Programming environment and compilations

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

We will talk about only

- Compilations for Cori (Haswell and KNL nodes) and Edison
- Compile/link lines

  Compiler +

  Compiler Flags +

  –I/path/to/headers +

  –L/path/to/library –l<library>

- Available libraries, linking examples
- Users will need to apply the above info to their own build systems.

Cori Haswell, KNL and Edison 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.
- Edison – a Cray XC30
  - Edison has 5586 dual-socket 12-core **Intel(R) Xeon(R) CPU E5-2695 v2 ("Ivy Bridge")** nodes @2.40GHz with 24 cores (48 threads) per node, two 256 bit vector units per core, 64 GB DDR3 1866 MHz memory. Edison nodes are interconnected with Cray's Aries network with Dragonfly topology.

Compilations on Cori and Edison are very similar

- Three programming environments are supported
  - Intel, GNU and Cray compilers are available on Cori. Intel is the default.
  - PrgEnv-intel, PrgEnv-gnu, and PrgEnv-cray loads the corresponding programming environment which includes the compilers and matching libraries.
  - Using module swap PrgEnv-Intel PrgEnv-cray to swap programing environment.
  - Compiler wrappers, ftn, cc and CC, are recommended instead of the native compiler invocations.

Compilations on Cori and Edison are very similar

- Default environment loads craype-haswell/craype-ivybridge module on Cori/Edison, which sets the env CRAY_CPU_TARGET to haswell/ivybridge for Cori/Edison – taking care of the architectural difference when compiling with the compiler wrappers.
- Cross compilation: compiling for compute nodes from login nodes

Default programming environment on Cori:

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

Compilations on Cori and Edison are very similar

- To compile for Cori Haswell and Edison:

```
#to use Intel compiler:

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

```
#to use GNU compiler
module swap PrgEnv-intel PrgEnv-gnu
ftn –O3 mycode.f90.       # Fortran:
cc –O3 mycode.c           # for C
CC –O3 myC++code.C.       # for C++
```

- Note, the compiler wrappers invoke the Intel, GNU, or Cray compilers under the hood.

To compile for Cori KNL

- Unlike Edison, Cori has two types of compute nodes, Haswell, and KNL, and applications are compiled for both Haswell and KNL nodes from the login nodes (Haswell nodes).
- do module swap craype-haswell craype-mic-knl before compiling for Cori KNL nodes

```
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++
```

Binary compatibility

Build system	Edison	Cori Haswell	Cori KNL
Edison	✓	✓	✓
Cori Haswell		✓	✓
Cori KNL			✓

- Edison binaries runs on Cori Haswell, and KNL; Haswell Binaries run on KNL
- We recommend a separate build for each platform for optimal performance.

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.

Compiler wrappers, ftn, cc and CC, are recommended

- Use ftn, cc, and CC to compile Fortran, C and C++ codes, respectively, instead of the underlying native compilers, such as ifort, icc, icpc, gfortran, gcc, g++, etc.
  - The compiler wrappers wraps the underlying compilers with the 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 on login nodes to run on compute nodes

Compiler wrappers, ftn, cc and CC, are recommended

- Use the –host=x86_64 configure option when compiling for KNL from a login node
- To compile on a KNL node, do salloc –N 1 –q interactive –C knl –t 4:00:00 to get on a compute node

- Compilers wrappers link statically by default
  - Preferred for performance at scale
- Use –dynamic or set an environment variable CRAYPE_LINK_TYPE=dynamic to link dynamically
  - A dynamically linked executable may take a some time to load shared libraries when running with a large number of processes
- User provided options take precedence

Why compiler wrappers?

- They include the architecture specific compiler flags into the compilation/link line 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
Edison Ivy Bridge	-xCORE-AVX-I	-march=corei7-avx	-h cpu=ivybridge	craype-ivybridge

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

- Automatically add the 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

What do compiler wrappers link by default?

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

