Beacon Training: Using the Intel® Many Integrated Core (MIC) Architecture: Native Mode and Intel® MPI

Vincent Betro, Ph.D.
NICS—March 21-22, 2013
This material is based upon work supported by the National Science Foundation under Grant Number 1137907.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Native Mode

- All code runs on the MIC
- Use the compiler flag ‘-mmic’
- Transfer all files (binary, input data, compiled libraries) to the card before executing
- Use $TMPDIR as a staging area
  - Ex: cp myfile $TMPDIR/mic0 or $TMPDIR/mic1
- Libraries that use configure scripts need to be set up using the proper compiler flags
  - export CC="icc -mmic" OR "mpiicc -mmic"
  - export CXX="icpc -mmic" OR "mpiicc -mmic"
  - export F77="ifort -mmic" OR "mpiifort -mmic"
  - export F90="ifort -mmic" OR "mpiifort -mmic"
- If the configure script fails, try forcing cross compile by using ./configure -host=x86_64-k1om-linux
Native Mode – Configure scripts

• If the cross compiling option is unavailable, then the configure script needs to be fooled with a dummy flag
  - export CC="icc -DMMIC" OR "mpiicc -DMMIC"
  - export CXX="icpc -DMMIC" OR "mpiicc -DMMIC"
  - export F77="ifort -DMMIC" OR "mpiifort -DMMIC"
  - export F90="ifort -DMMIC" OR "mpiifort -DMMIC"

• Run configure again

• Change all instances of -DMMIC to -mmic, before running make
  - files=$(find ./ -name Makefile)
  - perl -p -i -e 's/-DMMIC/-mmic/g' $files
  - export CC="icc -mmic" OR "mpiicc -mmic"
  - export CXX="icpc -mmic" OR "mpiicc -mmic"
  - export F77="ifort -mmic" OR "mpiifort -mmic"
  - export F90="ifort -mmic" OR "mpiifort -mmic"
  - make
Native Mode Shared Libraries

• Unless provided by a module, all shared libraries need to be recompiled for native mode use

• 1. Compile the library source code
   oicc -mmic -c -fpic mylib.c
   oicpc -mmic -c -fpic mylib.cpp
   oifort -mmic -c -fpic mylib.f90

• 2. Use the –shared compiler flag to create the library from the object file
   oicc -mmic -shared -o libmylib.so mylib.o
   oicpc -mmic -shared -o libmylib.so mylib.o
   oifort -mmic -shared -o libmylib.so mylib.o

• 3. Compile and link the native application code with the native shared object
   oicc -mmic main.c libmylib.so
   oicpc -mmic main.cpp libmylib.so
   oifort -mmic main.f90 libmylib.so

• 4. Copy binary and library over to MIC before executing
   ocp a.out $TMPDIR/mic#
   ocp libmylib.so $TMPDIR/mic#/lib
Native Mode File Extension

• In some cases a native mode application will be used in conjunction with a host only application using MPI

• Since having the same filename for both applications can get confusing, it is recommended to append an extension to the native mode applications

  o icc -mmic -o application.MIC source.c
  o icpc -mmic -o application.MIC source.cpp
  o ifort -mmic -o application.MIC source.f90
Using Intel’s Math Kernel Library (MKL) in Native Mode

- Works the same as in Offload Mode, except that instead of the call being offloaded to the MIC, all actions take place on MIC.

- Note: ALL libraries under /opt/intel/composerxe_mic/mkl/ must be copied over to the MIC card!

- Common errors if all libraries are not copied:
  - MKL FATAL ERROR: Cannot load <mkl-thread-layer>
  - MKL FATAL ERROR: Cannot load ***.so
Native Mode Example

• Run the native mode example found in the Beacon quickstart guide

• Key points:
  o logging in to Beacon via ssh
  o compiling with the \(-\text{mmic}\) flag
  o requesting an interactive node with \texttt{qsub}
  o copying native mode program to the local SSD scratch space
  o using custom script \texttt{micssh} to connect to a MIC
  o navigating to the \texttt{$\text{TMPDIR}$} folder
  o running directly on the MIC
Using Intel MPI

• Intel has now provided official Intel MPI (impi) libraries, which allow the user to run:
  – in native mode from the host
  – in offload mode from the host
  – in heterogeneous mode (symmetric and asymmetric)
  – run directly on the MIC (not suggested)
Using Intel MPI: Host

Access to the Intel MPI tools and libraries on Beacon is managed through the "module" system. The intel-mpi module is loaded by default upon login.

Part of the Intel MPI environment is the "mpiicc" command. This command ensures that icc is invoked with the necessary options for MPI.

