

Programming environment and compilation

NERSC

Zhengji Zhao

User Engagement Group
New User Training, Bekerley CA
Jan 25, 2019

This talk is about the basics of compilations on Cori

- Compilation overview

- Compile/link lines:

Compiler +

Compiler Flags +

-I/path/to/headers +

-L/path/to/library -l<library>

- Available libraries, and linking examples

Users will need to apply the above info to their own build systems

Cori system configurations

- Cori KNL and Haswell – a Cray XC40
 - Cori has 9688 single-socket [Intel® Xeon Phi™ Processor 7250 \("Knights Landing"\)](#) nodes @1.4 GHz with 68 cores (272 threads) per node, two 512 bit vector units per core, and 16 GB high bandwidth on-package memory (MCDRAM) with 5X the bandwidth of DDR4 DRAM memory (>400 GB/sec) and 96 GB DDR4 2400 MHz memory per node
 - In addition, Cori has 2388 dual-socket 16-core [Intel® Xeon™ Processor E5-2698 v3 \("Haswell"\)](#) nodes @2.3GHz with 32 cores (64 threads) per node, two 256 bit vector units per core, 128 GB 2133 MHz DDR4 memory
 - Cori nodes are interconnected with Cray's Aries network with Dragonfly topology
- Binary compatibility: Haswell binaries run on KNL, but not vice versa, because KNL supports the extended instruction sets
- Separate builds for Haswell and KNL are recommended for optimal performance

Compilations on Cori

- Three programming environments are supported on Cori
 - Intel, GNU and Cray compilers are available; **Intel** is the default
- The programming environment modules, “PrgEnv-intel”, “PrgEnv-gnu”, and “PrgEnv-cray”, which include the compilers and matching libraries, provide user friendly programming environments
- Use “module swap PrgEnv-Intel PrgEnv-cray” to switch compilers
- Using compiler wrappers provided by Cray, ftn, cc and CC for Fortran, C and C++ respectively, the header and library paths and libraries can be included in the compile/link line automatically.

Compilations on Cori (cont.)

- Cross compilation: compiling for compute nodes from login nodes (haswell)
- Default environment loads **craype-haswell** module on Cori, which sets “**CRAY_CPU_TARGET=haswell**” for Cori. Compilers build binaries that are optimized for Haswell processors by default when compiling with the compiler wrappers

Default programming environment on Cori:

```
zz217@cori05:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6
  2) nsrg/1.2.0
  3) intel/18.0.1.163
  4) craype-network-aries
  5) craype/2.5.12
  6) cray-libsci/17.09.1
  7) udreg/2.3.2-6.0.4.0_12.2__g2f9c3ee.ari
  8) ugni/6.0.14-6.0.4.0_14.1__ge7db4a2.ari
  9) pmi/5.0.12
 10) dmapp/7.1.1-6.0.4.0_46.2__gb8abda2.ari
 11) gni-headers/5.0.11-6.0.4.0_7.2__g7136988.ari
 12) xpmem/2.2.2-6.0.4.1_18.2__g43b0535.ari
 13) job/2.2.2-6.0.4.0_8.2__g3c644b5.ari
 14) dvs/2.7_2.2.36-6.0.4.1_16.2__g4c8274a
 15) alps/6.4.1-6.0.4.0_7.2__g86d0f3d.ari
 16) rca/2.2.15-6.0.4.1_13.1__g46acb0f.ari
 17) atp/2.1.1
 18) PrgEnv-intel/6.0.4
 19) craype-haswell
 20) cray-mpich/7.6.2
 21) altd/2.0
 22) darshan/3.1.4

zz217@cori05:~>
```

To compile for Cori Haswell

- Intel programming environment is the default

```
#to use Intel compilers
```

```
ftn -O3 mycode.f90      # Fortran  
cc -O3 mycode.c          # for C  
CC -O3 myC++code.C       # for C++
```

```
#to use GNU compilers
```

```
module swap PrgEnv-intel PrgEnv-gnu  
ftn -O3 mycode.f90      # Fortran  
cc -O3 mycode.c          # for C  
CC -O3 myC++code.C       # for C++
```

```
#to use Cray compilers
```

```
module swap PrgEnv-intel PrgEnv-cray  
ftn -O3 mycode.f90      # Fortran  
cc -O3 mycode.c          # for C  
CC -O3 myC++code.C       # for C++
```

