Hybrid MPI+OpenMP Parallel MD

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
University of Southern California

Email: anakano@usc.edu
Hybrid MPI+OpenMP Programming

Each MPI process spawns multiple OpenMP threads
MPI+OpenMP Calculation of $\pi$

- Each MPI process integrates over a range of width $1/nproc$, as a discrete sum of $nbin$ bins each of width $\text{step}$
- Within each MPI process, $nthreads$ OpenMP threads perform part of the sum as in $\text{omp\_pi.c}$
MPI+OpenMP Calculation of $\pi$: hpi.c

```c
#include <stdio.h>
#include <mpi.h>
#include <omp.h>
#define NBIN 100000
#define MAX_THREADS 8
void main(int argc, char **argv) {
    int nbin, myid, nproc, nthreads, tid;
    double step, sum[MAX_THREADS] = {0.0}, pi = 0.0, pig;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &nproc);
    nbin = NBIN/nproc; step = 1.0/(nbin*nproc);
    #pragma omp parallel private(tid)
    {
        int i;
        double x;
        nthreads = omp_get_num_threads();
        tid = omp_get_thread_num();
        for (i=nbin*myid+tid; i<nbin*(myid+1); i+=nthreads) {
            x = (i+0.5)*step; sum[tid] += 4.0/(1.0+x*x);
            printf("rank tid sum = %d %d %e\n", myid, tid, sum[tid]);
        }
        for(tid=0; tid<nthreads; tid++) pi += sum[tid]*step;
        MPI_Allreduce(&pi, &pig, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
        if (myid == 0) printf("PI = %f\n", pig);
    }
    MPI_Finalize();
}
```
MPI+OpenMP Example: hpi.c

- **Compilation on hpc.usc.edu**
  
  ```
  source /usr/usc/mpich/default/gm-intel/setup.csh
  mpicc -o hpi hpi.c -openmp
  ```

- **PBS script**
  
  ```
  #!/bin/bash
  #PBS -l nodes=2:ppn=1
  #PBS -l walltime=00:00:59
  #PBS -o hpi.out
  #PBS -j oe
  #PBS -N hpi
  source /usr/usc/mpich/default/gm-intel/setup.sh
  export OMP_NUM_THREADS=2
  WORK_HOME=/auto/rcf-12/anakano/hpc/cs596/
  cd $WORK_HOME
  np=$(cat $PBS_NODEFILE | wc -l)
  mpirun -np $np -machinefile $PBS_NODEFILE ./hpi
  ```

- **Output**
  
  ```
  rank tid sum = 1 1 6.434981e+04
  rank tid sum = 1 0 6.435041e+04
  rank tid sum = 0 0 9.272972e+04
  rank tid sum = 0 1 9.272932e+04
  PI = 3.141593
  ```
OpenMP threads handle blocks of linked-list cells in each MPI process (= spatial-decomposition subsystem)
Variables

- \texttt{vthrd[0|1|2]} = \# of OpenMP threads per MPI process in the x|y|z direction.
- \texttt{nthrds} = \# of OpenMP threads = \texttt{vthrd[0]} \times \texttt{vthrd[1]} \times \texttt{vthrd[2]}.
- \texttt{thbk[3]}: \texttt{thbk[0|1|2]} is the \# of linked-list cells in the x|y|z direction that each thread is assigned.

```c
/* Compute the \# of cells for linked-list cells */
for (a=0; a<3; a++)
  lc[a] = al[a]/RCUT; /* Cell size ≥ potential cutoff */
/* Size of cell block that each thread is assigned */
for (a=0; a<3; a++)
  thbk[a] = lc[a]/vthrd[a]; /* # of cells = integer multiple of the # of threads */
for (a=0; a<3; a++) {
  lc[a] = thbk[a]*vthrd[a]; /* Adjust # of cells/MPI process */
  rc[a] = al[a]/lc[a]; /* Linked-list cell length */
}```
OpenMP Threads for Cell Blocks

Variables
- **std** = scalar thread index.
- **vtd[3]**: **vtd[0 | 1 | 2]** is the x|y|z element of vector thread index.
- **mofst[3]**: **mofst[0 | 1 | 2]** is the x|y|z offset cell index of cell-block.

