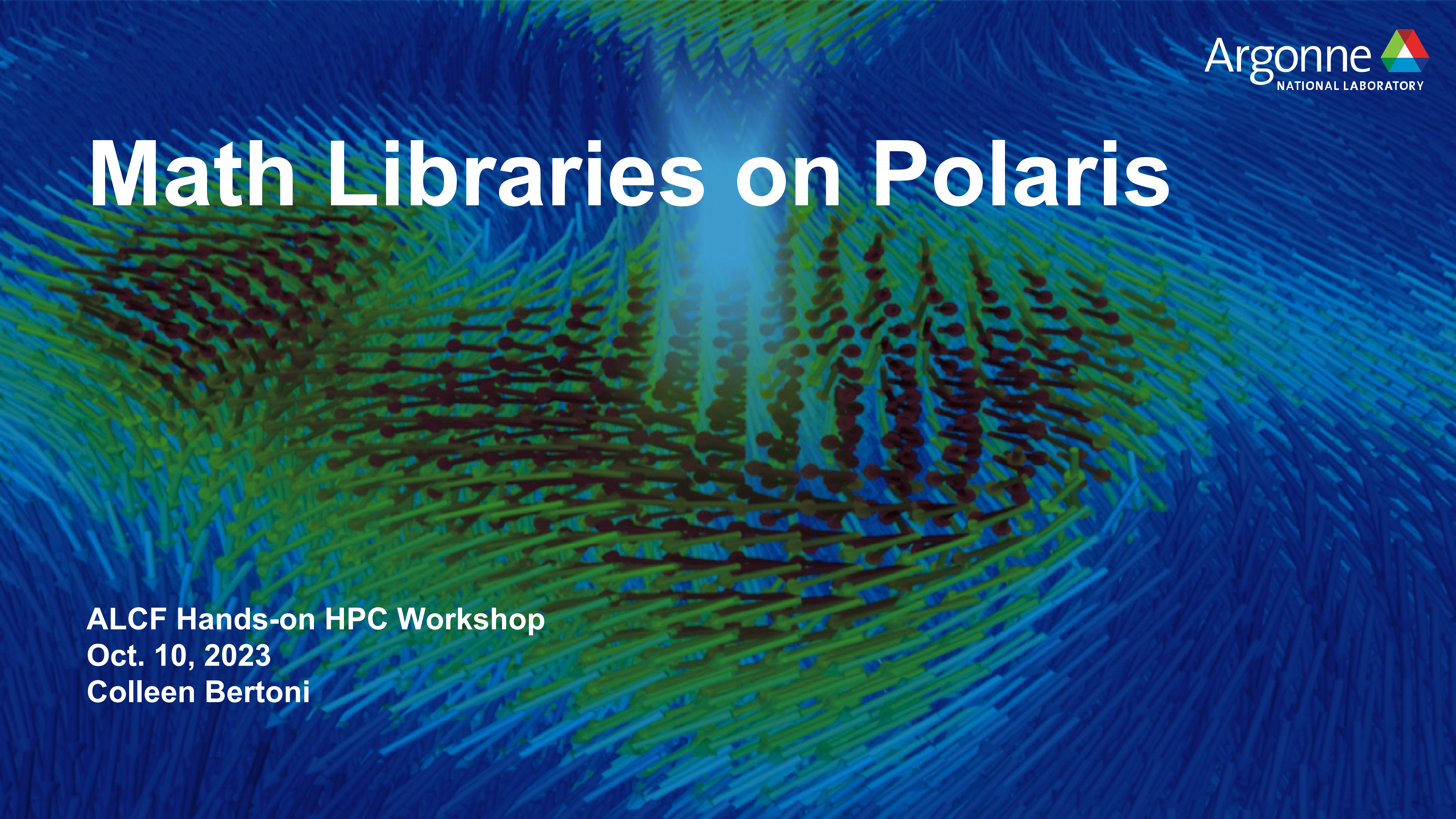
October 10-12, 2023

ALCF Hands-on HPC Workshop



Math Libraries on Polaris

ALCF Hands-on HPC Workshop
Oct. 10, 2023
Colleen Bertoni

Outline

- Getting Started (modules and compiler wrappers)
- Available Math Libraries (and how to find them)
- Sanity checking
- Debugging Examples / Game ☺

