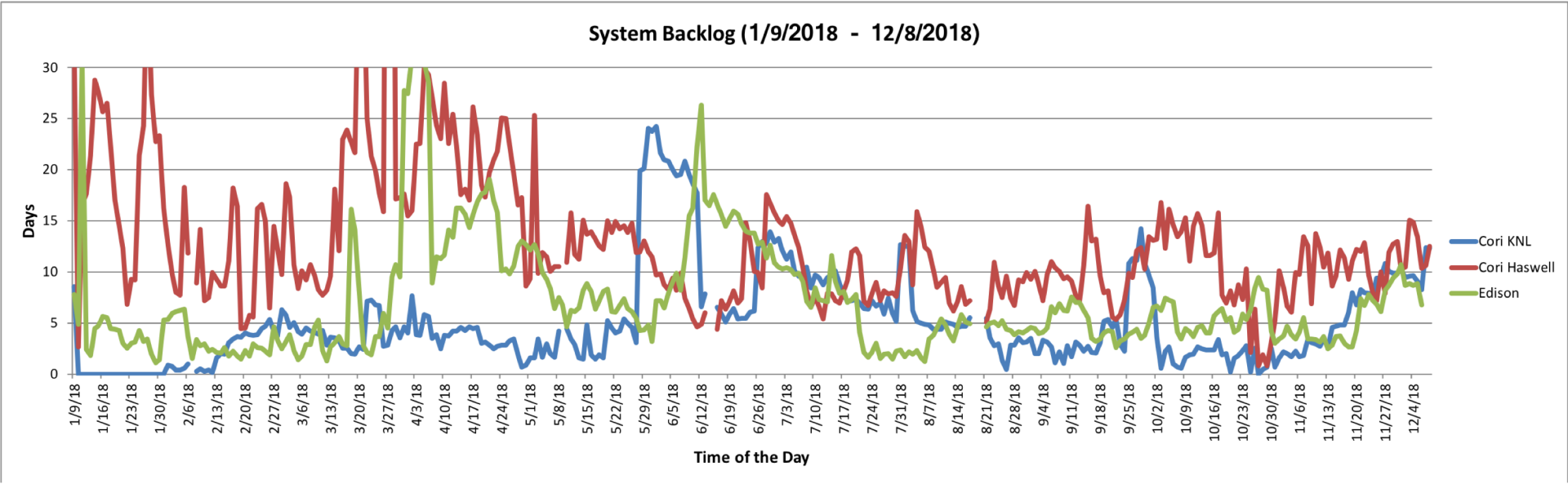
Introduction - The basics of compiling and running on KNL

Zhengji Zhao
User Engagement Group
Cori KNL User Training
April 16, 2019

System Backlogs



Cori KNL has a shorter backlog, so for a better queue turnaround, we recommend the Edison/Cori Haswell users transit to KNL.

