Outline of Assignment 4, Part I

Deadlock-free communication

pmd.c

if even
  MPI_Send()
  MPI_Recv()
else if odd
  MPI_Recv()
  MPI_Send()
else (self)
  memory copy

pmd_irecv.c

MPI_Irecv()
MPI_Send()  // OK to send to self
MPI_Wait()

Where?

2 in atom_copy()
2 in atom_move()

4 code segments in total

Computation (ns)/communication (μs-ms) overlap

MPI_Irecv()

?  
MPI_Send()
MPI_Wait()

[discovery ~]$ ping hpc-transfer.usc.edu
time=0.074 ms
...

No ifs & buts! cleaner
Counter=0

while [ $counter -lt 3 ]; do
    echo "***** Asynchronous *****"
    mpirun -n $SLURM_NTASKS ./pmd_irecv
    echo "***** Synchronous *****"
    mpirun -n $SLURM_NTASKS ./pmd

    let counter+=1
done

See “Bash scripting tutorial for beginners”
https://linuxconfig.org/bash-scripting-tutorial-for-beginners
Runtime Fluctuation

- Due to (1) network interference & (2) shared access to computing nodes, measured runtimes will fluctuate.

- The latter could be avoided by exclusive access (#SBATCH --exclusive), but please do not use this since it will cause very low utilization of computing resources & slow down other users’ work.

<table>
<thead>
<tr>
<th></th>
<th>Asynchronous</th>
<th>Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU &amp; COMT</td>
<td>5.190056e-01</td>
<td>5.163595e-01</td>
</tr>
<tr>
<td></td>
<td>1.406116e-01</td>
<td>1.690525e-01</td>
</tr>
</tbody>
</table>

---

pmd_irecv.c

pmd.c
• Start interactive job on discovery & start a MPI program on one of the allocated computing nodes

[anakano@discovery cs596]$ salloc --nodes=4 --ntasks-per-node=4 -t 30
salloc: Nodes d05-[33-36] are ready for job
[anakano@d05-33 cs596]$ mpirun -n 16 ./pmd_irecv
...

• In another terminal, log in to another allocated node & type ‘top’ to see running processes

[anakano@discovery cs596]$ ssh d05-34
[anakano@d05-34 ~]$ top
top - 07:42:03 up 47 days, 18:34, 2 users, load average: 4.37, 3.33, 3.15
Tasks: 315 total, 8 running, 307 sleeping, 0 stopped, 0 zombie

4 instances (ranks) of pmd_irecv are running per node
Resource Usage (2)

- Type ‘1’ (toggle to show detailed core usage): two users (including myself) are not making full use of cores; let others utilize the unused resources by avoiding exclusive access

<table>
<thead>
<tr>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
<th>%Cpu</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 us, 0.0 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st</td>
<td>%Cpu0</td>
<td>100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu1</td>
<td>100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu2</td>
<td>0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu3</td>
<td>76.3 us, 23.0 sy, 0.0 ni, 0.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu4</td>
<td>100.0 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu5</td>
<td>99.7 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu6</td>
<td>99.7 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu7</td>
<td>0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu8</td>
<td>0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu9</td>
<td>0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu10</td>
<td>0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st</td>
<td>%Cpu11</td>
</tr>
</tbody>
</table>

17 out of 24 cores unused
Note on Assignment 4, Part II

• Hands-on experience in a common situation of adding new analysis functionality to an existing MPI simulation code, \textit{via} minimally invasive surgery of the code

• Note the header, pmd.h, in the homework package, csci596-as04, was set for Part I:
  
  \begin{verbatim}
  int vproc[3] = {2,2,4}, nproc = 16;
  \end{verbatim}

  The number of MPI ranks should match \texttt{nproc} in pmd.h:
  
  \begin{verbatim}
  mpirun -n 16 ./pmd (also ./pmd_irecv)
  \end{verbatim}

• Due to ‘shadow’ analysis ranks, the total number of ranks to be spawned by \texttt{mpirun} in Part II should instead be twice the number of spatial subsystems, \texttt{nproc}, in pmd_split.c:

  In pmd_split.h:
  
  \begin{verbatim}
  int vproc[3] = {2,2,2}, nproc = 8;
  \end{verbatim}

  Run:
  
  \begin{verbatim}
  #SBATCH --nodes=2
  #SBATCH --ntasks-per-node=8
  mpirun -n $SLURM_NTASKS ./pmd_split // $SLURM_NTASKS = 16
  \end{verbatim}
Message Composition

- Multidimensional arrays are sent as one-dimensional arrays

```c
double rv[NMAX][3];
doubledbuf[NDBUF];
```

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbuf</td>
<td>rv[0][0]</td>
<td>rv[0][1]</td>
<td>rv[0][2]</td>
<td>rv[1][0]</td>
<td>rv[1][1]</td>
<td>rv[1][2]</td>
</tr>
</tbody>
</table>
```