```
zz217@cori07:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6                                 9) pmi/5.0.12                                      17) atp/2.1.1
  2) nsg/1.2.0                                       10) dmapp/7.1.1-6.0.4.0_46.2__gb8abda2.ari          18) PrgEnv-intel/6.0.4
  3) intel/18.0.1.163                                11) gni-headers/5.0.11-6.0.4.0_7.2__g7136988.ari    19) craype-haswell
  4) craype-network-aries                            12) xpmem/2.2.2-6.0.4.1_18.2__g43b0535.ari          20) cray-mpich/7.6.2
  5) craype/2.5.12                                   13) job/2.2.2-6.0.4.0_8.2__g3c644b5.ari             21) altd/2.0
  6) cray-libsci/17.09.1                             14) dvs/2.7_2.2.36-6.0.4.1_16.2__g4c8274a           22) darshan/3.1.4
  7) udreg/2.3.2-6.0.4.0_12.2__g2f9c3ee.ari          15) alps/6.4.1-6.0.4.0_7.2__g86d0f3d.ari
  8) ugni/6.0.14-6.0.4.0_14.1__ge7db4a2.ari          16) rca/2.2.15-6.0.4.1_13.1__g46acb0f.ari

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/cra
y/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/pe/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/pe/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/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin/for_main.o -L/opt/intel/comp
ilers_and_libraries_2018.1.163/linux/compiler/lib/intel64 -L/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64 -L/opt/intel/compilers_an
d_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/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/../../../ -L/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 -lmpichcxx --start-group -ldarshan -ldarshan-stubs --end-group
-lz --no-as-needed -lAtpSigHandler -lAtpSigHCommData --undefined=_ATP_Data_Globals --undefined=__atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugni
 -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lsci_intel_mpi -lsci_intel -L/opt/intel/c
ompilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lmpich_intel -lrt -lugni -lpthread -lpmi -L/opt/intel/compilers_and_libr
aries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lalpsutil -lpthread -lrca -lxpmem -lugni
 -lpthread -ludreg -lsci_intel -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lhugetlbfs --as-n
eeded -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed -lifport -lifcore -limf -lsvml -lm -lipgo -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/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_
```

More on the verbose output from compiler wrappers

```
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/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__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_1
8.2__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__ge7db4a2.ari/lib64 -L/opt/cray/udreg/2.3.2-6.0.4.0_12.2__g2f9c3ee.ari/lib64 -
L/opt/cray/pe/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/compilers_and_libraries_2018.1.163/linux/compiler
/lib/intel64_lin/for_main.o -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/intel6
4 -L/opt/intel/compilers_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/usr/lib/../lib64 -L/usr/lib/../lib64/ -L/opt/intel/compilers_and_libra
ries_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_6
4-suse-linux/lib/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../ -L/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 -lmpichcxx --start-group -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -lAtpSigHandler -lAtpSigHCommData --undefine
d=_ATP_Data_Globals --undefined=__atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugni -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64
_lin -limf -lm -lpthread -ldl -lsci_intel_mpi -lsci_intel -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lmpich_intel -lrt
-lugni -lpthread -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lalp
sutil -lpthread -lrca -lxpmem -lugni -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 -lsvml -lm -lipgo -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/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_



