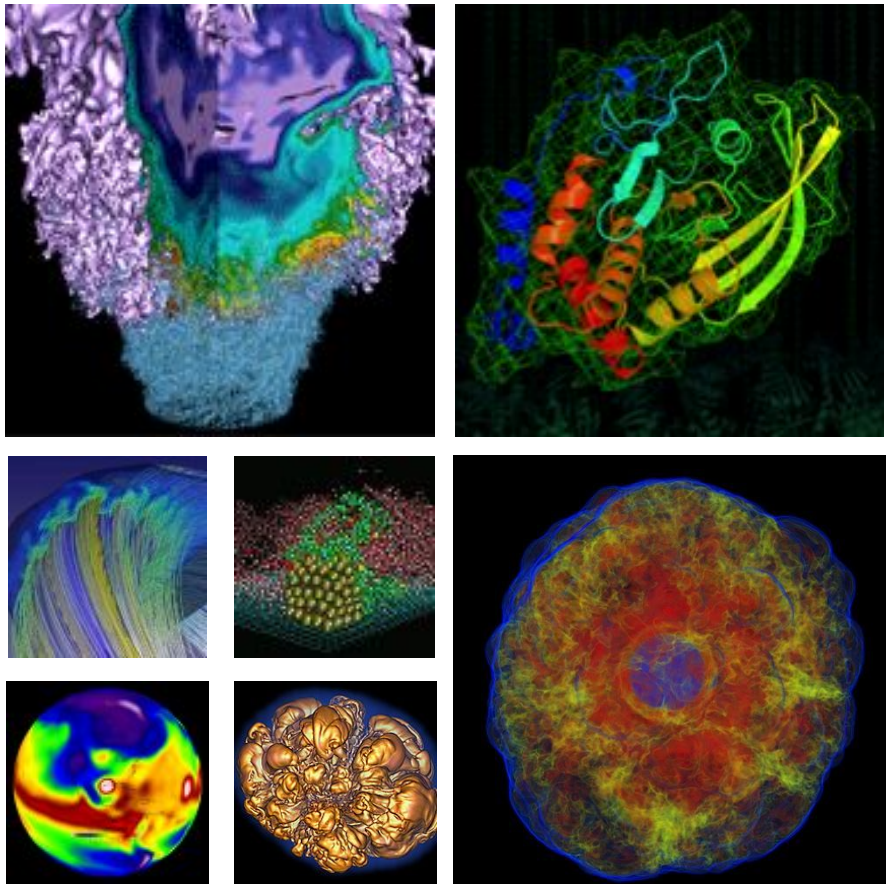
Compiling Codes at NERSC



Zhengji Zhao

NERSC New User Training,
February 23, 2017, Berkeley CA

NERSC 40 YEARS at the FOREFRONT 1974-2014

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

This talk will focus on

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

  Compiler +

  Compiler Flags +

  –I/path/to/headers +

  –L/path/to/library –l<library>

- Available compilers, libraries, how to compile - a couple of tips, compile for Cori KNL, summary

Separate builds for Cori and Edison is recommended

- **Cori KNL and Haswell**
  - Cori has over 9300 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 over 2000 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**
  - 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 Edison and Cori are very similar, and binaries can be built to be compatible, however, however, we recommend to a separate build for each platform.**

Intel, GNU and Cray compilers on Cori, and Edison

- **Intel compiler is default on both Cori and Edison**
- **Access through modules**
- **Three programming environments**
  - PrgEnv-intel, PrgEnv-gnu, and PrgEnv-cray
  - Module swap PrgEnv-Intel PrgEnv-cray to swap compilers and programing environment.

```
zz217@cori03:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.5                  9) pmi/5.0.10-1.0000.11050.0.0.ari  17) atp/2.0.3
  2) nsg/1.2.0                        10) dmapp/7.1.0-12.37                18) PrgEnv-intel/6.0.3
  3) intel/17.0.1.132                 11) gni-headers/5.0.7-3.1            19) craype-haswell
  4) craype-network-aries             12) xpmem/0.1-4.5                    20) cray-shmem/7.4.4
  5) craype/2.5.7                     13) job/1.5.5-3.58                   21) cray-mpich/7.4.4
  6) cray-libsci/16.09.1              14) dvs/2.7_0.9.0-2.243              22) altd/2.0
  7) udreg/2.3.2-4.6                  15) alps/6.1.3-17.12                 23) darshan/3.0.1.1
  8) ugni/6.0.12-2.1                  16) rca/1.0.0-8.1
zz217@cori03:~>
```