Getting Started

Getting Started: Modules

- Environments handled by modules on Polaris

```
> module list
Currently Loaded Modules:
  1) craype-x86-rome          6) craype/2.7.15          11) cray-pals/1.1.7
  2) libfabric/1.11.0.4.125   7) cray-dsmml/0.2.2       12) cray-libpals/1.1.7
  3) craype-network-ofi       8) cray-mpich/8.1.16      13) PrgEnv-nvhpc/8.3.3
  4) perftools-base/22.05.0   9) cray-pmi/6.1.2         14) craype-accel-nvidia80
  5) nvhpc/21.9              10) cray-pmi-lib/6.0.17
```

- These set available programs by PATH, INCLUDE_PATH, LD_LIBRARY_PATH, etc.
- Recommend always checking what it is when you compile and run

Getting Started: Modules

- Compilers:
  - gcc, nvc++, nvfortran, nvc, etc.
- Compiler Wrappers:
  - **CC: C++ compiler wrapper**
  - **cc: C compiler wrapper**
  - **ftn: Fortran compiler wrapper**
- Wrappers which select a specific underlying compiler based on the PrgEv loaded (so can refer to any underlying compiler on the system)

Getting Started: Modules

- Compilers:
  - gcc, nvc++, nvfortran, nvc, etc.
- Compiler Wrappers:
  - **CC: C++ compiler wrapper**
  - **cc: C compiler wrapper**
  - **ftn: Fortran compiler wrapper**
- Wrappers which select a specific underlying compiler based on the PrgEv loaded (so can refer to any underlying compiler on the system)

```
> cc --cray-print-opts=libs
-L/opt/cray/pe/mpich/8.1.16/ofi/nvidia/20.7/lib -L/opt/cray/pe/mpich/8.1.16/gtl/lib -
L/opt/cray/pe/pals/1.1.7/lib -L/opt/cray/pe/pmi/6.1.2/lib -
L/opt/cray/pe/dsmml/0.2.2/dsmml//lib -Wl,--as-needed,-lpals,--no-as-needed -Wl,--as-
needed,-lpmi,--no-as-needed -Wl,--as-needed,-lpmi2,--no-as-needed -Wl,--as-needed,-
lmpi_nvidia,--no-as-needed -lmpi_gtl_cuda -Wl,--as-needed,-ldsmml,--no-as-needed
```

Available Math Libraries

Available Math Libraries

- Like compilers, libraries are also managed by modules on Polaris
  - The modules work by setting PATH, LD_LIBRARY_PATH, etc. in your environment to point to the right library
- **Dynamic linking by default**
  - This means that by default, the needed library routines are located and loaded at runtime, not compile time (like static libraries).
    - So you'll need to be careful of the environment at runtime as well to be sure the libraries you want are available in LD_LIBRARY_PATH
  - Still need things available at compile time:
    - At compile time, the linker verifies that all symbols are either linked into the program or are in one of the shared libraries. But the definitions from the dynamic library are not inserted into the executable
    - At runtime, a dynamic loader checks which shared libraries were linked in, and attaches them to the executable

Available Math Libraries

- Like compilers, libraries are also managed by modules on Polaris
  - The modules work by setting PATH, LD_LIBRARY_PATH, etc. in your environment to point to the right library
- **Dynamic linking by default**
  - This means that by default, the needed library routines are located and loaded at runtime, not compile time (like static libraries).
    - So you'll need to be careful of the environment at runtime as well to be sure the libraries you want are available in LD_LIBRARY_PATH

- The NVIDIA-provided libraries are usually linked in with –rpath by default, so it will load the version you compiled with regardless of the current environment.