Note, the compiler wrappers, **ftn**, **cc**, and **CC**, are not Cray compilers; they invoke the Intel, GNU, or Cray compilers under the hood, depending on the loaded programming environment module (PrgEnv-<compiler>)

To compile for Cori KNL

- Applications are cross compiled for KNL nodes from the login nodes (Haswell)
- Do “**module swap craype-haswell craype-mic-knl**” before compiling for KNL to build binaries that are optimized for the KNL architecture

```
module swap craype-haswell craype-mic-knl
ftn -O3 mycode.f90          # Fortran
cc -O3 mycode.c             # for C
CC -O3 myC++code.C          # for C++
```

Compiler recommendations

- Will not recommend any specific compiler
 - Intel - better chance of getting processor specific optimizations, especially for KNL
 - Cray compiler – many new features and optimizations, especially with Fortran; useful tools like reveal work with Cray compiler only
 - GNU - widely used by open software
- Start with the compilers that vendor/code developers used so to minimize the chance to hit the compiler and code bugs, then explore different compilers for optimal performance

Compiler flags

Intel	GNU	Cray	Description/ Comment
-O2	-O0	-O2	default
default , or -O3	-O2 or -O3,-Ofast	default	recommended
-qopenmp	-fopenmp	default, or -h omp	OpenMP
-g	-g	-g	debug
-v	-v	-v	verbose

- Validity check after compilation
- Compilers' default behavior could vary between compilers
 - Default number of OpenMP threads used is all CPU slots available for Intel and GNU compilers; 1 for Cray compiler
 - use compiler man page for available compiler optimization flags, man ifort

Header and library paths and libraries

- Manually:
 - find out the paths to the headers, and libraries, then add
“-I <header path> -L<library path> -l<libraries>” to your compile/link lines
- Automatically:
 - Using the compiler wrappers, which can do this for you
 - Compiler wrappers are strongly recommended

Compiler wrappers, ftn, cc and CC

- Use ftn, cc, and CC to compile Fortran, C and C++ codes, respectively, instead of invoking the native compilers directly, such as ifort, icc, icpc, gfortran, gcc, g++, etc.
 - The compiler wrappers wraps the underlying compilers with additional compiler and linker flags depending on the modules loaded in the environment
 - The same compiler wrapper command (e.g. ftn) is used to invoke any compilers supported on the system (Intel, GNU, Cray)
- Compiler wrappers do cross compilation
 - Compiling applications on login nodes to run on compute nodes
 - For some applications, may need to set the `-host=x86_64` configure option (if available) when compiling for KNL from a login node
 - If compiling on a KNL node is needed, do `"salloc -N 1 -q interactive -C knl -t 4:00:00"` to get on to a compute node

Compiler wrappers, ftn, cc and CC (cont.)

- Compiler wrappers link statically by default
 - Preferred for performance at scale
 - This default will be “dynamic” when Cori is upgraded to CLE7 (end of July)
- Use the **-dynamic** option of the compiler wrappers or set the environment variable **“CRAYPE_LINK_TYPE=dynamic”** to link dynamically
 - May need to load the same set of modules at run time or set the LD_LIBRARY_PATH env so that shared libraries can be found. Alternatively, consider using the “-Wl,-rpath=<library path>” option when compiling
 - A dynamically linked executable may take some time to load shared libraries when running with a large number of processes

Why compiler wrappers?

- They include the architecture specific compiler flags into the compilation/link lines automatically

	Intel*)	GNU	Cray	Module
Cori KNL	-xMIC-AVX512	-march=knl	-h cpu=mic-knl	craype-mic-knl
Cori Haswell	-xCORE-AVX2	-march=core-avx2	-h cpu=haswell	craype-haswell

*) for the latest Intel compilers, -march=knl,haswell can be used instead of -xcode.