- **Other available compilers, such as bupc, llvm, etc, will not be covered in this presentation.**

Use compiler wrappers, ftn, cc and CC

- **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++,crayftn,craycc, crayc++**
  - The compiler wrappers are the same as the underlying compilers with the addition of flags included and libraries linked by default
  - The same compiler wrapper command (.e.g. ftn) can refer to any underlying compiler available on the system (Intel, GNU, Cray)
- **Compilers wrappers link statically**
  - Preferred for performance at scale
- **Use –dynamic to link dynamically**
  - For larger scale runs, dynamically linked code may take a long time to load the shared libraries
- **Cross compiling**
  - Compile on login nodes to run on compute nodes

Why compiler wrappers?

- **They include the architecture specific compiler flags into the compilation/link line automatically.**

	Intel	GNU	Cray	Module
Cori KNL	-xAVX512	-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

- **They automatically add the header and library paths and libraries on the compilation/link lines, so you do not have to explicitly provide them.**
  - Compiler wrappers use the pkg-config tools to dynamically detect paths and libs from the environment (loaded cray modules and some NERSC modules)
- **Yet allow user provided options to take the precedence**
  - You may need to remove –xhost option in your compilation/link line

What do compiler wrappers link by default?

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

```
zz217@cori09:~/tests/dgemm> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.5                  7) udreg/2.3.2-4.6                   13) job/1.5.5-3.58                   19) craype-haswell
  2) nsg/1.2.0                         8) ugni/6.0.12-2.1                   14) dvs/2.7_0.9.0-2.243              20) cray-shmem/7.4.4
  3) intel/17.0.1.132                  9) pmi/5.0.10-1.0000.11050.0.0.ari  15) alps/6.1.3-17.12                 21) cray-mpich/7.4.4
  4) craype-network-aries             10) dmapp/7.1.0-12.37                 16) rca/1.0.0-8.1                    22) altd/2.0
  5) craype/2.5.7                     11) gni-headers/5.0.7-3.1             17) atp/2.0.3                        23) darshan/3.0.1.1
  6) cray-libsci/16.09.1              12) xpmem/0.1-4.5                     18) PrgEnv-intel/6.0.3

zz217@cori09:~/tests/dgemm> ftn -v dgemmx.f -Wl,-ydgemm_
...

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-lin
ux/4.8/crtbeginT.o --build-id -static -m elf_x86_64 -L/opt/cray/pe/mpt/7.4.4/gni/sma/lib64 -L/opt/cray/pe/libsci/16.09.1/INTEL/15.0/x86_64/lib -L/opt/cray/dmapp
/default/lib64 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -L/usr/common
/software/darshan//3.0.1.1/lib -L/opt/cray/rca/1.0.0-8.1/lib64 -L/opt/cray/alps/6.1.3-17.12/lib64 -L/opt/cray/xpmem/0.1-4.5/lib64 -L/opt/cray/dmapp/7.1.0-12.37/
lib64 -L/opt/cray/pe/pmi/5.0.10-1.0000.11050.0.0.ari/lib64 -L/opt/cray/ugni/6.0.12-2.1/lib64 -L/opt/cray/udreg/2.3.2-4.6/lib64 -L/opt/cray/pe/atp/2.0.3/libApp -
L/lib64 -L/opt/cray/wlm_detect/1.1.0-4.2/lib64 -o a.out /opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64_lin/for_main.o -L/opt/intel/com
pilers_and_libraries_2017.1.132/linux/compiler/lib/intel64 -L/opt/intel/compilers_and_libraries_2017.1.132/linux/mkl/lib/intel64 -L/opt/intel/compilers_and_libr
aries_2017.1.132/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/gc
c/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_2017.1
.132/linux/compiler/lib/intel64/ -L/opt/intel/compilers_and_libraries_2017.1.132/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/ifortsk1ewd.o -ydgemm_ @/usr/common/software/da
rshan//3.0.1.1/share/ld-opts/darshan-base-ld-opts -lfmpich -lmpichcxx --start-group -ldarshan -ldarshan-stubs --end-group -lz --no-as-needed -lAtpSigHandler -lA
tpSigHCommData --undefined=_ATP_Data_Globals --undefined=__atpHandlerInstall -lpthread -lmpichf90_intel -lrt -lugni -lpmi -L/opt/intel/compilers_and_libraries_2
017.1.132/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lsma -lpmi -lsma -lpmi -ldmapp -lpthread -lsci_intel_mpi -lsci_intel -L/opt/intel/compilers_a
nd_libraries_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -ldl -lmpich_intel -lrt -lugni -lpthread -lpmi -L/opt/intel/compilers_and_libraries_2017.1.132/
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_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -ldl --as-needed -limf --no-as-needed --as-needed -lm --no-as-ne
eded --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/ifortsk1ewd.o: reference to dgemm_
/opt/cray/pe/libsci/16.09.1/INTEL/15.0/x86_64/lib/libsci_intel.a(dgemm_.o): definition of dgemm_
```