- You can compile with ”-v” to confirm

Available Math Libraries

- Like compilers, libraries are also managed by modules on Polaris

- “module load $library” to set the environment so that the library is pulled in

```
> module load cray-libsci
```

- “module show $library” shows you what “module load” is doing

```
> module show cray-libsci
-----------------------------------------------------------------------------------------------
  /opt/cray/pe/lmod/modulefiles/core/cray-libsci/21.08.1.2.lua:
-----------------------------------------------------------------------------------------------
help([[See /opt/cray/pe/libsci/21.08.1.2/release_info for cray-libsci release info.]])
whatis("Setup for Cray PE driver set and targeting modules.")
setenv("PE_LIBSCI_FIXED_PRGENV","CRAYCLANG")
prepend_path("PE_CRAYCLANG_FIXED_PKGCONFIG_PATH","/opt/cray/pe/libsci/21.08.1.2/CRAY/9.0/x86_64/lib/pkgconfig")
setenv("CRAY_LIBSCI_DIR","/opt/cray/pe/libsci/21.08.1.2")
setenv("CRAY_LIBSCI_BASE_DIR","/opt/cray/pe/libsci/21.08.1.2")
setenv("LIBSCI_BASE_DIR","/opt/cray/pe/libsci/21.08.1.2")
setenv("LIBSCI_VERSION","21.08.1.2")
setenv("CRAY_LIBSCI_VERSION","21.08.1.2")
prepend_path("PE_PRODUCT_LIST","CRAY_LIBSCI")
prepend_path("MANPATH","/opt/cray/pe/libsci/21.08.1.2/man:/opt/cray/pe/man/csmlversion")
setenv("CRAY_LIBSCI_PREFIX_DIR","/opt/cray/pe/libsci/21.08.1.2/NVIDIA/20.7/x86_64")
prepend_path("CRAY_LD_LIBRARY_PATH","/opt/cray/pe/libsci/21.08.1.2/NVIDIA/20.7/x86_64/lib")
setenv("PE_LIBSCI_MODULE_NAME","cray-libsci/21.08.1.2")
```

Available Math Libraries

- You can use “module avail” to see all the modules and libraries available
- GPU (PrgEnv-nvhpc)
  - Available in nvhpc module
  - cuBLAS, cuSOLVER, cuRAND, cuSPARSE, cuFFT, cuFFTW
  - For CUDA code:
    - `> CC –cuda –lcublas -L${NVIDIA_PATH}/math_libs/lib64/ …`
    - `> ftn –cudalib=cublas`
  - Nvidia docs:
    - https://docs.nvidia.com/hpc-sdk/index.html#math-libraries

```
> ls -ltr $NVIDIA_PATH/math_libs/lib64/
libcublas.so -> libcublas.so.11
libcublas.so.11.5.4.8
libcublasLt.so.11 -> libcublasLt.so.11.5.4.8
libcublasLt.so -> libcublasLt.so.11
libcublasLt.so.11.5.4.8
libcublasLt_static.a
libnvblas.so.11.5.4.8
libnvblas.so.11 -> libnvblas.so.11.5.4.8
libnvblas.so -> libnvblas.so.11
libcurand.so.10 -> libcurand.so.10.2.5.100
libcurand.so -> libcurand.so.10
libcublas_static.a
libcurand.so.10.2.5.100
libcusolver.so.11 -> libcusolver.so.11.2.0.100
libcusolver.so -> libcusolver.so.11
libcurand_static.a
libcusolver.so.11.2.0.100
libcusolverMg.so.11 -> libcusolverMg.so.11.2.0.100
libmetis_static.a
liblapack_static.a
libcusparse.so -> libcusparse.so.11
libcusolver_static.a
libcusparse.so.11.6.0.100
libcusparse_static.a
libcutensor.so.1 -> libcutensor.so.1.3.1
libcutensor.so -> libcutensor.so.1
libcutensor_static.a
libcufft.so.10 -> libcufft.so.10.5.2.100
libcufft.so -> libcufft.so.10
libcufft.so.10.5.2.100
libcufft_static.a
libcufft_static_nocallback.a
libcufftw_static.a
libcufftw.so.10.5.2.100
libcufftw.so -> libcufftw.so.10
```

