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

[Diagram showing the process of MPI and 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 $step$.
- Within each MPI process, $nthreads$ OpenMP threads perform part of the sum as in $omp\_pi.c$. 

![Diagram showing integration process with MPI and OpenMP](image.svg)
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-login2.usc.edu**
  source /usr/usc/mpich/default/mx-intel/setup.csh
  mpicc -o hpi hpi.c -openmp

• **PBS script**
  ```bash
  #!/bin/bash
  #PBS -l nodes=2:ppn=1,arch=x86_64
  #PBS -l walltime=00:00:59
  #PBS -o hpi.out
  #PBS -j oe
  #PBS -N hpi
  OMP_NUM_THREADS=2
  export OMP_NUM_THREADS
  WORK_HOME=/auto/rcf-proj2/an/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
  ```
Hybrid MPI+OpenMP Parallel MD

OpenMP threads handle blocks of linked-list cells in each MPI process (= spatial-decomposition subsystem)
**Linked-List Cell Block**

**Variables**

- `vthrd[0|1|2]` = # of OpenMP threads per MPI process in the x|y|z direction.
- `thbk[3]`: `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

- \( \text{std} = \) scalar thread index.
- \( \text{vtd}[3]: \text{vtd}[0|1|2] \) is the x|y|z element of vector thread index.
- \( \text{mofst}[3]: \text{mofst}[0|1|2] \) is the x|y|z offset cell index of cell-block.

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

Call \texttt{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 (1)

- Remove the critical section
  
  ```c
  if (bintra) lpe += vVal; else lpe += 0.5*vVal;
  ```

  by defining an array, `lpe_td[nthrd]`, where each array element stores the partial sum of the potential energy by a thread.
Avoiding Critical Sections (2)

- To avoid multiple threads to access an identical force array element, stop using the Newton’s third law:

```c
int bintra;
...
if (i<j && rr<rrCut) {
    ...
    if (bintra) lpe += vVal; else lpe_td[std] += 0.5*vVal;
    for (a=0; a<3; a++) {
        f = fcVal*dr[a];
        ra[i][a] += f;
        if (bintra) ra[j][a] -= f;
    }
}
```
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.)

```
$ qsub -I -l nodes=2:ppn=4,arch=x86_64 -l walltime=00:29:00
qsub: waiting for job 4685915.hpc-pbs.usc.edu to start
qsub: job 4685915.hpc-pbs.usc.edu ready

Begin PBS Prologue Wed Oct 29 09:05:16 PDT 2008
Job ID: 4685915.hpc-pbs.usc.edu
Username: anakano
Group: m-csci
Name: STDIN
Queue: quick
Shared Access: no
Nodes: hpc2051 hpc2052
PVFS: /scratch (126G)
TMPDIR: /tmp/4685915.hpc-pbs.usc.edu
End PBS Prologue Wed Oct 29 09:05:21 PDT 2008
```

[anakano@hpc2052 ~]$ bash
Running HMD at HPC (2)

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

   [anakano@hpc2052 ~]$ bash
   bash-3.00$ source /usr/usc/mpich/default/mx-intel/setup.sh
   bash-3.00$ OMP_NUM_THREADS=4
   bash-3.00$ export OMP_NUM_THREADS
   bash-3.00$ cd hpc/cs596
   bash-3.00$ cp $PBS_NODEFILE nodefile

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

   (Original nodefile)                      (Edited nodefile)
   hpc2052
   hpc2052
   hpc2052
   hpc2052
   hpc2052
   hpc2051
   hpc2051
   hpc2051
   hpc2051

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

   bash-3.00$ mpirun -np 2 -machinefile nodefile ./hmd
5. While the job is running, you can open another window & log in to both the nodes to check that all processors on each node are busy. Type ‘H’ to show individual threads.

```
[anakano@hpc-login2 ~]$ ssh hpc2052
[anakano@hpc2052 ~]$ top (then type H)
```

```
top - 07:48:59 up 11 days, 14:51, 1 user, load average: 3.22, 1.15, 0.54
Tasks: 150 total, 5 running, 145 sleeping, 0 stopped, 0 zombie
Cpu(s): 45.6% us, 0.0% sy, 0.0% ni, 54.4% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 16418456k total, 359736k used, 16058720k free, 96288k buffers
Swap: 1052216k total, 224k used, 1051992k free, 109684k cached

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>16070</td>
<td>anakano</td>
<td>21</td>
<td>0</td>
<td>78792</td>
<td>7172</td>
<td>1576</td>
<td>R</td>
<td>99.9</td>
<td>0.0</td>
<td>1:42.02</td>
<td>hmd</td>
</tr>
<tr>
<td>16079</td>
<td>anakano</td>
<td>16</td>
<td>0</td>
<td>78792</td>
<td>7172</td>
<td>1576</td>
<td>R</td>
<td>93.9</td>
<td>0.0</td>
<td>1:36.13</td>
<td>hmd</td>
</tr>
<tr>
<td>16077</td>
<td>anakano</td>
<td>16</td>
<td>0</td>
<td>78792</td>
<td>7172</td>
<td>1576</td>
<td>R</td>
<td>85.6</td>
<td>0.0</td>
<td>1:27.83</td>
<td>hmd</td>
</tr>
<tr>
<td>16078</td>
<td>anakano</td>
<td>16</td>
<td>0</td>
<td>78792</td>
<td>7172</td>
<td>1576</td>
<td>R</td>
<td>85.2</td>
<td>0.0</td>
<td>1:27.31</td>
<td>hmd</td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>16</td>
<td>0</td>
<td>4772</td>
<td>520</td>
<td>432</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:01.46</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>RT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.41</td>
<td>migration/0</td>
</tr>
</tbody>
</table>
Instead of the interactive PBS job, you can also submit a batch job using the following script (the uniq command will eliminate duplicated lines in $PBS_NODEFILE and retain only one line per node).

```
#!/bin/bash
#PBS -l nodes=2:ppn=4,arch=x86_64
#PBS -l walltime=00:00:59
#PBS -o hmd.out
#PBS -j oe
#PBS -N hmd
OMP_NUM_THREADS=4
export OMP_NUM_THREADS
source /usr/usc/mpich/default/mx-intel/setup.sh
WORK_HOME=/home/rcf-proj2/an/anakano/hpc/cs596
cd $WORK_HOME
Cat $PBS_NODEFILE | uniq > nodefile
np=$(cat nodefile | wc -l)
mpirun -np $np -machinefile nodefile ./hmd
```

No need if `omp_set_num_threads(nthrd)` is called in `hmd.c`
Strong Scalability of Hybrid MD

Dual quadcore Xeon
1 MPI process
1-8 threads