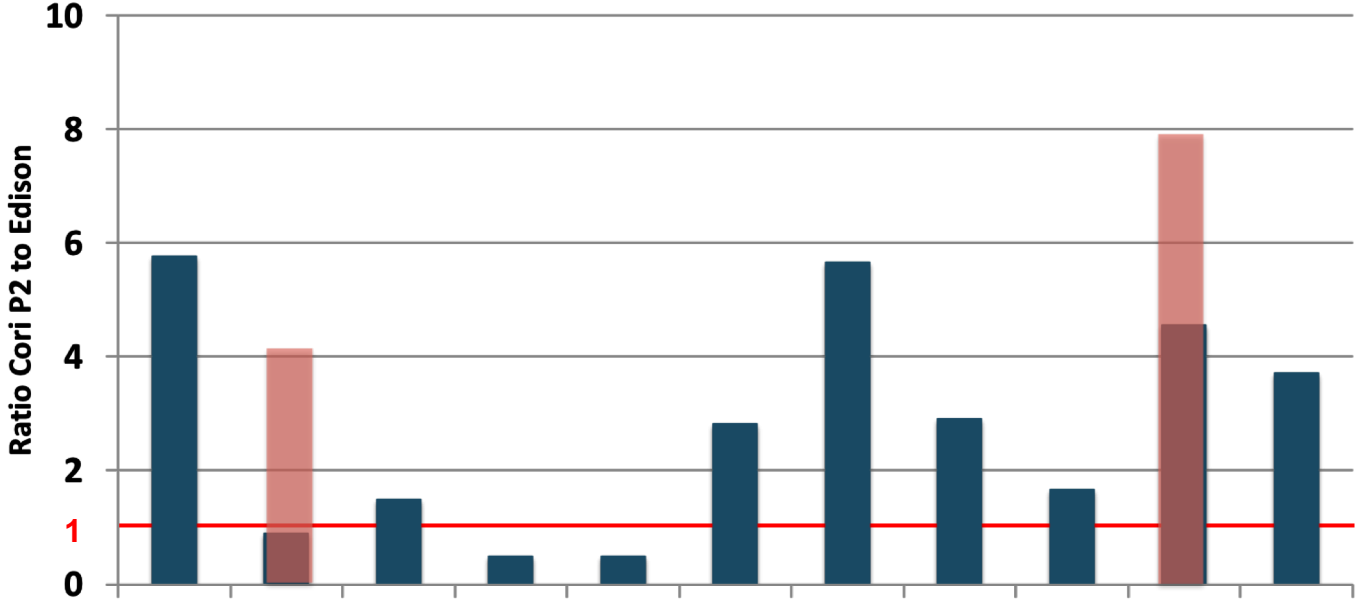
Agenda

- Difference between Edison/Cori Haswell (multi-core) and Cori KNL (many-core)
- How to compile for KNL
- How to run on KNL nodes

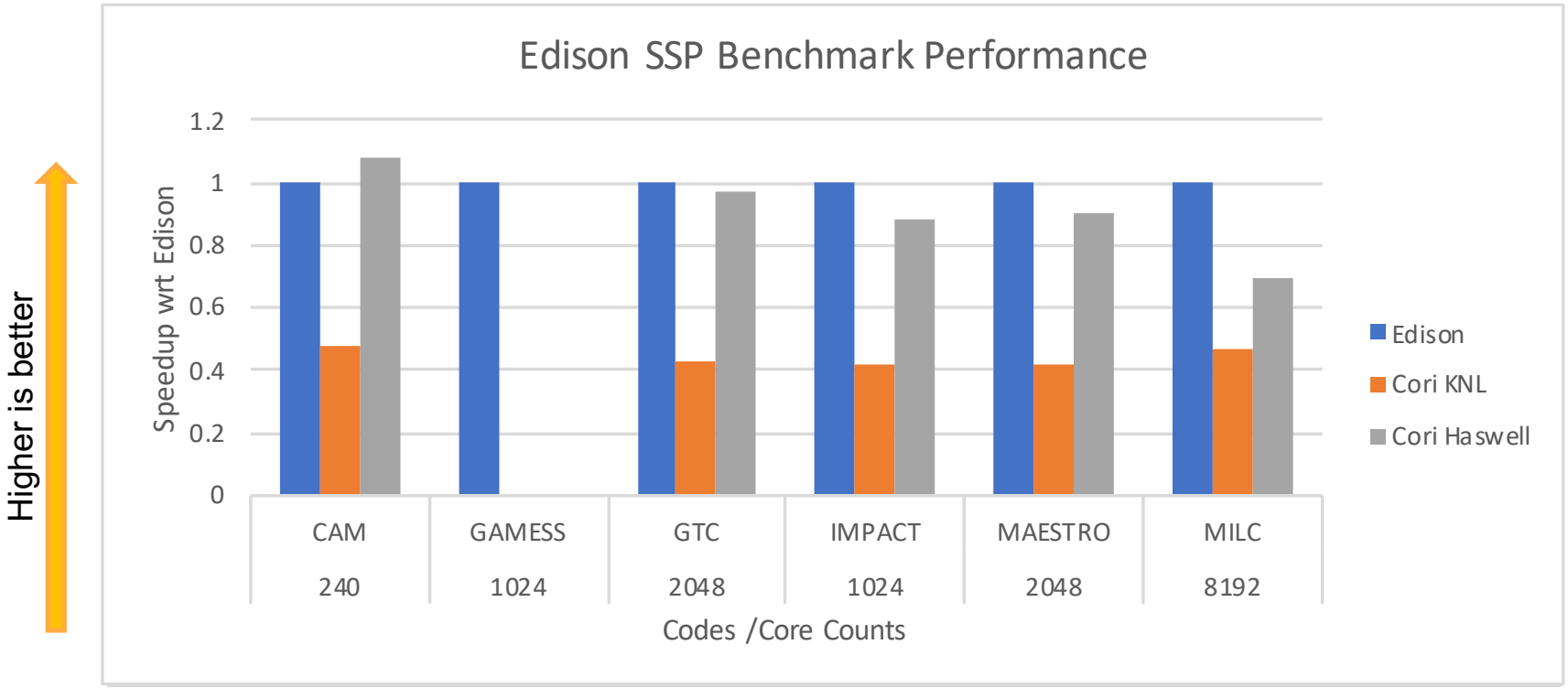
- Variable-time jobs – a way to improve your queue turnaround