Available Math Libraries

- GPU (PrgEnv-gnu)
  - Available in cudatoolkit-standalone and nvhpc modules (but you’ll need to hardcode the path)
  - cuBLAS, cuSOLVER, cuRAND, cuSPARSE, cuFFT, cuFFTW
  - For CUDA code:
    - `> CC –lcudart –lcublas --I/soft/compilers/cudatoolkit/cuda-11.4.4/include -L/soft/compilers/cudatoolkit/cuda-11.4.4/lib64 …`
  - Nvidia docs:
    - https://docs.nvidia.com/hpc-sdk/index.html#math-libraries

```
> ls -ltr /soft/compilers/cudatoolkit/cuda-11.4.4/lib64/
libnvToolsExt.so.1.0.0
libnvToolsExt.so.1 -> libnvToolsExt.so.1.0.0
libnvToolsExt.so -> libnvToolsExt.so.1
libcuinj64.so.11.4.120
libcuinj64.so.11.4 -> libcuinj64.so.11.4.120
libcuinj64.so -> libcuinj64.so.11.4
libaccinj64.so.11.4.120libcufilt.a
libOpenCL.so.1.0.0
libOpenCL.so.1.0 -> libOpenCL.so.1.0.0
libOpenCL.so.1 -> libOpenCL.so.1.0
libnvrtc.so.11.4.152
libnvrtc.so.11.2 -> libnvrtc.so.11.4.152
libnvrtc-builtins.so.11.4.152
libnvrtc-builtins.so.11.4 -> libnvrtc-builtins.so.11.4.152
libnvptxcompiler_static.a
libcudart.so.11.4.148
libcudart.so.11.0 -> libcudart.so.11.4.148
libcublasLt.so.11.6.5.2
libcublas.so.11 -> libcublas.so.11.6.5.2
libcublasLt.so.11 -> libcublasLt.so.11.6.5.2
libcublas.so.11.6.5.2
libnvblas.so.11.6.5.2
libnvblas.so.11 -> libnvblas.so.11.6.5.2
libcufftw.so.10 -> libcufftw.so.10.5.2.100
libcufft.so.10.5.2.100
libcufft.so.10 -> libcufft.so.10.5.2.100
libcurand.so.10.2.5.120
libcufftw.so.10.5.2.100
libcusolverMg.so.11.2.0.120
libcurand.so.10 -> libcurand.so.10.2.5.120
libcusolverMg.so.11 -> libcusolverMg.so.11.2.0.120
libcusolver.so.11.2.0.120
libcusparse.so.11.6.0.120
libcusolver.so.11 -> libcusolver.so.11.2.0.120
...
```

Available Math Libraries

- CPU
  - Nvidia libraries
    - BLAS & LAPACK can be found in the $NVIDIA_PATH/compilers/lib directory.
    - ScaLAPACK can be found in the $NVIDIA_PATH/comm_libs directory.
    - You'll need to link in with
      - `> -L ${NVIDIA_PATH}/compilers/lib -llapack -lblas`
    - Nvidia docs: https://docs.nvidia.com/hpc-sdk/compilers/hpc-compilers-user-guide/#lib-use-lapack-blas-ffts

Available Math Libraries

- CPU (continued)
  - Cray LibSci
    - BLAS, LAPACK, ScaLAPACK
    - “module load cray-libsci” will set the library paths for you
      - If it’s with the Cray compiler wrappers. Not the case with the other mpi compiler wrappers (e.g. llvm & oneapi)
    - Check which library is linked in:
      - libsci_compiler.so (serial non-threaded)
      - libsci_compiler_mp.so (parallel multi-threaded)
      - libsci_compiler_mpi.so (serial non-threaded, MPI)
      - libsci_compiler_mp_mpi.so (parallel non-threaded, MPI)
    - “man intro_libsci” for more information
  - Cray FFTW
    - module load cray-fftw/3.3.8.13
    - “man intro_fftw3” for more information