Compiler recommendations

- **Will not recommend any specific compiler**
  - Intel - better chance of getting processor specific optimizations
  - Cray compiler – many new features and optimizations, especially with Fortran
  - GNU - widely used by open software
- **Start with the compilers that vendor/code developers used, to minimize the chance to hit the compiler and code bugs, then explore different compilers if you care the 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**
  - Run tests and check with the references if provided
  - If higher Debug version to check the validity
- **Default behaviors could varies between compilers**
  - Default number of OpenMP threads
  - * -g: Cray compiler disables all optimizations, try –G2

Verbose output from compiler wrappers contains many useful information

```
zz217@cori05:~/tests/dgemm> ftn -v  dgemmx.f -o dgemmx -Wl,-ydgemm

...

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/mpt/7.4.4/gni/sma/lib64 -L/opt/cray/pe/libsci/16.09.1/INTEL/15.0/x86_64/lib -L/opt/cray/dmapp/default/lib64
 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -L/usr/common/software/darshan//3.
0.1.1/lib -L/opt/cray/rca/1.0.0-8.1/lib64 -L/opt/cray/alps/6.1.3-17.12/lib64 -L/opt/cray/xpmem/0.1-4.5/lib64 -L/opt/cray/dmapp/7.1.0-12.37/lib64 -L/opt/cray/pe/pmi/5.0
.10-1.0000.11050.0.0.ari/lib64 -L/opt/cray/ugni/6.0.12-2.1/lib64 -L/opt/cray/udreg/2.3.2-4.6/lib64 -L/opt/cray/pe/atp/2.0.3/libApp -L/lib64 -L/opt/cray/wlm_detect/1.1.
0-4.2/lib64 -o dgemmx /opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64_lin/for_main.o -L/opt/intel/compilers_and_libraries_2017.1.132/linux/com
piler/lib/intel64 -L/opt/intel/compilers_and_libraries_2017.1.132/linux/mkl/lib/intel64 -L/opt/intel/compilers_and_libraries_2017.1.132/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/../li
b64 -L/lib/../lib64/ -L/usr/lib/../lib64 -L/usr/lib/../lib64/ -L/opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64/ -L/opt/intel/compilers_and_li
braries_2017.1.132/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/ifortdGd0zx.o -ydgemm @/usr/common/software/darshan//3.0.1.1/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 -lpthre
ad -lmpichf90_intel -lrt -lugni -lpmi -L/opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lsma -lpmi -lsma -lpmi -
ldmapp -lpthread -lsci_intel_mpi -lsci_intel -L/opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -ldl -lmpich_intel -lrt -lugni -l
pthread -lpmi -L/opt/intel/compilers_and_libraries_2017.1.132/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_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -ldl --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 -li
rc_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



zz217@cori05:~/tests/dgemm> ftn -v  dgemmx.f -o dgemmx -mkl -Wl,-ydgemm