- Automatically add header and library paths and libraries on the compilation/link lines
 - Compiler wrappers use the pkg-config tools to dynamically detect paths and libs from the environment (working with cray modules and some NERSC modules)
 - The architecture specific builds of libraries will be linked into
- Allow user provided options to take precedence

Verbose output from compiler wrappers

- Depending on the modules loaded, compiler wrappers link to the MPI, LAPACK/BLAS/ScaLAPACK libraries, and more automatically
- Library names on Cori could be different from what you used before

```

Zz217@cori07:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6
  2) nsg/1.2.0
  3) intel/18.0.1.163
  4) craype-network-aries
  5) craype/2.5.12
  6) cray-libsci/17.09.1
  7) udreg/2.3.2-6.0.4.0_12.2_g2f9c3ee.ari
  8) ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari
  9) pmi/5.0.12
 10) dmapp/7.1.1-6.0.4.0_46.2_gb8abda2.ari
 11) gni-headers/5.0.11-6.0.4.0_7.2_g7136988.ari
 12) xpmem/2.2.2-6.0.4.1_18.2_g43b0535.ari
 13) job/2.2.2-6.0.4.0_8.2_g3c644b5.ari
 14) dvs/2.7.2.2-36-6.0.4.1_16.2_g4c8274a
 15) alps/6.4.1-6.0.4.0_7.2_g86d0f3d.ari
 16) rca/2.2.15-6.0.4.1_13.1_g46acb0f.ari
 17) atp/2.1.1
 18) PrgEnv-intel/6.0.4
 19) craype-haswell
 20) cray-mpich/7.6.2
 21) altd/2.0
 22) darshan/3.1.4

Zz217@cori07:~/tests/dgemm> ftn -v dgemmx.f -Wl,-ydgemm_
...

/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/bin/ld /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crt1.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crti.o /usr/lib64/gcc/x86_64-suse-linux/4.8/crtbeginT.o --build-id -static -m elf_x86_64 -L/opt/cray/pe/libsci/1.7.09.1/INTEL/16.0/x86_64/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/usr/common/software/darshan/3.1.4/lib -L/opt/cray/rca/2.2.15-6.0.4.1_13.1_g46acb0f.ari/lib64 -L/opt/cray/alps/6.4.1-6.0.4.0_7.2_g86d0f3d.ari/lib64 -L/opt/cray/xpmem/2.2.2-6.0.4.1_18.2_g43b0535.ari/lib64 -L/opt/cray/pmi/5.0.12/lib64 -L/opt/cray/ugni/6.0.14-6.0.4.0_14.1_ge7db4a2.ari/lib64 -L/opt/cray/udreg/2.3.2-6.0.4.0_12.2_g2f9c3ee.ari/lib64 -L/opt/cray/atp/2.1.1/libApp -L/lib64 -L/opt/cray/wlm_detect/1.2.1-6.0.4.0_22.1_gd26a3dc.ari/lib64 -o a.out /opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin_for_main.o -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64 -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64 -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -L/usr/lib64/gcc/x86_64-suse-linux/4.8/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/./lib64 -L/lib/./lib64 -L/lib/./lib64 -L/lib/./lib64 -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64 -L/opt/intel/compiler_and_libraries_2018.1.163/linux/mkl/lib/intel64/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/lib/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/ -L/usr/lib64 -L/usr/lib/tmp/ifortU7hqzk.o -ydgemm @/usr/common/software/darshan/3.1.4/share/ld-opts/darshan-base-ld-opts -lfmpich -lmpchcxx --start-group -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -latpSigHandler -latpSighCommData -undefined=ATP_Data_Globals -undefined=_atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugni -lpmpi -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmpi -lpthread -lalpsll -lwlm_detect -lalpsutil -lpthread -lrca -lxpmem -lugni -lpthread -ludreg -lscintel -L/opt/intel/compiler_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lugetlbfs --as-needed -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed -lifport -lifcore -limf -lsvml -lm -ligpo -lirc -lsvml -lc -lgcc -lgcceh -lirc_s -ldl -lc /usr/lib64/gcc/x86_64-suse-linux/4.8/crtd.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crtn.o /tmp/ifortU7hqzk.o: reference to dgemm
/opt/cray/pe/libsci/17.09.1/INTEL/16.0/x86_64/lib/libsci_intel.a(dgemm.o): definition of dgemm

