Announcements

- Programming Assignment #1 is available on web
  - 5% extra credit for turning in journal of time for study

MPI Calls

- Include <mpi.h> in your program
- If using mpich, …

- First call MPI_Init(&argc, &argv)
- MPI_Comm_rank(MPI_COMM_WORLD, &myrank)
  - Myrank is set to id of this process
- MPI_Wtime
  - Returns wall time
- At the end, call MPI_Finalize()

MPI Communication

- Parameters
  - var – a variable
  - num – number of elements in the variable to use
  - type {MPI_INT, MPI_REAL, MPI_BYTE}
  - root – rank of processor at root of collective operation
  - dest – rank of destination processor
  - status - variable of type MPI_Status;
- Calls (all return a code – check for MPI_Success)
  - MPI_Send(var, num, type, dest, tag, MPI_COMM_WORLD)
  - MPI_Recv(var, num, type, dest, MPI_ANY_TAG, MPI_COMM_WORLD, &status)
  - MPI_Bcast(var, num, type, root, MPI_COMM_WORLD)
  - MPI_Barrier(MPI_COMM_WORLD)

Programming Assignment Notes

- Assume that memory is limited
  - don’t replicate the board on all nodes
- Need to provide load balancing
  - goal is to speedup computation
  - must trade off
    - communication costs of load balancing
    - computation costs of making choices
    - benefit of having similar amounts of work for each processor
- Consider “back of the envelop” calculations
  - how fast can mpi move data?
  - what is the update time for local cells?
  - how big does the board need to be to see speedups?

OpenMP

- Support Parallelism for SMPs
  - provide a simple portable model
  - allows both shared and private data
  - provides parallel do loops
- Includes
  - automatic support for fork/join parallelism
  - reduction variables
  - atomic statement
    - one processes executes at a time
  - single statement
    - only one process runs this code (first thread to reach it)

Sample Code

```
program compute_pi
    integer n, i
    double precision w, x, sum, pi, f, a
c function to integrate
    f(a) = 4.d0 / (1.d0 + a*a)
    print *, \021Enter number of intervals: \021
    read *,n
c calculate the interval size
    w = 1.0d0/n
    sum = 0.0d0
!$OMP PARALLEL DO PRIVATE(x), SHARED(w)
!$OMP& REDUCTION(+: sum)
    do i = 1, n
       x = w * (i - 0.5d0)
       sum = sum + f(x)
    enddo
    pi = w * sum
    print *, \021computed pi = \021, pi
    stop
    end
```