...

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/mpt/7.4.4/gni/sma/lib64 -L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -
L/opt/cray/dmapp/default/lib64 -L/opt/cray/pe/mpt/7.4.4/gni/mpich-intel/16.0/lib -L/usr/common/software/darshan//3.0.1.1/lib -L/opt/cray/rca/1.0.0-8.1/lib64 -L/opt/cra
y/alps/6.1.3-17.12/lib64 -L/opt/cray/xpmem/0.1-4.5/lib64 -L/opt/cray/dmapp/7.1.0-12.37/lib64 -L/opt/cray/pe/pmi/5.0.10-1.0000.11050.0.0.ari/lib64 -L/opt/cray/ugni/6.0.
12-2.1/lib64 -L/opt/cray/udreg/2.3.2-4.6/lib64 -L/opt/cray/pe/atp/2.0.3/libApp -L/lib64 -L/opt/cray/wlm_detect/1.1.0-4.2/lib64 -o dgemmx /opt/intel/compilers_and_libra
ries_2017.1.132/linux/compiler/lib/intel64_lin/for_main.o -L/opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64 -L/opt/intel/compilers_and_librari
es_2017.1.132/linux/mkl/lib/intel64 -L/opt/intel/compilers_and_libraries_2017.1.132/linux/compiler/lib/intel64_lin -L/opt/intel/compilers_and_libraries_2017.1.132/linu
x/mkl/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_2017.1.132/linux/compiler/lib/intel64/ -L/opt/
intel/compilers_and_libraries_2017.1.132/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/ifortHQ3DJq.o --start-group -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 --end-group -ydgem
m @/usr/common/software/darshan//3.0.1.1/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_2017.1.132/linux/compiler/lib/intel64_lin -limf -lm -lpthread -ldl -lmpich_intel -lrt -lugni -lpthread -lpmi -L/opt/intel/compilers_and_libraries_2017.1.132
/linux/compiler/lib/intel64_lin -limf -lm -ldl -lsma -lpmi -lsma -lpmi -ldmapp -lpthread -lxpmem -ludreg -lpmi -lpthread -lalpslli -lpthread -lwlm_detect -lugni -lpthr
ead -lalpsutil -lpthread -lrca --as-needed -limf --no-as-needed --as-needed -lm --no-as-needed --as-needed -lpthread --no-as-needed --start-group -lmkl_intel_lp64 -lmk
l_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_6
4-suse-linux/4.8/crtend.o /usr/lib64/gcc/x86_64-suse-linux/4.8/../../../../lib64/crtn.o
/opt/intel/compilers_and_libraries_2017.1.132/linux/mkl/lib/intel64/libmkl_intel_lp64.a(_dgemm_lp64.o): definition of dgemm
/opt/intel/compilers_and_libraries_2017.1.132/linux/mkl/lib/intel64/libmkl_core.a(mkl_memory_patched.o): In function `mkl_serv_set_memory_limit':
mkl_memory.c:(.text+0x5a9): 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**
  - Access via modules, type “module avail” to see the available modules
  - Different builds for different compilers
  - Programming environment can selectively pick the matching libraries to load
- **NERSC also supports many libraries**
  - many of them interact with the Cray compiler wrappers, while some of them do not.
- **Where are the libraries ?**
  - Module avail
  - Module show <module name>