```

Verbose output from compiler wrappers (cont.)

```

z217@cori07:~/tests/dgmemm> ftn -v dgmemm.f -WL,-ydgmemm_
...

/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/bin/ld /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crt1.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crti.o /usr/lib64/gcc/x86_64-suse-linux/4.8/crtbeginT.o --build-id -static -m elf_x86_64 -L/opt/cray/pe/libsci/17.09.1/INTEL/16.0/x86_64/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/usr/common/software/darshan/3.1.4/lib -L/opt/cray/rca/2.2.15-6.0.4.1.13.1_g46acbf0f.ari/lib64 -L/opt/cray/alps/6.4.1-6.0.4.0.7.2_g86d0f3d.ari/lib64 -L/opt/cray/xpmem/2.2.2-6.0.4.1.18.2_g43b0535.ari/lib64 -L/opt/cray/pe/pmi/5.0.12/lib64 -L/opt/cray/ugni/6.0.14-6.0.4.0.14.1_ge7db42.ari/lib64 -L/opt/cray/udreg/2.3.2-6.0.4.0.12.2_g2f9c3ee.ari/lib64 -L/opt/cray/wlm_detect/1.2.1-6.0.4.0.22.1_gd26a3dc.ari/lib64 -o a.out /opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin /opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64_lin /opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/../../lib64/ -L/usr/lib/../../lib64/ -L/usr/lib/../../lib64/ -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64/ -L/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/lib -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/../../lib64/ -L/usr/lib64 /tmp/iftort7hqq2k.o -ydgmemm_ @/usr/common/software/darshan/3.1.4/share/ld-opts/darshan-base-ld-opts -lfmpich -lmpichcxx --start-group -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -latp5igHandler -latp5igHCommData -undefined AT_PData_Globals -undefined _atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugini -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lmpich-intel -lrt -lugini -lpthread -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmi -lpthread -latpslli -lpthread -lwlm_detect -lalp_sutil -lpthread -lrca -lxpmem -lugini -lpthread -ludreg -lsci_intel -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lhugetlbfs --as-needed -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed -lifport -lifcore -limf -lipo -lirc -lsvml -lc -lgcc -lgcc_eh -lirc_s -ldl -lc /usr/lib64/gcc/x86_64-suse-linux/4.8/crtend.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crtn.o
/tmp/iftort7hqq2k.o: reference to dgmemm_
/opt/cray/pe/libsci/17.09.1/INTEL/16.0/x86_64/lib/libsci_intel.a(dgmemm.o): definition of dgmemm_

z217@cori07:~/tests/dgmemm> ftn -v dgmemm.f -o dgmemm.x -mkl -WL,-ydgmemm_
Warning:
  Headers and libraries from cray-libsci/17.09.1 will be ignored because they conflict with -mkl.
...

/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/bin/ld /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crt1.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crti.o /usr/lib64/gcc/x86_64-suse-linux/4.8/crtbeginT.o --build-id -static -m elf_x86_64 -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.6.2/gni/mpich-intel/16.0/lib -L/usr/common/software/darshan/3.1.4/lib -L/opt/cray/rca/2.2.15-6.0.4.1.13.1_g46acbf0f.ari/lib64 -L/opt/cray/alps/6.4.1-6.0.4.0.7.2_g86d0f3d.ari/lib64 -L/opt/cray/xpmem/2.2.2-6.0.4.1.18.2_g43b0535.ari/lib64 -L/opt/cray/pe/pmi/5.0.12/lib64 -L/opt/cray/ugni/6.0.14-6.0.4.0.14.1_ge7db42.ari/lib64 -L/opt/cray/udreg/2.3.2-6.0.4.0.12.2_g2f9c3ee.ari/lib64 -L/opt/cray/wlm_detect/1.2.1-6.0.4.0.22.1_gd26a3dc.ari/lib64 -o dgmemm.x /opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin /opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64_lin /opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/../../lib64/ -L/usr/lib/../../lib64/ -L/usr/lib/../../lib64/ -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64/ -L/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../x86_64-suse-linux/lib -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64 -L/lib/../../lib64/ -L/usr/lib64 /tmp/iftort7hqq2k.o -ydgmemm_ @/usr/common/software/darshan/3.1.4/share/ld-opts/darshan-base-ld-opts -lfmpich -lmpichcxx --start-group -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -latp5igHandler -latp5igHCommData -undefined AT_PData_Globals -undefined _atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugini -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lmpich-intel -lrt -lugini -lpthread -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmi -lpthread -latpslli -lpthread -lwlm_detect -latpsutil -lpthread -lrca -lugini -lpthread -lxpmem -ludreg --as-needed -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed -lifport --no-as-needed -lifcore -limf -lipo -lirc -lpthread -lsvml -lc -lgcc -lgcc_eh -lirc_s -ldl -lc /usr/lib64/gcc/x86_64-suse-linux/4.8/crtend.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crtn.o
/tmp/iftort7hqq2k.o: reference to dgmemm_
/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64/libmkl_intel_lp64.a(dgmemm_lp64.o): definition of dgmemm_
/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64/libmkl_core.a(mkl_semaphore.o): In function 'mkl_serv_load_inspector':
mkl_semaphore.c:(.text+0x123): warning: Using 'dlopen' in statically linked applications requires at runtime the shared libraries from the glibc version used for linking

