Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

MIC-Native and Offload-lab: Running simple C Programs in Native and Offload Mode

In this lab you run simple programs in native and offload mode.
We then go on to offload a matrix-matrix multiplication and perform a scaling analysis.

Appropriate Environment

Start 3 xterm windows:

- 1 xterm with a shell on the login node supermic.smuc.lrz.de
- 1 xterm with a shell on a compute node i01r13??? (submit a job and look at llq to figure out the hostname of the allocated compute node)
- 1 xterm with a shell on the associated MIC i01r13???-mic0

Attention:

- **Compile on supermic.smuc.lrz.de**
- **Run on Compute nodes i01r13c?? for Offload and MPI**
- **Run on MICs i01r13c??-mic0/1 for Native Mode**

Lab 1: Running MIC binaries natively

- Compile the program hello.c for MIC using
  icpc –mmic hello.c -o hello-mic

- Try to launch the program on the host.

- Copy the program to a MIC, e.g. i01r13c01-mic0 using scp.

- Login to the MIC and execute the program.

- Execute the program on the host using micnativeloadex. Look at the output of *micnativeloadex program -l.*

- Get information about the number of cores on a MIC by using the tools *micinfo*, *micinfo –listdevices, micsmc –a* on the host.

- Login to the MIC and get information about the cores, memory etc. by inspecting files like */proc/cpufino*, */proc/meminfo* or using tools like *top*.

- Modify the hello world program, so that also the number of logical cores is printed out. Run the program on the host and on the MIC.

- Compile the program *pthreadspin.c* using "*icpc –mmic –O0 –lpthread*" for the MIC architecture. Run the program using *micnativeloadex*. Login to the MIC and watch the CPU load using *top* and *ps.* Look on the threads using *ps –eLF*.

Lab 2: Offloading simple code to Intel Xeon Phi

- Add a new code block which prints "MIC: Hello world from MIC" to the hello world program. Add an offload pragma for the MIC architecture.

- Run the code on the login node vs. the compute nodes.

- Extend the "hello world" functions to print out the hostname and the numbers of cores of the MIC and the host.

- Compile using one of the compiler options *–offload=optional*, *-offload=mandatory* (Default) and *–offload=none*. Run each time on the login node and a compute node.

- Try to figure out more about the environment under which offloaded code is running. Offload *system("cmd")* calls to get info from commands like *set*, *hostname*, *uname –a*, *whoami*, *id* etc.

Lab 3: Offloading simple numerical code to Intel Xeon Phi

- Use the exercises c1.c and c2.c.

- Exchange the OpenACC pragmas *"#pragma acc kernels*" with an appropriate Intel Offload pragma.

- Compile using "*icpc –restrict*". How many threads are executing the binary?

- Parallelise using the appropriate OpenMP worksharing construct. To set the number of threads on the MIC you can use:
    - export MIC_ENV_PREFIX=MIC
    - export MIC_OMP_NUM_THREADS=…

- Export OFFLOAD_REPORT=2 and rerun the 2 programs. Dito for *H_TRACE=1* and *H_TIME=1*.

Lab 4: Offloading MxM code to Intel Xeon Phi

- Parallelize the matrix-matrix multiplication *matrixmul.cpp* using OpenMP.
- Compile using *icpc –mmic –vec-report3 [-offload=optional] -openmp*
- Run the program on the MIC natively or via micnativeloadex.
- Watch the program again on the MIC and via micsmc -a.
- Add an appropriate *offload target(mic)* pragma around the region with the for-loops.

- Add a function call *checkoffload(void)* to the Offload region which checks if the code is really running on the Coprocessor. The routine should print out where it is running depending on the value of __MIC__.
- Also print out the number of current / max OMP threads (*omp_get_num_threads(), omp_get_max_threads*).

- Test the stong scaling of the code. Run the code with different numbers of threads, but with same matrix size 2000. Write a small script that exports OMP_NUM_THREADS and starts the program for the following sizes.

Number of Threads	**Runtime(s)**
1	
2	
4	
8	
16	
32	
64	
128	
236	

- Write the data into a file and plot it, e.g. with gnuplot (*module load gnuplot*).
- Repeat for larger matrix sizes.
- Compare with the native Host / Xeon Phi performance.