

Test the code with the following parameters:

1. $n1 = 90$, $n2 = 60$, $p1 = 3$, $p2 = 3$
2. $n1 = 125$, $n2 = 90$, $p1 = 4$, $p2 = 4$

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 $y = 0.45$. We'll wait until we have access to Janus to compare execution times.

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

1. If you were allowed to distribute the domain in either the horizontal ($p1 = 1$) or vertical ($p2 = 1$) slices, would you get better scalability and performance than with the approach you were asked to take ($p1$ and $p2$ 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 provide an detailed example for which you would be able to obtain superlinear scalability?

Problem 2

Write a paragraph describing your area of interest in parallel and high performance computing.