```

Available libraries

- Cray supports many software packages – Cray Developer Toolkits (CDT)
 - Modules from /opt/cray/pe/modulefiles, etc.
 - Access via modules, type “module avail” or “module avail –S < your string>” to see the available modules
 - There are different builds for different compilers
 - Programming environment modules allow the libraries built with the matching compilers to be linked to
- NERSC staff also supports many libraries
 - Modules from /usr/common/software/modulefiles, etc.
 - Some of them interact with the Cray compiler wrappers while many of them do not

Available libraries (cont.)

- Where are the libraries and header files ?
 - Use “module show <module name>” to see the installation paths
 - Run “ls -l <installation_path>/include” and “ls -l <installation_path>/lib” to see the library files
 - e.g., Cray MPICH library:

```
cori01:~> module show cray-mpich
-----
/opt/cray/pe/modulefiles/cray-mpich/7.7.3:
...
setenv  CRAY_MPICH_DIR /opt/cray/pe/mpt/7.7.3/gni/mpich-intel/16.0
Setenv  MPICH_DIR /opt/cray/pe/mpt/7.7.3/gni/mpich-intel/16.0
...
```

```
cori01:~> ls -l $CRAY_MPICH_DIR
drwxr-xr-x 2 root root 628 Mar 15 14:46 include
drwxr-xr-x 3 root root 1743 Mar 15 14:46 lib
```

Examples of linking to the Cray provided libraries

- Linking to Cray MPI and Cray Scientific libraries are automatic by default if compiler wrappers are used

```
CC parallel_hello.cpp    #or    ftn dgemmx1.f90
```

- Linking to HDF5 and NETCDF libraries are automatic, user just need to load the cray-hdf5 or cray-netcdf modules

```
module load cray-hdf5  
cc h5write.c
```

- Note the library name could be different. Using the `-v` option to see the library names and other detail about the linking

Examples of linking to the Cray provided libraries

- Linking to PETSc libraries are automatic, but users need to choose a proper module (e.g., real/complex, 32 or 64 bit integer builds)
 - E.g., “module load cray-petsc-complex-64”
 - Use “cc -v test1.c” to see the linking detail (test1.c can be any skeleton C code)
- Linking to fftw libraries – fftw 3 is the default
 - module load cray-fftw
 - Loading the cray-fftw module always links to the pthread version of the library, “-lfftw3f_mpi -lfftw3f_threads -lfftw3f -lfftw3_mpi -lfftw3_threads -lfftw3”, to link with the OpenMP version of FFTW, you need to manually provide the libraries

