NCCS Brown Bag Series

Chongxun (Doris) Pan
doris.pan@nasa.gov
July 30, 2013
Previous MIC talks (Part 1 and Part 2)

- What is MIC?
- MIC Programming Considerations
- Offload vs. Native
- Intel MPI on MIC
  - MPI+Offload Model
  - Native Model
  - Symmetric Model
- User Environment Variables
- General Performance Guidelines

Slides available at
http://www.nccs.nasa.gov/list_brown_bags.html
Today’s Agenda

- WRF 3.5 overview
- Build NETCDF
- Build WRF for both Xeon and Xeon Phi
- Run WRF in native and symmetric models
- Performance Comparison

Acknowledgements

John Michalakes from NREL
http://www.mmm.ucar.edu/wrf/WG2/mic.htm
Indraneil Gokhale from Intel
Hamid Oloso
New software features in WRF3.5

- ARW model is supported on the Intel Xeon Phi
- Modify shared memory tiling algorithm
- Thread the packing and unpacking of MPI messages (will improve performance of all dm+sm runs)
- Special optimized version of WSM5 component
  - More thread parallelism and vectorization
  - Improved memory latency
- A couple of performance mods that target on SNB nodes, using –xavx compiler option etc.

http://www.mmm.ucar.edu/wrf/users/wrfv3.5/updates-3.5.html
To run MPI applications on the Phi on Discover

- Request access to the “native” queue
- Have to use Intel 13 compiler and Intel MPI 4.1, e.g.
  - module load comp/intel-13.1.3.192
  - module load mpi/impi-4.1.0.024
- Set the Intel MPI environment
  - source /opt/intel/impi/4.1.0.024/intel64/bin/mpivars.sh
- Only one user environment setup required for $PATH and $LD_LIBRARY_PATH, serves both host and Phi
  - setenv PATH ${PATH}:/opt/intel/mic/bin
- Mounted file system
  - Refer to /home/cpan2/.login_scu8
  - /mnt/discover/home/$USER accessible from both host and Phi
#!/bin/sh

set -x
./usr/share/modules/init/sh
module purge
module load comp/intel-13.1.3.192

ROOTDIR=`pwd`

cd $ROOTDIR
export CXX="icpc"
export CC="icc"
export FC="ifort"
export F77="ifort"

./configure --prefix=$ROOTDIR/Intel64 --disable-cxx
make
make check
make install
#!/bin/csh -fx

source /usr/share/modules/init/csh
source ~cpan2/.login_scu8
module purge
module load comp/intel-13.1.3.192

set ROOTDIR=`pwd`
cd $ROOTDIR/
setenv CPPFLAGS "-DpgiFortran"
setenv CXX "icpc"
setenv CC "icc"
setenv F77 "ifort"

./configure NM=nm --prefix=$ROOTDIR/XeonPhi --disable-cxx --host=x86_64-k1om-linux --build=x86_64-unknown-linux

make CFLAGS=-mmic FCLAGS=-mmic LDFLAGS=-mmic
make install
# make check (Note that “make check” will throw errors because the executables are built on the host but can only run on the Phi)
Build WRF3.5 optimized for the SNB

```bash
#!/usr/bin/csh

setenv wrfsorc `pwd`
cd ${wrfsorc}

setenv NETCDF /discover/nobackup/cpan2/lib/NETCDF/Intel64

# clean -a removes all including configure.wrf file
./clean –a

./configure
# option 25, default everything else
# remove -DINTEL_ALIGN64 from the ARCH_LOCAL flags in configure.wrf

setenv J "-j 8"
# parallel build. Make sure to request a compute node for parallel build

compile em_real |& tee make.log
```
#! /bin/bash

source ~cpan2/.login_scu8.bash

wrfsrc=`pwd`
cd ${wrfsrc}

export NETCDF=/mnt/discover/home/cpan2/lib/netcdf-3.6.2

# clean -a removes all including configure.wrf file
./clean –a

./configure
# option 21, default everything else
# change “mpicc” to “mpiicc” in configure.wrf

export J="-j 8"
# parallel build. Make sure to request a compute node for parallel build

compile em_real | tee make.log
WRF is a hybrid MPI/OpenMP model. The decomposition for WRF contains two levels:

**Patches:** sections of entire model domain over distributed memory
- \( \text{nproc}_x/\text{nproc}_y \) is -1 in namelist by default, meaning automatic decomposition

**Tiles:** sections of a patch over shared memory
- 1 tile if not compiled with –openmp or only 1 thread is available
- Can be specified in namelist as “numtiles” or “numtiles\_x” and “numtiles\_y”
- Or via env variable WRF\_NUM\_TILES or WRF\_NUM\_TILES\_X and WRF\_NUM\_TILES\_Y
Run WRF on Discover

- We use a 12KM Conus benchmark test case. See details at [http://www.mmm.ucar.edu/wrf/WG2/benchv3/#_Toc212961288](http://www.mmm.ucar.edu/wrf/WG2/benchv3/#_Toc212961288)
- 48-hour, 12KM resolution over Continental US domain, time step 72 seconds. Test case run is 3 hours (149 time steps)
- The test cases use ~7GB of memory