Difference between Edison/Cori Haswell and Cori KNL

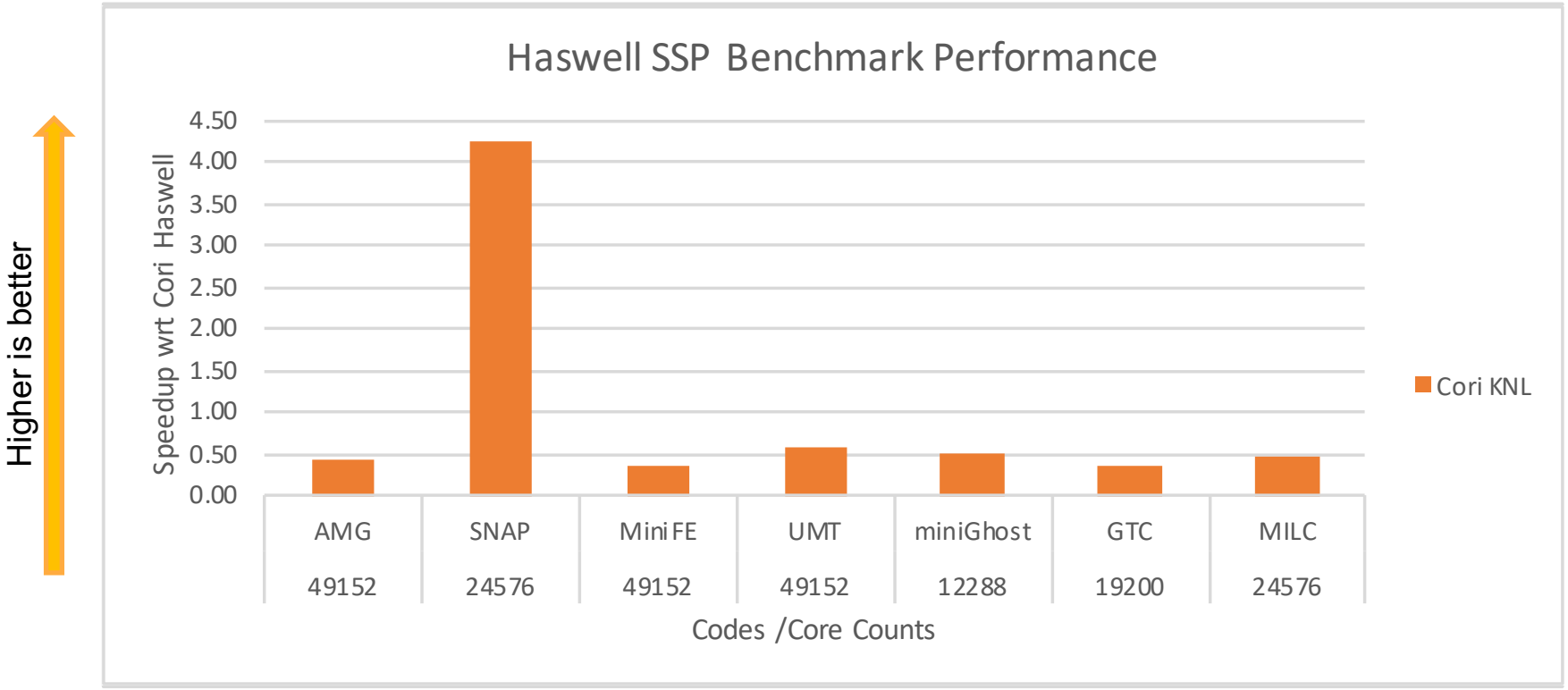
Configuration comparison between Edison and Cori KNL



	Cores/node	Threads/node	Socket/Node	Memory (DDR + HBM)/node (GB)	Memory/Core (GB)	Clock speed (GHz)
Edison	24	48	2	64	2.67	2.4
Cori KNL	68	272	1	96 + 16	1.4 +0.24	1.4
Cori Haswell	32	64	2	128	4.0	2.3