zz217@cori07:~/tests/dgemm> ftn -v dgemmx.f -o dgemm.x -mkl -Wl,-ydgemm_
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/gn
i/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/pe/pmi/5.0.12/li
b64 -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/pe/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 dgemm.x /opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin/for_main.o -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/opt/intel/compilers_and_libraries_2018.1.163/
linux/compiler/lib/intel64_lin -L/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64_lin -L/usr/lib64/gcc/x86_64-suse-linux/4.8/ -L/usr/lib64/gcc/x86_64-su
se-linux/4.8/../../../../lib64 -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/ -L/lib/../lib64 -L/lib/../lib64/ -L/usr/lib/../lib64 -L/usr/lib/../lib64/ -L/opt/i
ntel/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-lin
ux/4.8/../../../../x86_64-suse-linux/lib/ -L/usr/lib64/gcc/x86_64-suse-linux/4.8/../../../ -L/lib64 -L/lib/ -L/usr/lib64 -L/usr/lib /tmp/ifortSBiwCG.o --start-group -lmkl_
intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 --end-group -ydgemm_ @/usr/common/software/darshan/3.1.4/share/ld-opts/darshan-base-ld-opts -lfmpich -lmpichcxx --start-gr
oup -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -lAtpSigHandler -lAtpSigHCommData --undefined=_ATP_Data_Globals --undefined=__atpHandlerInstall -lpthread -lmp
ichf90_intel -lrt -lugni -lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lmpich_intel -lrt -lugni -lpthread
-lpmi -L/opt/intel/compilers_and_libraries_2018.1.163/linux/compiler/lib/intel64_lin -limf -lm -ldl -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lalpsutil -lpthread -
lrca -lugni -lpthread -lxpmem -ludreg --as-needed -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed --start-group -lmkl_intel_lp64 -
lmkl_intel_thread -lmkl_core -liomp5 --end-group -lifport -lifcore -limf -lsvml -lm -lipgo -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/ifortSBiwCG.o: reference to dgemm_
/opt/intel/compilers_and_libraries_2018.1.163/linux/mkl/lib/intel64/libmkl_intel_lp64.a(_dgemm_lp64.o): definition of dgemm_
/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
```

Note, -Wl,--start-group … -Wl,--end-group for static linking

Available libraries

- Cray supports many software packages – Cray Developer Toolkits (CDT)
  - Access via modules, type “module avail” or “module avail –S” 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 also supports many libraries
  - Some of them interact with the Cray compiler wrappers while many of them do not.
- Where are the libraries ?
  - Use “module show <module name> “ to see the installation paths
  - ls –l <installation_path> to see the library files

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 detailed link line information.

Examples of linking to the Cray provided libraries

- Linking to PETSc libraries are automatic, but users need to choose a proper module (real/complex,32/64 bit integer)
  - E.g., module load cray-petsc-complex-64
  - Use cc –v test1.c to see the linking detail

- 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 OpenMP implementation, 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

  ftn –qopenmp –v test2.f90 # this will automatically link to elpa and MKL ScaLAPACK libraries

  - Type module show <module name> to check if the envs <libname>_PKGCONFIG_LIBS, PE_PKGCONFIG_PRODUCTS, and PKG_CONFIG_PATH are defined in the modulefile, which compiler wrappers look for.

- Most of the NERSC provided modulefiles do not interact with the compiler wrappers, user need to provide the include path and library path and libraries manually, e.g. GSL

  module load gsl; ftn test3.f90 $GSL

  - GSL is set as -I/usr/common/software/gsl/2.1/intel/include -L/usr/common/software/gsl/2.1/intel/lib -lgsl -lgslcblas

Liking example

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, -mkl={parallel,sequential,cluster}
- For intel compiler, use –mkl flag
  - ftn test1.f90 –mkl # default to parallel –multi-threaded lib
  - The loaded cray-libsci will be ignored if –mkl is used.

Linking to Intel MKL library

- For GNU compiler (e.g., to link to 32-bit integer build):
  - Save the MKLROOT from the Intel compiler module, and then
  - Threaded: -L$MKLROOT/lib/intel64 –Wl,--start-group -lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_core -liomp5 -Wl,--end-group –lpthread –lm –ldl
  - ScaLAPACK: -L$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
  - Note that mkl modules could be out-dated

Linking to Intel MPI library – Use native 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
```

  - Note that the binaries linked to the Intel MPI need to run with srun instead of mpirun to get a proper process/thread affinity, http://www.nersc.gov/users/computational-systems/cori/running-jobs/advanced-running-jobs-options/#toc-anchor-6
  - Native Intel compilers link dynamically