- We demonstrate running the test case:
  1. On SNB nodes only
  2. In Native mode: On Phi coprocessors only
  3. In Symmetric mode: On both SNB and Phi
Example script to run WRF on SNB

```bash
#!/bin/csh -fx

module purge
module load comp/intel-13.1.3.192
module load mpi/impi-4.1.0.024

setenv I_MPI_PIN_MODE mpd
setenv I_MPI_PIN_DOMAIN auto
setenv OMP_NUM_THREADS 4
setenv KMP_STACKSIZE 64M
setenv KMP_AFFINITY "scatter,granularity=thread"
setenv KMP_BLOCKTIME infinite
setenv KMP_LIBRARY turnaround
setenv WRF_NUM_TILES 32

cd $NOBACKUP/WRFV3.5-SNB/test/em_real
rm rsl.*
mpiexec.hydra -np 4 ./wrf.exe >& out.snb
```

KMP_LIBRARY=turnaround
KMP_BLOCKTIME=infinite

keep active all of the threads involved in the parallel computation in order to minimize the execution time of a single job. Best suited for dedicated computing resources (PBS jobs).
Process Pinning (binding) – Pin a particular MPI process to a CPU and avoid undesired process migration

I_MPI_PIN_DOMAIN =<pinmode>
<pinmode> =
  mpd       mpd daemon pins processes before launching MPI processes. **Best performance for both SNB and MIC on Discover**
  pm        Hydra launcher pins MPI processes before launching MPI processes.
  lib       MPI library pins processes but this does not offer co-location of CPU and memory
Intel MPI Support for MPI/OpenMP hybrid applications extends to MIC

Additional env variable, \texttt{I\_MPI\_PIN\_DOMAIN}, to control process pinning for hybrid applications

- Mapping rule -- 1 MPI process per domain
- Inside the domain, pin OpenMP threads with \texttt{KMP\_AFFINITY}

\texttt{I\_MPI\_PIN\_DOMAIN} =<size>[::<layout>]

<size> =
- \texttt{omp} Adjust to OMP\_NUM\_THREADS
- \texttt{auto} \#total CPUs / \#MPI procs
- <n> Number

<layout> =
- \texttt{platform} According to BIOS numbering
- \texttt{compact} Close to each other -- Default
- \texttt{scatter} Far away from each other
Thread Pinning

Thread pinning within a MPI domain is controlled by **KMP_AFFINITY**

**KMP_AFFINITY** =<type>[,<modifier>]

<type> =
- compact  Pack threads close to each other
- scatter   Round-robin threads to cores
- balanced  Keep OMP thread ids consecutive (MIC only)
- explicit  Use the proclist modifier to pin threads
- none      Does not pin threads (Default)

<modifier> = granularity
- core      Bound to core (Default)
- fine/thread Bound to thread
Example script to run WRF in Native Mode – Launch the script from the Host

```bash
#!/bin/bash
source ~cpan2/.login_scu8.bash

export I_MPI_PIN_MODE=mpd
export I_MPI_PIN_DOMAIN=auto
export OMP_NUM_THREADS=180
export KMP_STACKSIZE=62M
export KMP_AFFINITY="balanced,granularity=thread"
export KMP_PLACE_THREADS=60C,3T
export KMP_BLOCKTIME=infinite
export KMP_LIBRARY=turnaround
export WRF_NUM_TILES_X=3
export WRF_NUM_TILES_Y=60
export I_MPI_MIC=1
export I_MPI_FABRIC=shm:ofa
export I_MPI_DEBUG=5

cd /mnt/discover/home/cpan2/WRFV3.5-Phi/test/em_real
rm rsl.*

$ cat ./wrf.sh
#!/bin/sh
ulimit -s unlimited
*

mpiexec.hydra -f hostfile -np 1 ./wrf.sh ./wrf.exe | tee out.phi
```

- Works also for multiple Phi coprocessors
- Make sure to set stacksize unlimited for the Phi. Otherwise the code will seg fault when memory exceeds ~5GB.

$ cat ./wrf.sh
#!/bin/sh
ulimit -s unlimited
*

```
Example script to run WRF in Native Mode – Launch the script from the Phi

```
#!/bin/sh
. /mnt/discover/home/cpan2/Native-Sample/.profile_phi
ulimit -s unlimited

export I_MPI_PIN_MODE=mpd
export I_MPI_PIN_DOMAIN=auto
export OMP_NUM_THREADS=180
export KMP_STACKSIZE=62M
export KMP_AFFINITY="balanced,granularity=thread"
export KMP_PLACE_THREADS=60C,3T
export KMP_BLOCKTIME=infinite
export KMP_LIBRARY=turnaround
export WRF_NUM_TILES_X=3
export WRF_NUM_TILES_Y=60
export I_MPI_MIC=1
export I_MPI_FABRIC=shm:ofa
export I_MPI_DEBUG=5

