Assignment 3

Timothy H. Kaiser, PH.D.
tkaiser@mines.edu
N-body - Bag of tasks

- Goal: Pull together a number of concepts we have discussed
  - Basic MPI
  - Bag of tasks
  - OpenMPI
  - Scripting
  - Get you to think about improving programs
  - Profiling
  - Hybrid Programming
  - Communicators
N-body - Bag of Tasks

- New concepts
- Libraries
- Transactional Memory
- Phi programming
- These will not be covered in this part of the assignment
Scaled back assignment

- Have workers be completely independent
- Each reads its own data
- Writes its own output
- Gets command line arguments from the manager
- Manager tells workers when to quit
The beginning

• Add code to create a basic MPI program
• Redirect where nnnn = MPI Task number
  • stdin -> mpi.nnnn
  • stdout -> out.nnnn
  • stderr -> err.nnnn
Scaled back First Part

- MPI master reads and processes command line arguments
- Bcasts these to all processors
- Master becomes the manager
  - Does no additional part of the original calculation
  - Only tracks the time for each worker, telling them when to quit and
  - Each worker runs the rest of the normal calculation, sending its current time to the manager
int main(int argc, char *argv[]) {
    real dt_param = 0.03;       // control parameter to determine time step size
    real dt_dia = 1;           // time interval between diagnostics output
    real dt_out = 1;           // time interval between output of snapshots
    real dt_tot = 10;          // duration of the integration
    bool init_out = false;     // if true: snapshot output with start at t = 0
    // with an echo of the input snapshot
    bool x_flag = false;       // if true: extra debugging diagnostics output

    if (!read_options(argc, argv, dt_param, dt_dia, dt_out, dt_tot, init_out, x_flag))
        return 1;                // halt criterion detected by read_options()

    int n;                       // N, number of particles in the N-body system
    cin >> n;

    real t;                      // time
    cin >> t;

    real * mass = new real[n];   // masses for all particles
    real (* pos)[NDIM] = new real[n][NDIM]; // positions for all particles
    real (* vel)[NDIM] = new real[n][NDIM]; // velocities for all particles

    get_snapshot(mass, pos, vel, n);
    evolve(mass, pos, vel, n, t, dt_param, dt_dia, dt_out, dt_tot, init_out, x_flag);

    delete[] mass;
    delete[] pos;
    delete[] vel;
}
int main(int argc, char *argv[]) {
    real dt_param = 0.03;     // control parameter to determine time step size
    real dt_dia = 1;          // time interval between diagnostic output
    real dt_out = 1;          // time interval between output of snapshots
    real dt_tot = 10;         // duration of the integration
    bool init_out = false;    // if true: snapshot output with start at t = 0
                               // with an echo of the input
    bool x_flag = false;      // if true: extra debugging diagnostics output

    if (! read_options(argc, argv, dt_param, dt_dia, dt_out, dt_tot, init_out, x_flag))
        return 1;                // halt criterion detected by read_options()

Bcasts the command line arguments
Bcasts the command line arguments
Does the com_split
Does the com_split
Calls the master subroutine
Calls the master subroutine
Workers do this...

Gets the command line arguments from the master
Does the com_split
Calls the worker subroutine which contains the following...

```c
{  
    int n;                       // N, number of particles in the N-body system  
    cin >> n;  
    real t;                      // time  
    cin >> t;  
    real * mass = new real[n];                  // masses for all particles  
    real (* pos)[NDIM] = new real[n][NDIM];     // positions for all particles  
    real (* vel)[NDIM] = new real[n][NDIM];     // velocities for all particles  
    get_snapshot(mass, pos, vel, n);  
    evolve(mass, pos, vel, n, t, dt_param, dt_dia, dt_out, dt_tot, init_out,  
           x_flag);  
    delete[] mass;  
    delete[] pos;  
    delete[] vel;  
}```
Manager Routine Details

- Manager Routine
  - Gets passed the end_time from main
  - Gets current time from worker
  - If time < end_time just return it
  - If time >= end_time don’t send anything back but increment the number of done workers
  - When all of the workers are done send them a negative time, telling them to quit
Worker Routine Details

- Worker
- Sets up and calls a modified version of the routine “evolve”
- Modify the routine “evolve”
  - Create the variable
    - `real dt_com = dt_out/10.0;`
  - Add the block:
    ```
    if (t >= t_com){
      t_com = t +  dt_com;
      MPI_Send(t,...)
      MPI_Rec(t,...)
    }
    ```
  - Replace:
    ```
    if (t >= t_end)
      break;
    ```
  - With:
    ```
    if (t < 0)
      break;
    ```
Provided with…

- http://hpc.mines.edu/examples/nbody/index.html
- http://hpc.mines.edu/examples/index.html
Run on AuN and Mc2

- AuN
  - 16 MPI tasks on 1 node
- Mc2
  - 16 MPI tasks on 1 node
  - 1 MPI task on each of 16 nodes
Turn in via Blackboard

- Tar ball for Aun and a separate tar ball for Mc2
- Source
- Scripts used to run
- Outputs from runs
- Friday Nov 15, 5:00 PM
Where we may go in the future

- May give you the full example which you can use to:
  - Profile
  - OpenMPI/Hybrid
  - Run on Phi nodes
  - Transactional Memory