Liking example

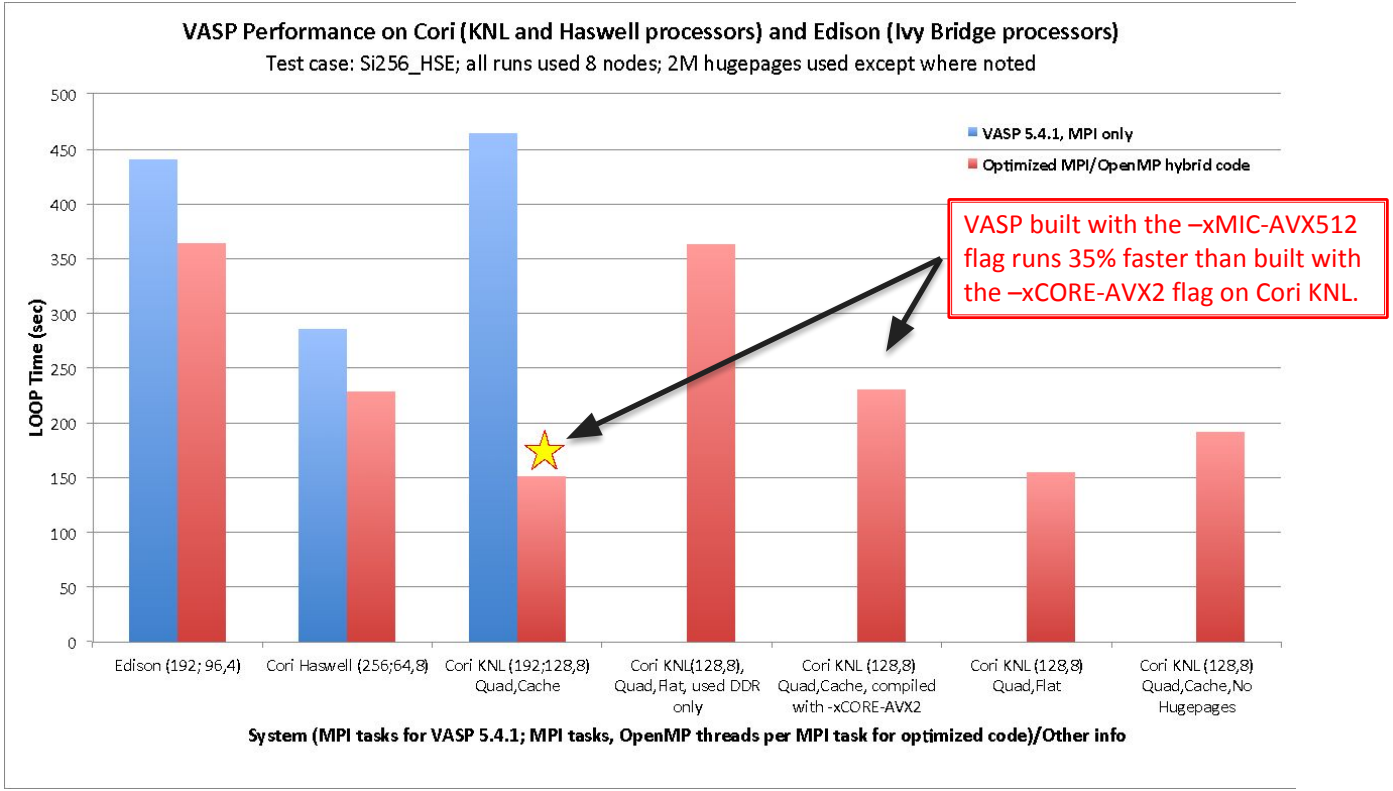
Summary

- Compilations for Cori and Edison are very similar
- To compile for Cori KNL, do
  - `module swap craype-haswell craype-mic-knl`
- Use compiler wrappers where possible, they
  - add architecture specific optimization flags
  - link to the Cray MPI and LibSci libraries and other Cray provided libraries
- Use available libraries where possible
  - Use module avail command to check available libraries
  - Use module show <module name> to see the installation paths if needed
- 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/
  - http://www.nersc.gov/users/computational-systems/edison/programming/
  - 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>)


NERSC
Thank You!



Building your application separately for each platform could be important to get optimal performance.