cd /mnt/discover/home/cpan2/WRFV3.5-Phi/test/em_real
rm rsl.*

mpiexec.hydra -np 1 ./wrf.exe | tee out.phi
```

• Running applications on multiple Phi coprocessors is currently supported ONLY when the script is issued from the Host
Load Balancing with Symmetric Runs

**Situation**
- Host and MIC computation performance are different
- Host and MIC internal communication speed is different
- MPI in symmetric mode is like running on a heterogeneous cluster

**Solutions**
- **Approach 1:** Adapt MPI mapping of the hybrid code
  - Example: $m_1$ processes and $m_2$ threads per host, $n_1$ process and $n_2$ threads per MIC card
- **Approach 2:** Change code internal mapping of workload to MPI processes
  - Example: uneven split of calculation grid for MPI processes on host vs. MIC

**Analyze and improving MPI/thread load balance of application with Trace Analyzer and Collector (ITAC)**
Symmetric WRF Runs

Approach 1: Balance performance by tuning the # of MPI ranks and OMP threads on both the host and Phi.

Host
8 MPI ranks and 2 threads
Executable compiled for the Host, wrf.exe

Phi
7 MPI ranks and 34 threads
Executable compiled for the Phi, wrf.exe.mic

An example:
mpiexec.hydra –genv I_MPI_DEBUG 5 \
–genv I_MPI_PIN_MODE mpd \
-env OMP_NUM_THREADS 2 -env KMP_AFFINITY compact \ 
-host borg01w001 –n 8 ./wrf.exe \
: -env OMP_NUM_THREADS 34 -env KMP_AFFINITY balanced \
-host borg01w001-mic0 –n 7 /discover/home/cpan2/MIC/wrf.exe.mic
Example script to run WRF in Symmetric Mode

$ cat ./wrf.sh
#!/bin/sh
export RUNDIR=$1
shift
cd $RUNDIR
ulimit -s unlimited
source $RUNDIR/'hostname`.envvars
$
$ cat phi.envvars
#!/bin/sh
.
/discover/home/cpan2/.profile_phi
export OMP_NUM_THREADS=34
export KMP_LIBRARY=turnaround
export KMP_BLOCKTIME=infinite
export KMP_STACKSIZE=32M
export OMP_SCHEDULE=STATIC
export KMP_AFFINITY=balanced
$ cat sandy.envvars
#!/bin/sh
export OMP_NUM_THREADS=2
export KMP_LIBRARY=turnaround
export KMP_BLOCKTIME=infinite
export KMP_STACKSIZE=32M
export OMP_SCHEDULE=DYNAMIC

#!/bin/bash
source ~/.login_scu8.bash
ulimit -s unlimited
export I_MPI_MIC=1
export I_MPI_FABRIC=shm:ofa
export I_MPI_DEBUG=5
export I_MPI_PIN_MODE=mpd
export I_MPI_PIN_DOMAIN=auto

export RUNDIR="/mnt/discover/home/cpan2/WRFV3.5-Phi/test/em_real_symmetric"
cd $RUNDIR
rm rsl.*
sandy_name=`hostname`
mic_name="$sandy_name"-nic0"
rm -f $sandy_name.envvars $mic_name.envvars
ln -s sandy.envvars $sandy_name.envvars
ln -s phi.envvars $mic_name.envvars

mpirun.hydra -host $sandy_name -n 8 $RUNDIR/wrf.sh
$RUNDIR /discover/nobackup/cpan2/WRFV3.5-SNB/main/wrf.exe : -host $mic_name -n 7 $RUNDIR/wrf.sh $RUNDIR $RUNDIR/wrf.exe | tee combo.out
## Performance Comparison

<table>
<thead>
<tr>
<th>Conus 12KM Test Case</th>
<th>Configurations (nodes x MPI ranks x OMP threads)</th>
<th>Compute Time (Secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon SNB Only (16 cores per node)</td>
<td>1 x 4 x 4</td>
<td>106.1</td>
</tr>
<tr>
<td></td>
<td>1 x 8 x 2</td>
<td>107.1</td>
</tr>
<tr>
<td>Xeon Phi Only (60 cores per node)</td>
<td>1 x 1 x 180</td>
<td>122.2</td>
</tr>
<tr>
<td></td>
<td>1 x 1 x 240</td>
<td>127.0</td>
</tr>
<tr>
<td>Symmetric</td>
<td>1x(SNB+KNC) =1x(8x2 + 7 x 34)</td>
<td>92.0</td>
</tr>
<tr>
<td></td>
<td>1x(SNB+KNC) =1x(8x2 + 8 x 30)</td>
<td>97.1</td>
</tr>
</tbody>
</table>

For larger resolution WRF runs, the symmetric runs can yield ~50% of performance gain compared to SNB-only runs.
Thank You!

Lots of documentations and tutorial videos are offered by Intel

http://software.intel.com/mic-developer

More brownbag tutorials to come …

Intel MPI on MIC
Running WRF on MIC
Performance analysis with VTune Amplifier and Tracer Analyzer
Language Extensions for Offload
Maximize Vectorization
Performance tuning topics for MIC