High Performance Computing (Nov 5)

Question

Write a parallel version of the code in Slide 47, and paste your code here. How much does parallel execution with 2 nodes reduce the execution time?

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc,char* argv[])
{
  int N = 1000;
  int i,total = 0;
  double x,y;

  srand(time(NULL)); /* initialization */
  for(i=0;i<N;i++) {
    /* two random numbers 0<x,y<1 */
    x = (double)rand()/RAND_MAX;
    y = (double)rand()/RAND_MAX;
    if( x*x + y*y < 1 ) {
      total = total+1;
    }
  }
 printf( ”pi=%lf¥n”,4.0*total/N);
}
```

Answer

In the serial version, the total number of points in the quarter circle is kept in a variable, named total. If the program is simply executed by multiple processes in parallel, each process will have the subtotal of points in the quarter circle. Therefore, a parallel version needs to gather subtotals to calculate the total number of points in the quarter circle. A simple way of doing this is to use MPI_Reduce as follows.

```
-------------------------------------------------------------------------
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc,char* argv[])
{
  int N = 1000;
  int i, total = 0;
  int id, np, subtotal = 0;
  double x,y;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &id);
  MPI_Comm_size(MPI_COMM_WORLD, &np);
  srand(time(NULL)); /* initialization */
  for(i=0;i<N/np;i++) {
    /* two random numbers 0<x,y<1 */
    x = (double)rand()/RAND_MAX;
    y = (double)rand()/RAND_MAX;
    if( x*x + y*y < 1 ) {
      subtotal = subtotal+1;
    }
  }
  MPI_Reduce(&subtotal, &total, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
  printf( ”pi=%lf¥n”,4.0*total/N);
  MPI_Finalize();
  return 0;
}
-------------------------------------------------------------------------
```

You would observe that the execution time is not halved by running the program with 2 nodes and could rather increase with the number of nodes. This is because the execution time of the subtotal calculation is too short and hence the communication time is dominant in the total execution time. The subtotal calculation time becomes longer with the problem size, N, while the communication

time does not depend on N, so the computation time gradually becomes dominant by increasing N. When N is sufficiently large, the execution time would be almost proportional to the number of nodes. In such a situation, the performance is expressed as “scalable with the number of nodes.”