3ba5 - Computer Engineering - 15 April 2005

Project 3 - MPI Message Passing- part a.

Description

The following procedure will copy the shifter example C source and make file into a subdirectory of your home directory, will compile and link it, and finally run it as five processes.

1. Make a subdirectory in your home directory:

```
CHIMPHOME=/usr/local/depot/chimpv2.1.1c
CHIMPARCH=sun5
export CHIMPHOME CHIMPARCH
cd $HOME
rm -rf chimp
mkdir chimp
```

2. Copy the following files into $HOME/chimp as follows:

```
cd /usr/local/depot/chimpv2.1.1c/examples
cp -pr  shifter-mpi make $HOME/chimp
```

3. Compile the shifter-mpi example in your subdirectory:

```
cd $HOME/chimp/shifter-mpi
make sun5
```

4. Run the example with parameters of your choosing, e.g.

```
chimp run.chp 5 sun5
```

i. Follow the above procedure and experiment with step 4 in order to see it running on a different number of processes.
ii. Modify the source code so that the numbers circulate to the process with rank one *below* them and, b) each process accumulates and then prints the sum of *all* the rank numbers.
iii. Modify your code of part ii so that each process initialises a *vector* **a** of length three such that process P_i with rank = i will have $a_0 = i$, $a_1 = i+1$, $a_2 = i-1$. The processes then circulate these vectors as in part ii and, each accumulates all the vectors into a **sum** vector, which they then print before halting.

Project objective

No submissions are expected from this part a) but completion of it will substantially assist you with part b) of this project.

Project 3 b) - MPI Parallel Matrix Multiplication

Description

1. The purpose of this section is to encourage you to develop an MPI implementation of the mesh-based parallel matrix multiplication algorithm which works correctly.
   Warning – using malloc causes MPI problems so use static arrays.

a) Write a procedure which will deterministically generate non-trivial nxn test matrices A and B, ie. design suitable functions f and g, $f \neq g$, and assign $a_{ij} = f(i,j,n)$ and $b_{ij} = g(i,j,n)$. Values that readily identify the elements coordinates will assist you in the debugging.

b) Write a procedure to compute the matrix product C = A.B using the classical SISD matrix multiplication algorithm.

c) Write an MPI application which, given n, will generate nxn matrices A and B on n^2 P_{ij}, where P_{ij} computes a_{ij} and b_{ij}, and will then multiply them in parallel to compute C = A.B, using the mesh-based algorithm so that each P_{ij} completes storing c_{ij}. Verify that your algorithm works correctly by using the procedures developed in parts a) and b).

d) Write an MPI application which, given n, will generate nxn matrices A and B on m^2 P_{ij}, where n=lxm and l is integer, so that initially P_{ij} computes A_{ij} and B_{ij}, the i-j^{th} lxl submatrices of A and B. Then the m^2 P_{ij} compute C = A.B using the mesh-based algorithm so that each P_{ij} completes storing C_{ij}. Again verify that your algorithm works correctly using the procedures developed in parts a) and b).

2. Discuss, but do <u>not</u> implement, ways in which your algorithm of part 1 d) could be -

a) Adapted to handle the case when m does not divide n evenly.

b) Optimised to try to minimise the total execution time of the application for a wide range of n.

Project Deadline and Regulations

Project submissions incorporating your application designs, results and documentation are to be handed in to the Secretaries' Office, O'Reilly Institute by 4:30 pm Friday 20 May 2005.