Combining OpenMP and MPI

Timothy H. Kaiser,Ph.D..
tkaiser@mines.edu

Overview

- Discuss why we combine MPI and OpenMP
- Show how to compile and link hybrid programs
  - Intel Compiler
  - Portland Group Compiler
- Run Scripts
- Challenge: What works for Stommel code
  - 1 node
  - 2 nodes

Machine “might” drive program design

- Valid methodology for hybrid machines
- For example assume a machine:
  - 268 Nodes
  - Each node has 8 cores or processors
- We can have (per node)
  - 1 MPI process and 8 OpenMP threads
  - 2 MPI processes and 4 OpenMP threads
  - 4 MPI processes and 2 OpenMP threads

Why Combine OpenMP and MPI

- OpenMP might not require copies of data structures
- Can have some interesting designs that overlap computation and communication
- Overcome the limits of small processor counts on SMP machines

Compilers

- Intel
  - Fortran : ifort,
  - Fortran with MPI: mpif77, mpif90
  - C/C++ :icc
  - C/C++ with MPI: mpcc, mpCC
  - Option to support OpenMP
    - -openmp

Compilers

- Portland Group
  - Fortran : pgf77, pgf90
  - Fortran with MPI: mpif77, mpif90
  - C/C++ :pgcc
  - C/C++ with MPI: mpcc, mpCC
  - Option to support OpenMP
    - -mp
  - pgifortref.pdf has good examples

Run Scripts

```
#!/bin/bash -x
#SBATCH --job-name="hybrid"
#comment = "glorified hello world"
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --ntasks=16
#SBATCH --exclusive
#SBATCH --export=ALL
#SBATCH --time=00:05:00

# Go to the directoy from which our job was launched
cd $SLURM_SUBMIT_DIR

# Run the job.

EXEC=/opt/utility/phostname


export OMP_NUM_THREADS=2


srun $EXEC -1 > output.$SLURM_JOBID

# You can also use the following format to set
# --nodes             - # of nodes to use
# --ntasks-per-node  - ntasks = nodes*ntasks-per-node
# --ntasks            - total number of MPI tasks
#srun --nodes=$NODES --ntasks=$TASKS --ntasks-per-node=$TPN $EXE > > output.$SLURM_JOBID

export OMP_NUM_THREADS=4
srun --nodes=2 --ntasks-per-node=2 --ntasks=4 $EXEC -2 >> output.$SLURM_JOBID
```

Hybrid Hello World

```
program hybrid
    implicit none
    include 'mpif.h'
    integer numnodes,myid,my_root,ierr
    character (len=MPI_MAX_PROCESSOR_NAME):: myname
    integer mylen
    integer OMP_GET_MAX_THREADS,OMP_GET_THREAD_NUM
    call MPI_INIT( ierr )
    call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
    call MPI_COMM_SIZE( MPI_COMM_WORLD, numnodes, ierr )
    call MPI_Get_processor_name(myname,mylen,ierr)
!$OMP PARALLEL
!$OMP CRITICAL
  write(unit=*,fmt="(i4,a,a)",advance="no")myid," running on ",trim(myname)
  write(unit=*,fmt="(a,i2,a,i2)")" thread= ",OMP_GET_THREAD_NUM()," of ",OMP_GET_MAX_THREADS()
!$OMP END CRITICAL
!$OMP END PARALLEL
    call MPI_FINALIZE(ierr)
end program
```

```
mpif90 -openmp short.f90 -o short
```

2 nodes   1MPI task/node 4 threads

```
match shortlist oneprogram 1 > applist
export OMP_NUM_THREADS=4
```

```
0 running on compute-2-25.local thread=  0 of  4
0 running on compute-2-25.local thread=  1 of  4
0 running on compute-2-25.local thread=  2 of  4
0 running on compute-2-25.local thread=  3 of  4

1 running on compute-3-14.local thread=  0 of  4
1 running on compute-3-14.local thread=  1 of  4
1 running on compute-3-14.local thread=  2 of  4
1 running on compute-3-14.local thread=  3 of  4
```