Core-to-core performance comparison

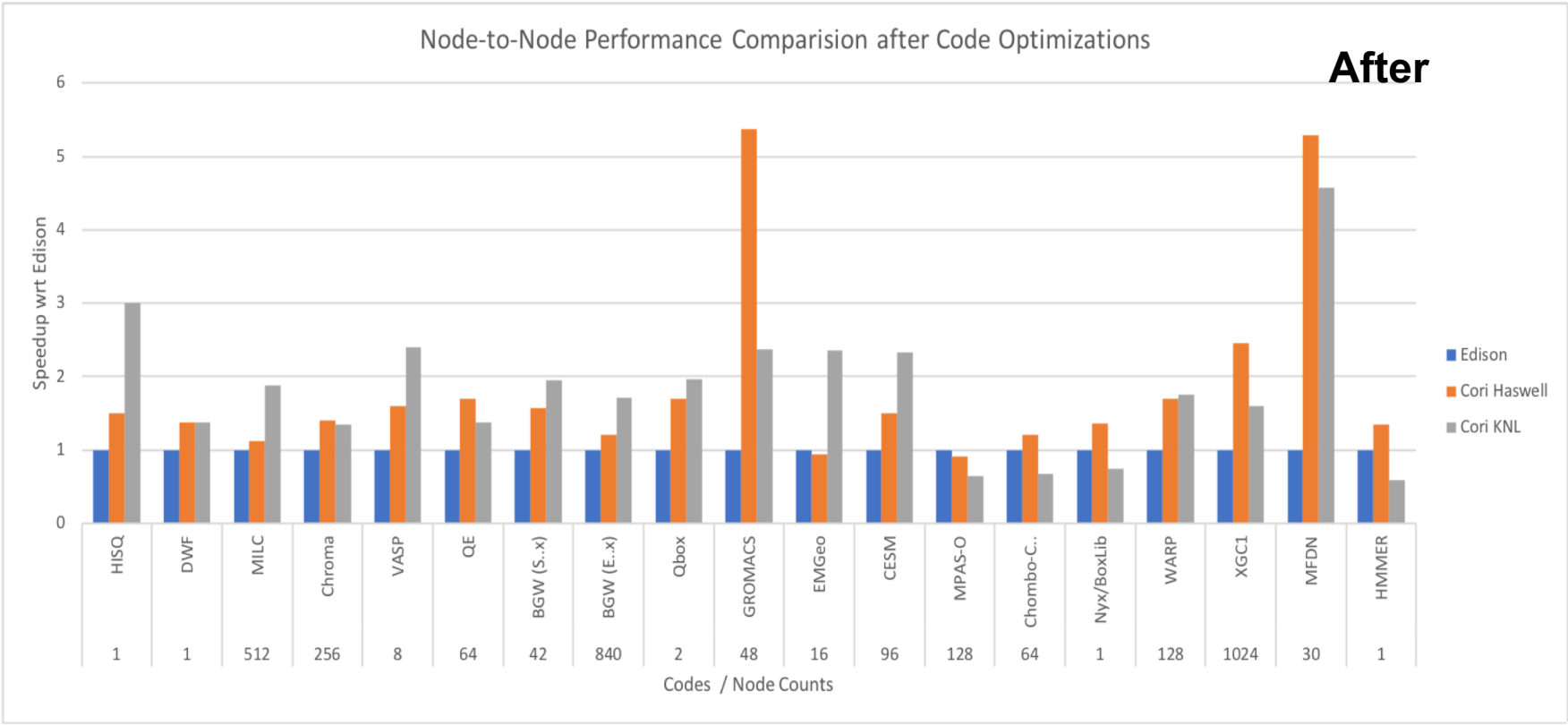


Core-to-core performance comparison

Slide from Helen He

NESAP Code Performance before/after Optimizations

Higher is better



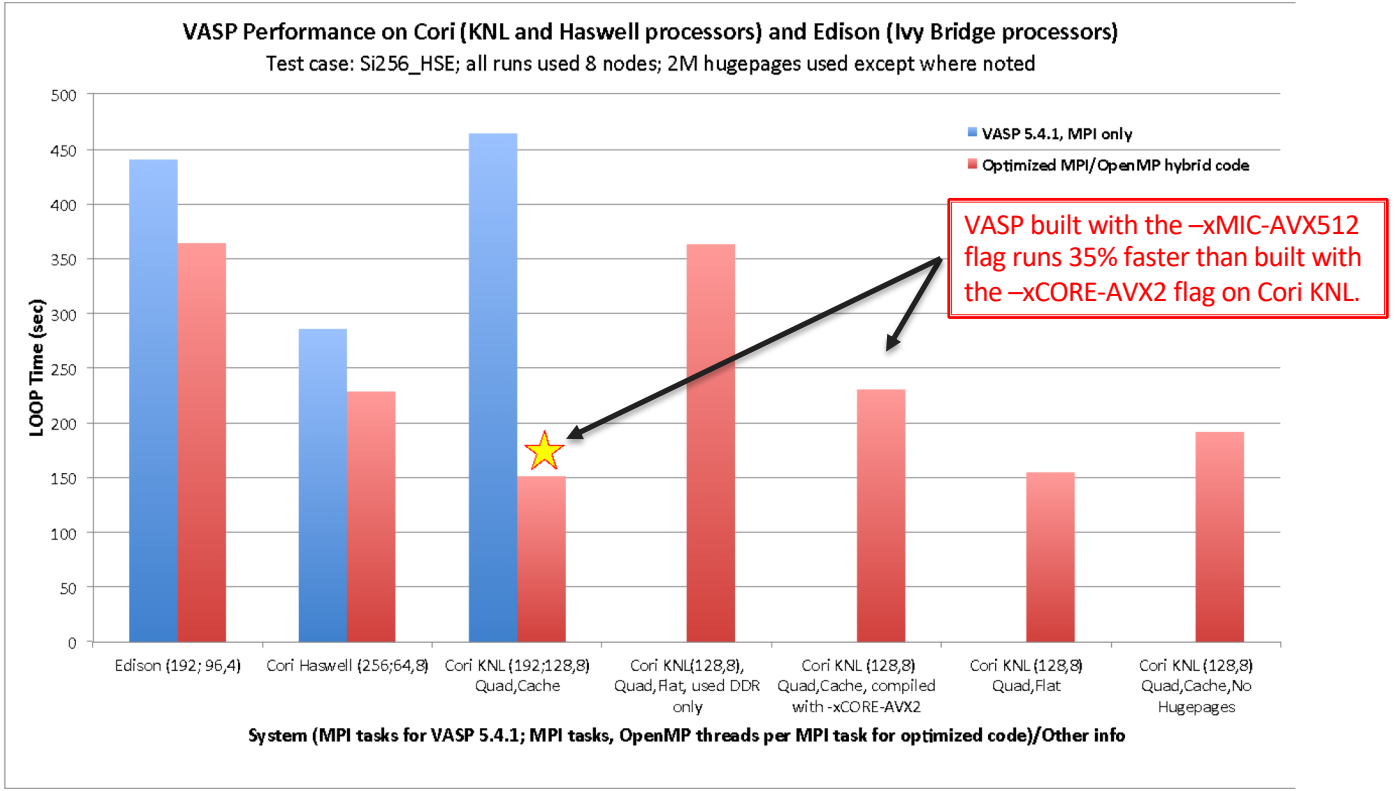
How to compile for KNL

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
- Not vice versa

A separate build of your application for each platform is recommended for optimal performance



We will talk about only

- Compilation for Cori KNL nodes
- Compile/link lines

  Compiler +

  Compiler Flags +

  –I/path/to/headers +

  –L/path/to/library –l<library>

- Available compilers, libraries, etc.

You need to apply these info to your build systems.

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 -cont

- Cross compiling: applications are compiled for compute nodes from the login nodes. Cori has two types of compute nodes, KNL, and Haswell
- Cori default environment loads craype-haswell module, which sets the env CRAY_CPU_TARGET to haswell

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:~>
```

To compile for KNL

- Do module swap craype-haswell craype-mic-knl before compiling for Cori KNL nodes, then use the Cray provided compiler wrappers instead of the native compiler invocations

```
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 the 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

- May need to use the –host=x86_64 configure option (if supported) to help the configure script to skip compiler tests.
- 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 long 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 compile/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.