```
zz217@cori06:~> echo $MODULEPATH
/opt/cray/pe/ari/modulefiles:/opt/cray/ari/modulefiles:/opt/cray/pe/craype/2.5.7/modulefiles:/opt/cray/pe/modulefiles:/opt/cray/modulefiles:/opt/modulefiles:/usr/common/software/modulefiles:/usr/syscom/nsg/modulefiles:/usr/syscom/nsg/opt/modulefiles:/usr/common/das/modulefiles:/usr/common/ftg/modulefiles:/opt/cray/craype/default/modulefiles
```

Demo for a few commands

- **Module avail**
- **Module show <module name>**
- **Module load cray-petsc**
- **Module load fftw**
- **Module load cray-netcdf**

- **Intel® Math Kernel Library Link Line Advisor**
  - https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/
  - Learn from compiler verbose output, -mkl={serial,thread,cluster}
- **cdt modules could be useful to recover the previous programming environments**

Resolving missing libraries and unresolved symbols

- **Only shared libraries available**
  - E.g., /usr/bin/ld: cannot find –ldl
  - Try -dynamic
- **Static library linking order, try -Wl,--start-group … -Wl,--end-group, e.g.,**

```
LLIBS    =    -Wl,--start-group $(MKLROOT)/lib/intel64/libmkl_intel_lp64.a \
              $(MKLROOT)/lib/intel64/libmkl_scalapack_lp64.a
$(MKLROOT)/lib/intel64/libmkl_blacs_intelmpi_lp64.a \
              $(MKLROOT)/lib/intel64/libmkl_intel_thread.a $(MKLROOT)/lib/intel64/libmkl_core.a
-Wl,--end-group \
              /usr/common/software/wannier90/1.2/knl/intel/lib/libwannier.a \
              -Wl,-zmuldefs
```

- **Use grep –R and readelf –s |grep <symbol> to search for the unresolved symbols.**

How to compile for Cori KNL

Best: Use compiler options to build for KNL

```
module swap craype-haswell craype-mic-knl
```

- **The loaded craype-* module sets the target that the compiler wrappers (cc, CC, ftn) build for**
  - Eg `-mknl` (GNU compiler), `-hmic-knl` (Cray compiler)
- **craype-haswell is default on login nodes**
- **craype-mic-knl is for KNL nodes**

```
10:51 sleak@cori11:~$ module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.7                    13) xpmem/0.1-4.5
  2) nsg/1.2.0                          14) job/1.5.5-3.58
  3) modules/3.2.10.4                   15) dvs/2.7_0.9.0-2.201
  4) intel/16.0.3.210                   16) alps/6.1.3-17.12
  5) craype-network-aries               17) rca/1.0.0-6.21
  6) craype/2.5.5                       18) atp/2.0.2
  7) cray-libsci/16.06.1                19) PrgEnv-intel/6.0.3
  8) udreg/2.3.2-4.6                    20) craype-haswell
  9) ugni/6.0.12-2.1                    21) cray-shmem/7.4.0
 10) pmi/5.0.10-1.0000.11050.0.0.ari    22) cray-mpich/7.4.0
 11) dmapp/7.1.0-12.37                  23) altd/2.0
 12) gni-headers/5.0.7-3.1
```

```
10:51 sleak@cori11:~$ module avail craype

------------------------ /opt/cray/pe/craype/2.5.5/modulefiles ----
craype-accel-host      craype-hugepages256M  craype-intel-knc
craype-accel-nvidia20  craype-hugepages2M    craype-ivybridge
craype-accel-nvidia35  craype-hugepages32M   craype-mic-knl
craype-broadwell       craype-hugepages4M    craype-network-aries
craype-haswell         craype-hugepages512M  craype-network-none
craype-hugepages128M   craype-hugepages64M   craype-sandybridge
craype-hugepages16M    craype-hugepages8M
```

How to compile for Cori KNL

Best: Compiler settings to target KNL

Alternate:

```
CC -axMIC-AVX512,CORE-AVX2 <more-options> mycode.c++
```

- **Only valid when using Intel compilers (cc, CC or ftn)**
- **-ax<arch> adds an "alternate execution paths" optimized for different architectures**
  - Makes 2 (or more) versions of code in same object file
- **NOT AS GOOD as the craype-mic-knl module**
  - (module causes versions of libraries built for that architecture to be used - eg MKL)

How to compile for Cori KNL

Recommendations:

- **For best performance, use the craype-mic-knl module**

```
module swap craype-haswell craype-mic-knl
CC -O3 -c myfile.c++
```

- **If the same executable must run on KNL *and* Haswell nodes, use craype-haswell but add KNL-optimized execution path**