2 nodes 2 MPI task/node 4 threads

```
match shortlist oneprogram 2 > applist
export OMP_NUM_THREADS=4
```

```
0 running on compute-2-25.local thread=  0 of  4
0 running on compute-2-25.local thread=  1 of  4
0 running on compute-2-25.local thread=  2 of  4
0 running on compute-2-25.local thread=  3 of  4
1 running on compute-2-25.local thread=  0 of  4
1 running on compute-2-25.local thread=  1 of  4
1 running on compute-2-25.local thread=  2 of  4
1 running on compute-2-25.local thread=  3 of  4

2 running on compute-3-14.local thread=  0 of  4
2 running on compute-3-14.local thread=  1 of  4
2 running on compute-3-14.local thread=  2 of  4
2 running on compute-3-14.local thread=  3 of  4
3 running on compute-3-14.local thread=  0 of  4
3 running on compute-3-14.local thread=  1 of  4
3 running on compute-3-14.local thread=  2 of  4
3 running on compute-3-14.local thread=  3 of  4
```

2 nodes 1 MPI task/node 8 threads

```
match shortlist oneprogram 1 > applist
export OMP_NUM_THREADS=8
```

```
0 running on compute-2-25.local thread=  0 of  8
0 running on compute-2-25.local thread=  1 of  8
0 running on compute-2-25.local thread=  2 of  8
0 running on compute-2-25.local thread=  3 of  8
0 running on compute-2-25.local thread=  4 of  8
0 running on compute-2-25.local thread=  5 of  8
0 running on compute-2-25.local thread=  6 of  8
0 running on compute-2-25.local thread=  7 of  8

1 running on compute-3-14.local thread=  0 of  8
1 running on compute-3-14.local thread=  1 of  8
1 running on compute-3-14.local thread=  2 of  8
1 running on compute-3-14.local thread=  3 of  8
1 running on compute-3-14.local thread=  4 of  8
1 running on compute-3-14.local thread=  5 of  8
1 running on compute-3-14.local thread=  6 of  8
1 running on compute-3-14.local thread=  7 of  8
```

2 nodes 4 MPI task/node 2 threads

```
match shortlist oneprogram 4 > applist
export OMP_NUM_THREADS=2
```

```
0 running on compute-2-25.local thread=  0 of  2
0 running on compute-2-25.local thread=  1 of  2
1 running on compute-2-25.local thread=  0 of  2
1 running on compute-2-25.local thread=  1 of  2
2 running on compute-2-25.local thread=  0 of  2
2 running on compute-2-25.local thread=  1 of  2
3 running on compute-2-25.local thread=  0 of  2
3 running on compute-2-25.local thread=  1 of  2

4 running on compute-3-14.local thread=  0 of  2
4 running on compute-3-14.local thread=  1 of  2
5 running on compute-3-14.local thread=  0 of  2
5 running on compute-3-14.local thread=  1 of  2
6 running on compute-3-14.local thread=  0 of  2
6 running on compute-3-14.local thread=  1 of  2
7 running on compute-3-14.local thread=  0 of  2
7 running on compute-3-14.local thread=  1 of  2
```

Test Program on CSM machines

```
/opt/utility/phostname -h
phostname arguments:
          -h : Print this help message

no arguments : Print a list of the nodes on which the command is run.

 -f or -1    : Same as no argument but print MPI task id and Thread id
               If run with OpenMP threading enabled OMP_NUM_THREADS > 1
               there will be a line per MPI task and Thread.

 -F or -2    : Add columns to tell first MPI task on a node and and the
               numbering of tasks on a node. (Hint: pipe this output in
               to sort -r

 -a          : Print a listing of the environmental variables passed to
               MPI task. (Hint: use the -l option with SLURM to prepend MPI
               task #.)
```