Available additional Math libraries

- More non-Cray-supported ones are available under “/soft/modulefiles”:
- module use /soft/modulefiles
  - GSL (https://www.gnu.org/s/gsl/manual/gsl-ref.html)
    - module load gsl
  - MAGMA (https://icl.utk.edu/magma/)
    - module load magma
  - Ginkgo (https://github.com/ginkgo-project/ginkgo)
    - module load ginko
  - AMD Optimizing CPU libraries (https://www.amd.com/en/developer/aocl.html)
    - module load aocl
- E4S (deployed by ParaTools, and only for PrgEnv-gnu)
  - “module load e4s” to see what’s available
- oneMKL (supports onemkl SYCL APIs via cu* math libraries)
  - module load oneapi

Available Math Libraries

- A little more info:
  - https://docs.alcf.anl.gov/polaris/applications-and-libraries/libraries/math-libraries/

Sanity Checking

Sanity checking

- **Dynamic linking by default**
- ldd ./exe: prints the shared objects (shared libraries) required by the exe
  - Check that you're linked to what you think you are!

```
> ldd gamess.00.x
linux-vdso.so.1 (0x000014ecd289d000)
libmpi_nvidia.so.12 => /opt/cray/pe/lib64/libmpi_nvidia.so.12 (0x000014eccfc56000)
libsci_nvidia_mp.so.5.0 => /opt/cray/pe/lib64/libsci_nvidia_mp.so.5.0 (0x000014eccc1d8000)
libdl.so.2 => /lib64/libdl.so.2 (0x000014eccc1d3000)
libmpi_gtl_cuda.so.0 => /opt/cray/pe/lib64/libmpi_gtl_cuda.so.0 (0x000014eccbfbe000)
libcusolver.so.11 => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/math_libs/11.8/lib64/libcusolver.so.11 (0x000014ecb9d06000)
libcublas.so.11 => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/math_libs/11.8/lib64/libcublas.so.11 (0x000014ecb40a8000)
libcublasLt.so.11 => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/math_libs/11.8/lib64/libcublasLt.so.11 (0x000014ec8fb22000)
libcudaforwrapblas.so => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/compilers/lib/libcudaforwrapblas.so
(0x000014ec8f8e1000)
libcudaforwrapblas117.so => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/compilers/lib/libcudaforwrapblas117.so
(0x000014ec8f6d0000)
libcudart.so.11.0 => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/cuda/11.8/lib64/libcudart.so.11.0 (0x000014ec8f427000)
libcudafor_118.so => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/compilers/lib/libcudafor_118.so (0x000014ec8946d000)
libcudafor.so => /opt/nvidia/hpc_sdk/Linux_x86_64/23.3/compilers/lib/libcudafor.so (0x000014ec8925a000)
...
```