```
CC -axMIC-AVX512,CORE-AVX2 -O3 -c myfile.c++
```

What to link

Utility libraries

- **Not performance-critical (by definition)**
  - KNL can run Xeon binaries .. can use Haswell-targeted versions
- **I/O libraries (HDF5, NetCDF, etc) should fit in this category too**
  - (for Cray-provided libraries, compiler wrapper will use craype-* to select best build anyway)

What to link

Performance-critical libraries

- **MKL: has KNL-targeted optimizations**
- **PETsc, SLEPc, Caffe, Metis, etc:**
  - (soon) has KNL-targeted builds
- **Modulefiles will use craype-{haswell,mic-knl} to find appropriate library**
  - Currently FFTW,LibSci,Petsc, TPSL have separate builds for KNL
- **Key points:**
  - Someone else has already prepared libraries for KNL
  - No need to do-it-yourself
  - Load the right craype- module

Summary

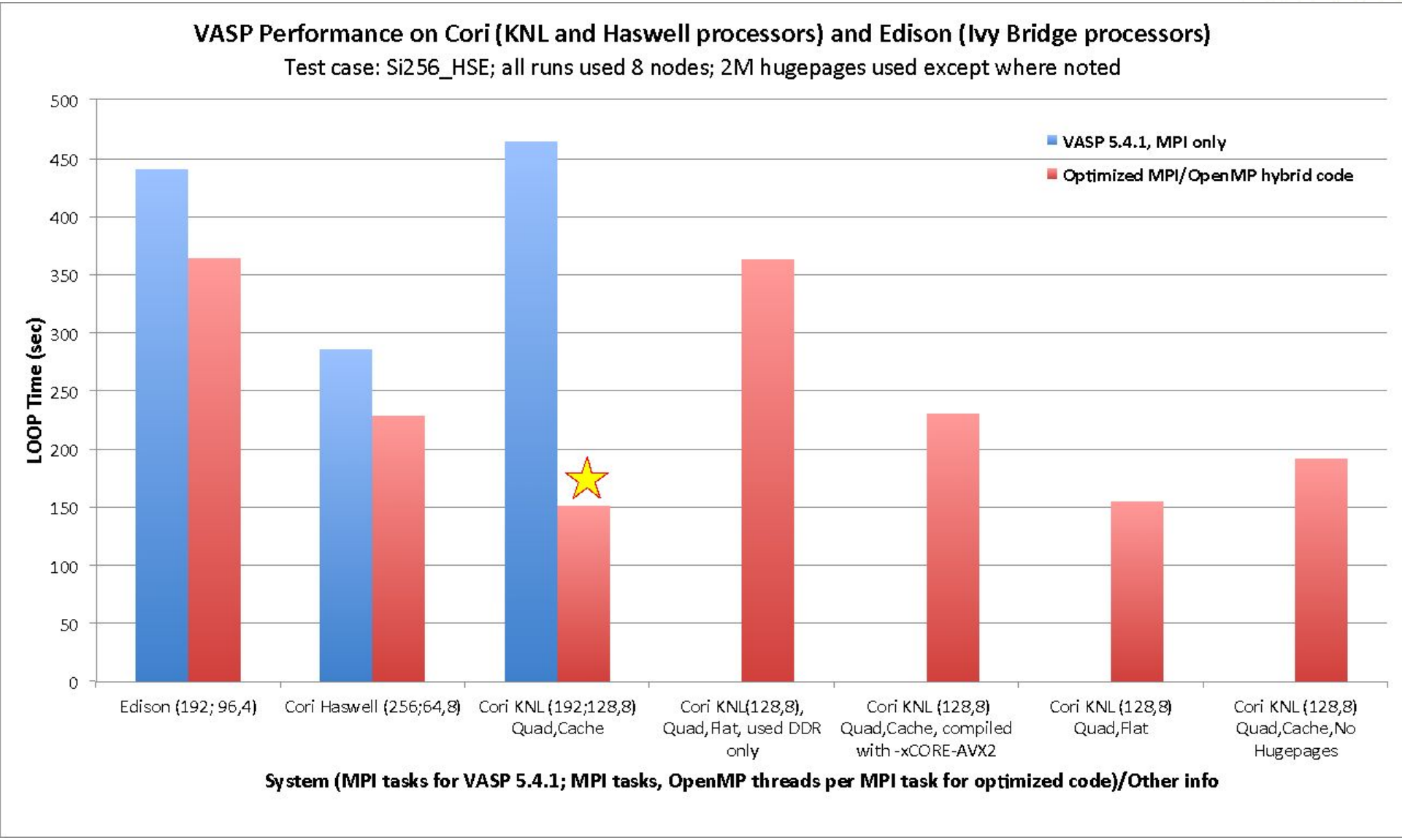
- **Use compiler wrappers where possible, which allows to include architecture specific optimization flag and link to the libraries**
  - Check for the –xHost option
- **Explore available compilers, Intel, Cray, GNU**
  - Validity check after builds is important
- **Use provided libraries where available**
  - Use module show <librarymodule> to see the paths if needed

Summary -continued

- **For Cori KNL, do**
  - `module swap craype-haswell craype-mic-knl`
- **Build on login nodes whenever possible**
- **Learn from the compiler verbose output**
- **Read compiler man pages**

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/
- **man pages:**
  - ifort, icc, icpc, crayftn,etc
  - man ld (-Wl,-zmuldefs, -Wl,-y<symbol>



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

Acknowledgement

- Steve Leak for providing some of the slides used in this talk.

Thank you!