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

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 on other non-cray systems

```
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

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.

Liking example

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, 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)

How to Run jobs on KNL

Cori KNL Queue Policy

QOS	Max nodes	Max time (hrs)	Submit limit	Run limit	Priority	Charge
regular	9489	48	5000	-	4	90 [6]
interactive [4]	64	4	1	1	-	90
debug	512	0.5	5	2	3	90
premium	9489	48	5	-	2	180 [6]
low	9489	48	5000	-	5	45 [7]
scavenger [2]	9489	48 [8]	5000	-	6	0
special [5]	custom	custom	custom	custom	-	custom

- Jobs use 1024+ nodes on Cori KNL get 20% charging discount
- "interactive" qos available on Cori Haswell and KNL, job starts immediately or get canceled in 5 minutes, up to 64 nodes on Cori per repo

Difference between Edison/Haswell and Cori KNL

	Cores/node	Threads/node	Socket/Node	Memory (DDR + HBM)/node (GB)	Memory/Core (GB)	Clock speed (GHz)
Edison	24	48	2	64	2.67	2.4
Cori Haswell	32	64	2	128	4.0	2.3
Cori KNL	68	272	1	96 + 16	1.4 +0.24	1.4

- KNL has a lot more (slower) cores on the node
- A much reduced per core memory

Interactive batch job on KNL nodes

Edison:

salloc -N 2 –q debug –t 30:00

Cori KNL:

salloc -N 2 –q debug –t 30:00 -C knl

salloc -N 2 –q interactive –t 4:00:00 -C knl

Use of interactive queue is highly recommended!

Sample job script to run a MPI job

Edison:

```
#!/bin/bash -l
#SBATCH –N 2
#SBATCH –q regular
#SBATCH –t 1:00:00
#SBATCH –L SCRATCH

srun –n 48 ./a.out
#or srun –n48 –c2 --cpu_bind=cores ./a.out
```

Cori KNL:

