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

- IBM
  - Fortram
    - bgxlf_r, bgxlf90, bgxlf90_r, bgxlf95_r, bgxlf2003_r,bgxlf2008_r
  - C
    - bgxlc_r, bgxlc++_r, bgxlC_r, bgcc_r, bgc89_r,bgc99_r
  - MPI
    - mpixlc_r, mpixlC_r, mpixlcxx_r, mpixlf2003_r, mpixlf2008_r, mpixlf77_r, mpixlf90_r, mpixlf95_r
  - -qsmp=mop

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
#PBS -l nodes=2:ppn=8
#PBS -l naccesspolicy=singlejob
#PBS -l walltime=00:20:00
#PBS -N testIO
#PBS -o outx8.$PBS_JOBID.pbs
#PBS -e errx8.$PBS_JOBID.pbs
#PBS -r n
#PBS -V
##PBS -q ALLNODES
#-----------------------------------------------------
cd $PBS_O_WORKDIR

#save a nicely sorted lisst of nodes
sort -u $PBS_NODEFILE > shortlist

#generate our program mapping to run "short"
echo `pwd`/short > oneprogram

#recall that “match” takes our list of nodes and program name
match shortlist oneprogram  1 > applist

#set the number of threads to use
export OMP_NUM_THREADS=4

#run our hybrid program
mpiexec  -app applist
```

We use the same method that we used for MPMD programs to map a single program to our nodes

Our applist

```
[tkaiser@ra ~/hybrid]$ cat applist
-host  compute-2-25.local  -np  1  /lustre/home/tkaiser/hybrid/short
-host  compute-3-14.local  -np  1  /lustre/home/tkaiser/hybrid/short
[tkaiser@ra ~/hybrid]$
```

Run Scripts (Slurm)

```
#!/bin/bash -x
#SBATCH --job-name="hybrid"
#comment= "glorified hello world"
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --ntasks=8
#SBATCH --exclusive
#SBATCH --export=ALL
#SBATCH --time=10:00:00
#SBATCH -o ascript_out.%j

## Go to the directoy from which our job was launched
cd $SLURM_SUBMIT_DIR
JOBID=`echo $SLURM_JOBID`
mkdir ascript_$JOBID
cd ascript_$JOBID

echo "trying runjob"
export OMP_NUM_THREADS=4
printenv OMP_NUM_THREADS > env_$JOBID
srun --label  $SLURM_SUBMIT_DIR/docol.exe > output
echo "got to the bottom"
```

Cool short example program

See: http://hpc.mines.edu/bgq/mc2/

```
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

mpixlf90_r -qsmp=omp short.f90 -o short

2 nodes 1 MPI 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
```

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