Sanity Checking

- You can use the pattern "-Wl,-y<symbol_name>" at link time to report which library the linker is currently using for the symbol <symbol_name>"

```
ftn -Wl,-ydgemm_ -o /home/bertoni/w2-polaris/gamess/gamess.00.21.9.x -i8 -m64 -mcmodel=medium -
mp=gpu -gpu=cc80,lineinfo -Minfo=mp,accel -g -I/home/bertoni/w2-polaris/gamess/object gamess.o
gamess_version.o unport.o util.o aldeci.o algnci.o basccn.o basecp.o basext.o basg3l.o bashuz.o
bashz2.o baskar.o basminix.o basn21.o basn31.o baspcn.o basg3x.o bassto.o casino.o ccaux.o
ccddi.o ccqaux.o ccquad.o ccsdt.o ceeis.o cepa.o cnglob.o chgpen.o cimf.o ciminf.o cimi.o
cimlib.o cimsub.o cisgrd.o comp.o cosmo.o cosprt.o cphf.o cpmchf.o cprohf.o cphf_tddnlr.o
cpuhf.o dccc.o
...
/usr/bin/ld: ccaux.o: reference to dgemm_
/usr/bin/ld: ccddi.o: reference to dgemm_
...
/usr/bin/ld: /opt/cray/pe/libsci/21.08.1.2/NVIDIA/20.7/x86_64/lib/libsci_nvidia_mpi_mp.so:
reference to dgemm_
/usr/bin/ld: /opt/cray/pe/libsci/21.08.1.2/NVIDIA/20.7/x86_64/lib/libsci_nvidia_mp.so:
definition of dgemm_
```

Sanity Checking

Switched to using Nvidia’s BLAS and LAPACK

- You can use the pattern "-Wl,-y<symbol_name>" at link time to report which library the linker is currently using for the symbol <symbol_name>"

```
ftn -Wl,-ydgemm_ -o /home/bertoni/w2-polaris/gamess/gamess.00.21.9.x -i8 -m64 -mcmodel=medium -
mp=gpu -gpu=cc80,lineinfo -Minfo=mp,accel -g -I/home/bertoni/w2-polaris/gamess/object gamess.o
gamess_version.o unport.o util.o aldeci.o algnci.o basccn.o basecp.o basext.o basg3l.o bashuz.o
bashz2.o baskar.o basminix.o basn21.o basn31.o baspcn.o basg3x.o bassto.o casino.o ccaux.o
ccddi.o ccqaux.o ccquad.o ccsdt.o ceeis.o cepa.o cnglob.o chgpen.o cimf.o ciminf.o cimi.o
cimlib.o cimsub.o cisgrd.o comp.o cosmo.o cosprt.o cphf.o cpmchf.o cprohf.o cphf_tddnlr.o
cpuhf.o dccc.o
...
/usr/bin/ld: ccaux.o: reference to dgemm_
/usr/bin/ld: ccddi.o: reference to dgemm_
...
/usr/bin/ld: /opt/nvidia/hpc_sdk/Linux_x86_64/21.9/compilers/lib/libblas.so: definition of
dgemm_
/usr/bin/ld: /opt/nvidia/hpc_sdk/Linux_x86_64/21.9/compilers/lib/liblapack.so: reference to
dgemm_
```

Sanity Checking

- Please reach out to ALCF support if there are any issues!
- support@alcf.anl.gov

Debugging Game!

Debugging Game #1

- Problem: Undefined reference error

```
> make
...
/usr/bin/ld: gamess.o: in function `energx_':
/home/bertoni/w2-polaris/gamess/object/gamess.F:1519: undefined reference to `ddot_'
/usr/bin/ld: gamess.o: in function `cigrad_':
/home/bertoni/w2-polaris/gamess/object/gamess.F:1694: undefined reference to `dcopy_'
/usr/bin/ld: gamess.o: in function `hfgrad_':
/home/bertoni/w2-polaris/gamess/object/gamess.F:2231: undefined reference to `dcopy_'
/usr/bin/ld: gamess.o: in function `wfncc_':
/home/bertoni/w2-polaris/gamess/object/gamess.F:3079: undefined reference to `idamax_'
/usr/bin/ld: aldeci.o: in function `detinp_':
/home/bertoni/w2-polaris/gamess/object/aldeci.f:619: undefined reference to `dscal_'
/usr/bin/ld: /home/bertoni/w2-polaris/gamess/object/aldeci.f:657: undefined reference to `dscal_'
/usr/bin/ld: aldeci.o: in function `wtdm12_':
```

Debugging Game #1

- Issue: Usually this means that there's a library missing at link time
  - Either a –L${LIBRARY_PATH} is missing, a module is not loaded, or a flag like "-cudalib=cublas" is missing
- Solution:
  - In this case we were missing a blas library, and it can be resolved with "module load cray-libsci" (or pointing to a blas library) and rebuilding