```
#!/bin/bash -l
#SBATCH –N 1
#SBATCH -C knl
#SBATCH –q regular
#SBATCH –t 1:00:00
#SBATCH –L SCRATCH

srun –n68 -c4 --cpu_bind=cores ./a.out
```

Sample job script to run an MPI + OpenMP code

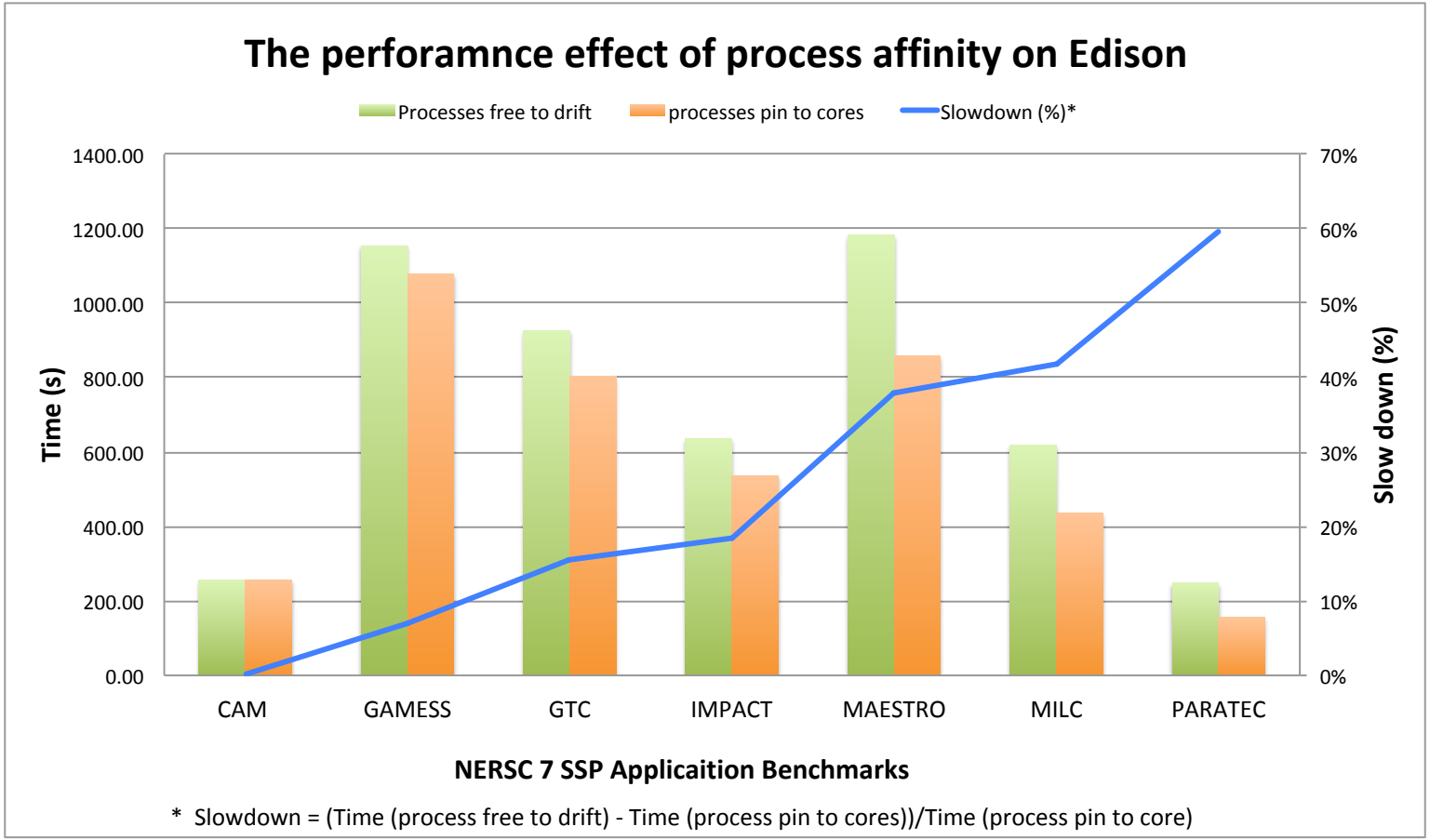
```
#!/bin/bash -l
#SBATCH –N 1
#SBATCH –q regular
#SBATCH –t 1:00:00
#SBATCH -C knl