The command "mpiexec" can be used to launch the MPI program on the host (Xeon) node, even if there are multiple nodes requested:

```
mpiexec -n 2 -host beacon#/ ./mpi_hello
```

**Note:** the host option is unnecessary if there is only one node requested or if you wish to use the PBS_NODEFILE entries for each MPI rank, round-robin.
Using Intel MPI: MIC

In order to tell "mpiexec" to use the "micssh" command to access the MIC, we use the "micmpiexec" command to run the MPI program:

```
micmpiexec -n 2 -host beacon#-mic0 -wdir $TMPDIR
$TMPDIR/mpi_hello.MIC
```

**Note:** While it is not recommended, one may access the MIC through micssh and run the code using mpiexec.hydra –n np exe args

**Note:** If you run mpiexec from the MIC, the LD_LIBRARY_PATH does not follow through micssh, so one must add locations to the local LD_LIBRARY_PATH as seen by the MIC.
Using Intel MPI: MIC 2 MIC

• If one MIC card is not sufficient for your domain decomposition, you may use both MIC cards on the node or even multiple MIC cards on multiple nodes.

• If you need to use MICs on multiple nodes, you must request multiple nodes with qsub and check which ones they are with “cat $PBS_NODEFILE”

```
micmpiexec -n 2 -wdir $TMPDIR -host beacon#-mic0 $TMPDIR/mpi_hello.MIC : -n 2 -wdir $TMPDIR -host beacon#-mic1 $TMPDIR/mpi_hello.MIC
```
Using Intel MPI: Heterogeneous

- Depending on the workflow of the problem, it might behoove the user to use heterogeneous mode where MPI tasks are spread between the host and the MICs.

- This can be done in symmetric mode, where the same number of MPI tasks are allocated on both host and MIC.

- This can also be done in asymmetric mode, where the numbers of tasks on host and MIC differ.
Using Intel MPI: Heterogeneous

micmpiexec -n 2 -wdir $TMPDIR -host beacon#-mic0 $TMPDIR/mpi_hello.MIC : -n 2 -wdir $TMPDIR -host beacon#-mic1 $TMPDIR/mpi_hello.MIC : -n 2 -host beacon# ./mpi_hello

*Note: In this symmetric example, two MIC cards are being used. Technically, only one MIC card and one host are supported at this time, but you will find that multiple MICs often work fine with simple codes.*
Using Intel MPI: Heterogeneous

```bash
micmpiexec -n 2 -wdir $TMPDIR -host beacon#-mic0
$TMPDIR/mpi_hello.MIC : -n 2 -wdir $TMPDIR -host
beacon#-mic1 $TMPDIR/mpi_hello.MIC : -n 4 -host beacon#
./mpi_hello
```

Note: In this asymmetric example, two MIC cards are being used. Technically, only one MIC card and one host are supported at this time, but you will find that multiple MICs often work fine with simple codes.
Intel MPI

• The installed Intel MPI library implements the Message Passing Interface, version 2.2 (MPI-2.2) specifications

• 3 Programming models are supported
  – Co-processor only model
  – Symmetric model
  – MPI offload model

• Intel MPI compilers have an extra ‘i’ in their name: mpiicc, mpiicpc, mpiifort

• MPI applications should be launched from the host compute node using micmpiexec
Intel MPI with Native Mode Libraries

• Any native mode library, other than OpenMP MKL, and MPI, needs to be copied over to $TMPDIR/mic#/lib for all MICs or linked from lustre

• At this point, you need to augment your LD_LIBRARY_PATH by appending all libs not in $TMPDIR/lib.
Launching MPI Applications from Compute Node Host

• It is assumed that all Native MPI applications have been copied to $TMPDIR/mic# on all assigned compute nodes

• Since the host MPI application is typically located in a different directory than $TMPDIR, the working directory must be specified for each host

• The machines that will be run across can either be specified manually or through a machinefile
Launching an MPI Application with Manually Specified Hosts

• Launch an MPI application on mic0 of the current compute node with

  o micmpiexec -n 1 -wdir $TMPDIR -host beacon#-mic0 $TMPDIR/application.MIC

• Launch an MPI application on both mic0 and mic1 of the current compute node with

  o micmpiexec -n 1 -wdir $TMPDIR -host beacon#-mic0 $TMPDIR/application.MIC : -n 1 -wdir $TMPDIR -host beacon#-mic1 $TMPDIR/application.MIC

• Launch an MPI application on both MICs and the compute node with

  o micmpiexec -n 1 -wdir $TMPDIR -host beacon#-mic0 $TMPDIR/application.MIC : -n 1 -wdir $TMPDIR -host beacon#-mic1 $TMPDIR/application.MIC : -n 1 -host beacon# ./application
Launching an MPI Application with a Machine File

• The machine file needs to be of the form `<host>:`<number of ranks>

• Sample machine file named `hosts_file`:
  - beacon11:8
  - beacon12:8
  - beacon11-mic0:2
  - beacon11-mic1:2
  - beacon12-mic0:2
  - beacon12-mic1:2

• Launch the MPI application using
  - `micmpiexec -machinefile hosts_file -n 16 ./application : -n 8 -wdir $TMPDIR -genv LD_LIBRARY_PATH $TMPDIR/lib:/lib64:/lib $TMPDIR/application.MIC`
Launching an MPI Application with Process Pinning

- export I_MPI_PIN=1
- export I_MPI_PIN_PROCESSOR_LIST="0-6,8-14"
- export I_MPI_DEBUG=4 or 5

- Allows you to pin MPI processes to sockets on the host.
Now for some examples!
Contact

Vincent Betro
vbetro@tennessee.edu

NICS Support
help@nics.utk.edu