Examples of linking to the NERSC provided library modules

- Some of the NERSC provided modulefiles are written to interact with the Cray compiler wrappers, e.g., elpa module on Cori

```
module load elpa
#automatically link to elpa and MKL ScaLAPACK libraries
ftn -qopenmp -v test2.f90
```

- Type “module show <module name>” to check if the envs “<libname>_PKGCONFIG_LIBS”, “PE_PKGCONFIG_PRODUCTS”, and “PKG_CONFIG_PATH” are defined in the modulefiles, which compiler wrappers look for
- Most of the NERSC provided modulefiles do not interact with the compiler wrappers, user need to provide the include and library paths and libraries manually, e.g., GSL

```
module load gsl
ftn test3.f90 $GSL
#where GSL=-I/usr/common/software/gsl/2.1/intel/include -L/usr/common/software/gsl/2.1/intel/lib -lgsl -lgslcblas
```

Linking to Intel MKL library

- Resource:
 - Intel® Math Kernel Library Link Line Advisor,
<https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/>
 - Learn from Intel compiler verbose output using the “-mkl={parallel,sequential,cluster}” flag
- For intel compiler, use -mkl flag
 - `ftn test1.f90 -mkl` #default to parallel, the multi-threaded MKL
 #the loaded cray-libsci will be ignored if -mkl is used.

Linking to Intel MKL library (cont.)

- For GNU compiler (e.g., to link to 32-bit integer build):
 - Save the `$MKLROOT` from the Intel compiler module, and then
 - Threaded: `“-L$MKLROOT/lib/intel64 -Wl,--start-group -lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_core -lgomp -Wl,--end-group -lpthread -lm -ldl”`
 - ScaLAPACK: `“-L$MKLROOT/lib/intel64 -Wl,--start-group -lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_core -Wl,--end-group -lgomp -lpthread -lm -ldl”`
 - Notice that the NERSC provided mkl modules could be outdated

Linking to Intel MPI library: use native Intel compilers

- Cray MPICH libraries are recommended for performance especially at scale
- Compiler wrappers links to Cray MPICH libraries
- However, if you need to link to Intel MPI library, do

```
module load impi  
mpiifort test1.f90          #or mpiicpc test1.C
```

- Note that the binaries linked to the Intel MPI need to run with srun instead of mpirun to get a proper process/thread affinity, <https://docs.nersc.gov/jobs/examples/#using-intel-mpi>
- Native Intel compilers link dynamically

Summary

- Three supported programming environments: **Intel**, GNU, and Cray
- **Use compiler wrappers where possible,**
 - Add architecture specific optimization flags
 - Automatically add the header and library paths in to the compile/link lines, and link to the Cray MPI, LibSci and other Cray provided libraries if the modules are loaded
- To compile for Cori KNL, do
 - `module swap craype-haswell craype-mic-knl`
- There are many libraries available, use them where possible
 - Use “module avail” command to check available libraries
 - Use “module show <module name>” to see the installation paths if needed

Summary (cont.)

- Most NERSC staff support modules do not interact with the compiler wrappers
 - Users need to provide the header and library paths and libraries manually
- To link applications dynamically, use the “-dynamic” compiler wrapper option or set the env “CRAYPE_LINK_TYPE=dynamic” before compiling
- **Learn from the compiler verbose output (-v)**

Recommended readings

- NERSC website, especially,
 - <http://www.nersc.gov/users/computational-systems/cori/programming/compiling-codes-on-cori/>
 - We are moving user documentation pages to <http://docs.nersc.gov>,
 - <https://docs.nersc.gov/development/compilers/>
 - For further compiler optimizations read intel slides: e.g., <https://www.nersc.gov/users/training/events/intel-compilers-tools-and-libraries-training-march-6-2018/>
- Compiler and linker man pages:
 - ifort, icc, icpc, crayftn, etc.
 - man ld (“-Wl,-zmuldefs”, “-Wl,-y<symbol>”)



Thank You!