Run Scripts

```
#!/bin/bash -x
#SBATCH --job-name="hybrid"
#comment = "glorified hello world"
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --ntasks=16
#SBATCH --exclusive
#SBATCH --export=ALL
#SBATCH --time=00:05:00

# Go to the directoy from which our job was launched
cd $SLURM_SUBMIT_DIR

# Run the job.

EXEC=/opt/utility/phostname


export OMP_NUM_THREADS=2


srun $EXEC -1 > output.$SLURM_JOBID

# You can also use the following format to set
# --nodes            - # of nodes to use
# --ntasks-per-node  - ntasks = nodes*ntasks-per-node
# --ntasks           - total number of MPI tasks
#srun --nodes=$NODES --ntasks=$TASKS --ntasks-per-node=$TPN $EXE > > output.$SLURM_JOBID

export OMP_NUM_THREADS=4
srun --nodes=2 --ntasks-per-node=2 --ntasks=4 $EXEC -2 >> output.$SLURM_JOBID
```

Sorted Output

```
export OMP_NUM_THREADS=2
/opt/utility/phostname -1
compute028 0000 0000
compute028 0000 0001
compute028 0001 0000
compute028 0001 0001
compute028 0002 0000
compute028 0002 0001
compute028 0003 0000
compute028 0003 0001
compute028 0004 0000
compute028 0004 0001
compute028 0005 0000
compute028 0005 0001
compute028 0006 0000
compute028 0006 0001
compute028 0007 0000
compute028 0007 0001
compute029 0008 0000
compute029 0008 0001
compute029 0009 0000
compute029 0009 0001
compute029 0010 0000
compute029 0010 0001
compute029 0011 0000
compute029 0011 0001
compute029 0012 0000
compute029 0012 0001
compute029 0013 0000
compute029 0013 0001
compute029 0014 0000
compute029 0014 0001
compute029 0015 0000
compute029 0015 0001
```

```
export OMP_NUM_THREADS=4
srun --nodes=2 --ntasks-per-node=2 -ntasks=4 /opt/utility/phostname -2
task    thread             node name  first task    # on node
0000      0000            compute028        0000         0000
0000      0001            compute028        0000         0000
0000      0002            compute028        0000         0000
0000      0003            compute028        0000         0000
0001      0000            compute028        0000         0001
0001      0001            compute028        0000         0001
0001      0002            compute028        0000         0001
0001      0003            compute028        0000         0001
0002      0000            compute029        0002         0000
0002      0001            compute029        0002         0000
0002      0002            compute029        0002         0000
0002      0003            compute029        0002         0000
0003      0000            compute029        0002         0001
0003      0001            compute029        0002         0001
0003      0002            compute029        0002         0001
0003      0003            compute029        0002         0001
```

Challenges

- Modify one of the Stommel program versions to be hybrid

- Run on one node
  - 8 MPI
  - 4 MPI x 2 OpenMP
  - 2 MPI x 8 OpenMP
  - 8 OpenMP

- Run on two nodes
  - 16 MPI
  - 4 MPI x 4 OpenMP
  - 2 MPI x 8 OpenMP
  - 8 MPI x 2 OpenMP

Run times

StomOmpf_02a

pure OpenMP x 8	22.81
Combined 8 MPI x 1 OpenMP	3.34
Combined 2 MPI x 4 OpenMP	37.85
Combined 4 MPI x 2 OpenMP	23.27

StomOmpf_02d

pure OpenMP x 8	3.54
Combined 8 MPI x 1 OpenMP	3.54
Combined 2 MPI x 4 OpenMP	4.5
Combined 4 MPI x 2 OpenMP	4.38

serial	18.79
pure MPI x 8	3.36

A new way to do hybrid

- MPC - a thread library supporting multiple parallel programming models
  - POSIX thread
  - Intel TBB version 2.1 (Thread building blocks)
  - MPI version 1.3
  - OpenMP version 2.5
  - Hybrid MPI/OpenMP
- Unified Parallel Framework for HPC
- All done with user level threads
- Adds built in checkpointing
- French Atomic Energy Commission
  - http://www-hpc.cea.fr/en/red/docs/MPC-V2.htm
  - Open Source