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4

# launching 1 task every 4 cores/16 CPUs
srun –n16 –c16 --cpu_bind=cores ./a.out
```

```
#!/bin/bash -l
#SBATCH –N 1
#SBATCH –q regular
#SBATCH –t 1:00:00
#SBATCH -C knl

export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=4

# launching 1 task every 2 cores/8 CPUs
srun –n32 –c8 --cpu_bind=cores ./a.out
```

- Using –c option to spread processes evenly over on the CPUs on the node
- Using –cpu_bind=cores to pin the processes to the cores on the node
- Using OMP environment variables to fine control the thread affinity

In the examples above, 64 cores /256 CPUs out of 68 cores/272 CPUs are used.

Process affinity is important to get optimal performance



Run date: July 2017

Affinity verification methods

- NERSC has provided pre-built binaries from a Cray code (xthi.c) to display process thread affinity: check-mpi.intel.cori, check-mpi.cray.cori, check-hybrid.intel.cori, etc.

```
% srun -n 32 -c 8 --cpu_bind=cores check-mpi.intel.cori|sort -nk 4
Hello from rank 0, on nid02305. (core affinity = 0,1,68,69,136,137,204,205)
Hello from rank 1, on nid02305. (core affinity = 2,3,70,71,138,139,206,207)
Hello from rank 2, on nid02305. (core affinity = 4,5,72,73,140,141,208,209)
Hello from rank 3, on nid02305. (core affinity = 6,7,74,75,142,143,210,211)
```

- Intel compiler has a run time environment variable KMP_AFFINITY, when set to "verbose”:

```
OMP: Info #242: KMP_AFFINITY: pid 255705 thread 0 bound to OS proc set {55}
OMP: Info #242: KMP_AFFINITY: pid 255660 thread 1 bound to OS proc set {10,78}
OMP: Info #242: OMP_PROC_BIND: pid 255660 thread 1 bound to OS proc set {78} …
```

- Cray compiler has a similar env CRAY_OMP_CHECK_AFFINITY, when set to "TRUE”:

```
[CCE OMP: host=nid00033 pid=14506 tid=17606 id=1] thread 1 affinity:  90
[CCE OMP: host=nid00033 pid=14510 tid=17597 id=1] thread 1 affinity:  94  …
```

Slide from Helen He

A few useful commands

- Commonly used commands: sbatch,salloc,scancel,srun,squeue,sinfo,sqs,scontrol,sacct
- sinfo –format="%F %b" for available features of nodes, or sinfo –format="%C %b"
    - A/I/O/T (allocated/idle/other/total)
- scontrol show node <nid>
- ssh_job <jobid> to ssh to the head compute nodes of your running jobs

Summary

- The “-C knl” is used to request KNL nodes
- Recommend explicitly use of the srun’s –cpu-bind and -c options to pin the processes to the cores/CPUs, and spread the MPI tasks evenly over the cores/CPUs on the nodes
- Use OpenMP envs, OMP_PROC_BIND and OMP_PLACES to fine pin threads to the CPUs allocated to the tasks
- Consider using 64 cores out of 68 in most cases
- The interactive queue is highly recommended
- Submit shorter jobs for a better queue turnaround. Use variable-time jobs automatically split a long running job to multiple shorter ones.


NERSC
Thank You!

Variable-time jobs

to improve the queue turnaround

Who is relevant to variable-time jobs?

- Users who want to a improved the queue turnaround
- Users who need to run long jobs, including jobs running for more than 48 hours - the max time allowed on Cori and Edison.

- Provided the code can do checkpointing by itself

Variable-time jobs

- Slurm allows jobs submitted with a minimum time limit in addition to the time limit, e.g.,

  #SBATCH –time=48:00:00

  #SBATCH –time-min=2:00:00

- Jobs specified the --time-min can start the execution earlier than they would otherwise with a time limit anywhere between the time-min and the time limit.
  - This is performed by a backfill scheduling algorithm to allocate resources otherwise reserved for higher priority jobs.

Variable-time jobs - continued

- The pre-terminated job can be requeued to resume from where the previous execution left off.
  - #SBATCH –requeue
  - scontrol requeue <jobid>
- Requeuing the pre-terminated job can be done automatically until the cumulative execution time reaches the requested time limit or the job completes earlier before the requested time limit.

https://docs.nersc.gov/jobs/examples/#variable-time-jobs

Sample job script for variable-time jobs

```
#!/bin/bash
#SBATCH -J test
#SBATCH -q regular
#SBATCH -C haswell
#SBATCH -N 1
#SBATCH --comment=64:00:00
#SBATCH --time-min=00:30:00    #the minimum amount of time the job should run
#SBATCH --time=48:00:00
#SBATCH --error=test-%j.err
#SBATCH --output=test-%j.out
#
#SBATCH --signal=B:USR1@120
#SBATCH --requeue
#SBATCH --open-mode=append