Debugging Game #2

- Problem: Modules are all loaded but it can’t find cublas?!

```
> module list
Currently Loaded Modules:
  1) craype-x86-rome          6) craype/2.7.15         11) cray-pals/1.1.7
  2) libfabric/1.11.0.4.125   7) cray-dsmml/0.2.2      12) cray-libpals/1.1.7
  3) craype-network-ofi       8) cray-mpich/8.1.16     13) PrgEnv-nvhpc/8.3.3
  4) perftools-base/22.05.0   9) cray-pmi/6.1.2        14) craype-accel-nvidia80
  5) nvhpc/21.9              10) cray-pmi-lib/6.0.17

> CC  f.c -cuda -lcublas

/usr/bin/ld: cannot find -lcublas: No such file or directory
pgacclnk: child process exit status 1: /usr/bin/ld
```

Debugging Game #2

- Issue: Path to library is not set for the linker to find it
- The nvhpc libraries are not in the searched libraries by default
- Solution: Add in the path to the library

```
> CC f.c -cuda -lcublas -L${NVIDIA_PATH}/math_libs/lib64/
>
```

Debugging Game #3

- Problem: Everything compiles fine, and then you try to run:

```
> ./jobscript.sh
...

+ mpiexec -n 8 -ppn 8 --cpu-bind verbose,depth -d 8 ./set_gpu_affinity.sh /home/bertoni/w2-
polaris/gamess/gamess.00.21.9.x
cpubind:depth x3005c0s37b0n0 pid 54555 rank 0 0: mask 0x000000ff
cpubind:depth x3005c0s37b0n0 pid 54556 rank 1 1: mask 0x0000ff00
cpubind:depth x3005c0s37b0n0 pid 54557 rank 2 2: mask 0x00ff0000
cpubind:depth x3005c0s37b0n0 pid 54558 rank 3 3: mask 0xff000000
cpubind:depth x3005c0s37b0n0 pid 54559 rank 4 4: mask 0x000000ff,0x0
cpubind:depth x3005c0s37b0n0 pid 54560 rank 5 5: mask 0x0000ff00,0x0
cpubind:depth x3005c0s37b0n0 pid 54561 rank 6 6: mask 0x00ff0000,0x0
cpubind:depth x3005c0s37b0n0 pid 54562 rank 7 7: mask 0xff000000,0x0

/home/bertoni/w2-polaris/gamess/gamess.00.21.9.x: error while loading shared libraries: libfabric.so.1:
cannot open shared object file: No such file or directory
```

Debugging Game #3

- Issue: I didn't set modules correctly at runtime

```
> ldd gamess.00.21.9.x
        linux-vdso.so.1 (0x00007ffc5fff4000)
        libmpi_nvidia.so.12 => /opt/cray/pe/lib64/libmpi_nvidia.so.12 (0x0000145d731a4000)
        libsci_nvidia_mp.so.5.0 => /opt/cray/pe/lib64/libsci_nvidia_mp.so.5.0 (0x0000145d6f726000)
        libdl.so.2 => /lib64/libdl.so.2 (0x0000145d6f721000)
...

        librt.so.1 => /lib64/librt.so.1 (0x0000145d3f9e2000)
        libm.so.6 => /lib64/libm.so.6 (0x0000145d3f897000)
        libc.so.6 => /lib64/libc.so.6 (0x0000145d3f6a2000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x0000145d3f483000)
        libfabric.so.1 => not found
```

- Solution: module load the needed module for libfabric

Summary

- Polaris uses compiler wrappers, some libraries are already linked in
- "module avail" to see additional available modules
- Do sanity checks!
  - ldd ./exe to check what's linked in
- Please reach out to ALCF support if there are any issues!
  - support@alcf.anl.gov

Thanks!

Questions?