```
std = omp_get_thread_num();
vtd[0] = std/(vthrd[1]*vthrd[2]);
vtd[1] = (std/vthrd[2])%vthrd[1];
vtd[2] = std%vthrd[2];
for (a=0; a<3; a++)
   mofst[a] = vtd[a]*thbk[a];
```

Call `omp_get_thread_num()` within an OpenMP parallel block.
Threads Processing of Cell Blocks

- Start with the MPI parallel MD program, `pmd.c`
- Within each MPI process, parallelize the outer loops over central linked-list cells, `mc[]`, in the force computation function, `compute_accel()`, using OpenMP threads
- If each thread needs separate copy of a variable (e.g., loop index `mc[]`), declare it as `private` in the OpenMP parallel block

```c
#pragma omp parallel private(mc,...)
{
    ...
    for (mc[0]=mofst[0]+1; mc[0]<=mofst[0]+thbk[0]; (mc[0])++)
                {
                }
    ...
}
```
Avoiding Critical Sections

• A cell block should be at least 3-cells wide, so that no two threads are accessing the same atoms simultaneously
• Remove the critical section
  
  ```c
  if (bintra) lpe += vVal; else lpe += 0.5*vVal;
  ```

  by defining an array, `lpe[nthrd]`, where each array element stores the partial sum of the potential energy by a thread
Running HMD at HPC

1. Interactively submit a PBS job, & wait until you are allocated nodes. 
   (Note that you will be automatically logged in to one of the allocated 
   nodes.)

   hpc-master(18): qsub -I -l nodes=2:ppn=4 -l walltime=00:29:00
   qsub: waiting for job 1932791.hpc-pbs.usc.edu to start
   qsub: job 1932791.hpc-pbs.usc.edu ready

   Begin PBS Prologue Tue Oct 24 12:57:44 PDT 2006
   Job ID:            1932791.hpc-pbs.usc.edu
   Username:          anakano
   Group:             m-csci
   Name:              STDIN
   Queue:             quick
   Shared Access:     no
   Nodes:             hpc0097 hpc0104
   PVFS:              /scratch (128G)
   TMPDIR:            /tmp/1932791.hpc-pbs.usc.edu
   End PBS Prologue Tue Oct 24 12:57:47 PDT 2006

   [anakano@hpc0104 ~]$
Running HMD at HPC

2. Type the following sequence of commands. (In this example, hpc/cs596 is my working directory, where the executable hmd is located.)

[anakano@hpc0104 ~]$ bash
bash-2.05b$ source /usr/usc/mpich/default/gm-intel/setup.sh
bash-2.05b$ OMP_NUM_THREADS=4
bash-2.05b$ export OMP_NUM_THREADS
bash-2.05b$ cd hpc/cs596
bash-2.05b$ cp $PBS_NODEFILE nodefile

3. Edit nodefile, which originally consisted of 8 lines, to delete 6 lines.

(Original nodefile)  (Edited nodefile)

hpc0104                 hpc0104
hpc0104
hpc0104
hpc0104
hpc0097
hpc0097
hpc0097
hpc0097
hpc0097
hpc0097

4. Submit a two-process MPI program (named hmd); each of the MPI process will spawn 4 OpenMP threads.

bash-2.05b$ mpirun -np 2 -machinefile nodefile ./hmd
Running HMD at HPC

5. While the job is running, you can open another window & log in to both the nodes to check that both processors on each node are busy.

hpc-master(2): `rlogin hpc0104`

[anakano@hpc104 ~]$ `top`

13:00:36 up 93 days, 14:03, 1 user, load average: 1.41, 0.37, 0.23
70 processes: 65 sleeping, 5 running, 0 zombie, 0 stopped

CPU states: cpu user nice system irq softirq iowait idle

   total  86.8%  0.0%  0.0%  0.0%   0.0%  0.0%  13.1%
   cpu00  89.1%  0.0%  0.0%  0.0%   0.0%  0.0%  10.8%
   cpu01  88.1%  0.0%  0.0%  0.0%   0.0%  0.0%  11.8%
   cpu02  82.1%  0.0%  0.0%  0.0%   0.0%  0.0%  17.8%
   cpu03  88.1%  0.0%  0.0%  0.0%   0.0%  0.0%  11.8%

Mem: 3976844k av, 1747492k used, 2229352k free, 0k shrd, 155128k buff
1411876k active, 92572k inactive
Swap: 1052248k av, 6792k used, 1045456k free

PID USER     PRI  NI SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
24918 anakano 25   0 45436  44M 34200 R  23.7  1.1   0:35   3 hmd
24941 anakano 25   0 45436  44M 34200 R  22.2  1.1   0:33   0 hmd
24940 anakano 25   0 45436  44M 34200 R  22.0  1.1   0:30   1 hmd
24942 anakano 25   0 45436  44M 34200 R  18.8  1.1   0:32   2 hmd
  1 root      15   0  296  276  224 S  0.0  0.0   1:04   2 init
...