#timelimit per job, the amount of time (in seconds) needed for checkpointing (same as in --signal)
#and the checkpont command if any
max_timelimit=48:00:00    #if not set default to 48:00:00
ckpt_overhead=120         #if not set, default to 60 seconds
ckpt_command=

# requeueing the job if remaining time >0
. /global/common/cori/software/variable-time-job/setup.sh
requeue_job func_trap USR1


#user setting
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=1

#srun must execute in background and catch signal on wait command
srun -n 1 -c 64 --cpu_bind=cores ./a.out &

wait
```

Provide ckpt-command only if your application needs external trigger to initiate the checkpointing. Leave blank if none

Variable-time script for CP2K

```
#!/bin/bash -l
#SBATCH -q regular
#SBATCH -N 1
#SBATCH -C knl
#SBATCH -J md
#SBATCH --comment=96:00:00
#SBATCH --time-min=00:30:00   #the minimum amount of time the job should run
#SBATCH --time=48:00:00
#SBATCH --signal=B:USR1@60
#SBATCH --requeue
#SBATCH --open-mode=append

#timelimit per job, and the amount of time (in seconds) needed for checkpointing (same as in
--signal)
max_timelimit=48:00:00
ckpt_overhead=60
ckpt_command=

#requeueing the job if remaining time >0
. /global/common/cori/software/variable-time-job/setup.sh
requeue_job func_trap USR1

module load cp2k
srun -n 68 ./cp2k.popt run.inp >> run.out &

wait
```

Variable-time script for VASP atomic relaxation jobs

```
#!/bin/bash
#SBATCH -J ata_vasp
#SBATCH -q regular
#SBATCH -C knl
#SBATCH -N 2
#SBATCH --time=48:0:00
#SBATCH --error=ata-%j.err
#SBATCH --output=ata-%j.out
#SBATCH --mail-user=zz217@nersc.gov
#
#SBATCH --comment=96:00:00
#SBATCH --time-min=02:0:00
#SBATCH --signal=B:USR1@300
#SBATCH --requeue
#SBATCH --open-mode=append

#user setting
export OMP_PROC_BIND=true
export OMP_PLACES=threads
export OMP_NUM_THREADS=8

#srun must execute in background and catch signal on wait command
module load vasp/20171017-knl
srun -n 8 -c32 --cpu_bind=cores vasp_std &
```

```
# put any commands that need to run to continue the next job (fragment) here
ckpt_vasp() {
   set -x
   restarts=`squeue -h -O restartcnt -j $SLURM_JOB_ID`
   echo checkpointing the ${restarts}-th job

   #to terminate VASP at the next ionic step
   echo LSTOP = .TRUE. > STOPCAR
   #wait until VASP to complete the current ionic step, write out WAVECAR file and quit
   srun_pid=`ps -fle|grep srun|head -1|awk '{print $4}'`
   echo srun pid is $srun_pid
   wait $srun_pid

   #copy CONTCAR to POSCAR
   cp -p CONTCAR POSCAR
   set +x
}

ckpt_command=ckpt_vasp
max_timelimit=48:00:00
ckpt_overhead=300

# requeueing the job if remaining time >0
. /global/common/cori/software/variable-time-job/setup.sh
requeue_job func_trap USR1

wait
```

More information

- 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/
  - Transitioning to NERSC Docs: 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/
  - Cori KNL: http://www.nersc.gov/users/computational-systems/cori/running-jobs/example-batch-scripts-for-knl/
  - Transitioning to NERSC Docs: https://docs.nersc.gov/jobs/
- Compiler and linker man pages:
  - ifort, icc, icpc, crayftn, etc.
  - man ld (-Wl,-zmuldefs, -Wl,-y<symbol>)
- Contact NERSC Consulting:
  - Call at 800-666-3772 or 510-486-8600, option #3
  - File consulting tickets at help.nersc.gov or https://my.nersc.gov/tickets.php

Compute node reservations for hands-on

- We have 6 separate reservations under repo nintern:

  Apr 16: noon - 5pm, 256 KNL nodes (ReservationName: knl_apr16);
  32 Haswell nodes (ReservationName: hsw_apr16)
  Apr 17: noon - 5pm, 256 KNL nodes (ReservationName: knl_apr17);
  32 Haswell nodes (ReservationName: hsw_apr17)
  Apr 18: 9am - 5pm, 128 KNL nodes (ReservationName: knl_apr18)
  16 Haswell nodes (ReservationName: hsw_apr18)

- Use "--reservation=knl_apr16 -A nintern" with sbatch or salloc to use the reservation and also charge to the nintern repo instead of your own.
- Use the interactive queue if all reserved nodes are used.
- Use squeue –A nintern or squeue –R <ReservationName> to check jobs under a repo or a reservation, respectively.