# Math (and other) Libraries on Aurora

Colleen Bertoni
Argonne Leadership Computing Facility

# Outline

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

# Getting Started

# Getting Started: Modules

- Environments handled by modules on Aurora

```
> module list

Currently Loaded Modules:
  1) gcc-runtime/13.3.0-ghotoln (H)    8) libxml2/2.13.5
  2) gmp/6.3.0-mtokfaw          (H)    9) hwloc/2.11.3-mpich-level-zero
  3) mpfr/4.2.1-gkcdl5w         (H)   10) yaksa/0.3-7ks5f26           (H)
  4) mpc/1.3.1-rdrlvsl          (H)   11) mpich/opt/develop-git.6037a7a
  5) gcc/13.3.0                       12) libfabric/1.22.0
  6) oneapi/release/2025.0.5          13) cray-pals/1.4.0
  7) libiconv/1.17-jjpb4sl      (H)   14) cray-libpals/1.4.0

  Where:
   H:  Hidden Module
```

- 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:
  - icpx, icx, ifx
- MPI compiler Wrappers:
  - **mpicxx: C++ compiler wrapper**
  - **mpicc: C compiler wrapper**
  - **mpifort / mpif90: Fortran compiler wrapper**

# Getting Started: Modules

- Compilers:
  - icpx, icx, ifx
- MPI compiler Wrappers:
  - **mpicxx: C++ compiler wrapper**
  - **mpicc: C compiler wrapper**
  - **mpifort / mpif90: Fortran compiler wrapper**

```
> mpicxx –show

icpx -I/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/mpich-develop-git.6037a7a-sxnhr7p/include
-L/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/mpich-develop-git.6037a7a-sxnhr7p/lib -lmpicxx -Wl,-rpath
-Wl,/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/mpich-develop-git.6037a7a-sxnhr7p/lib -lmpi
```

# Getting Started: Linking

- Like compilers, libraries are also managed by modules on Aurora
  - The modules work by setting PATH, LD_LIBRARY_PATH, etc. in your environment to point to the right library
- **Library linking on Aurora is dynamic linking**
  - 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

# Getting Started: Linking

- Like compilers, libraries are also managed by modules on Aurora
  - The modules work by setting PATH, LD_LIBRARY_PATH, etc. in your environment to point to the right library
- **Library linking on Aurora is dynamic linking**
  - 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 MPICH library with the MPI wrappers is 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

# Getting Started: Linking

- Like compilers, libraries are also managed by modules on Aurora
- “module load $library” to set the environment so that the library is pulled in

```
> module load petsc
```

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

```
> module show petsc
[...]
conflict("petsc")
prepend_path{"LD_LIBRARY_PATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/lib",delim=":"}
prepend_path{"LIBRARY_PATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/lib",delim=":"}
prepend_path{"C_INCLUDE_PATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/include",delim=":"}
prepend_path{"CPLUS_INCLUDE_PATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/include",delim=":"}
prepend_path{"INCLUDE","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/include",delim=":"}
prepend_path{"CPATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/include",delim=":"}
prepend_path{"CMAKE_PREFIX_PATH","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx/.",delim=":"}
setenv("PETSC_DIR","/opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/petsc-3.21.4-ytrgfcx")
[...]
```

# Available Math Libraries

# Available Math libraries

- Intel oneMKL (default)
- Non-exhaustive list of non-Intel-provided libraries: `module avail`
  - AMReX: Software Framework for Block Structured AMR
    - module load amrex
  - boost: Optimized C++ libraries
    - module load boost
  - double-batched-fft-library: FFT on GPUs (https://github.com/intel/double-batched-fft-library)
    - module load double-batched-fft-library
  - Heffte: highly efficient FFTs for Exascale
    - module load heffte
  - HYPRE: scalable parallel linear solvers
    - module load hypre
  - PETSc: scalable (parallel) partial differential equations (PDEs)
    - module load magma
  - superLU: general unsymmetric sparse linear systems
    - module load ginko
  - netlib-scalapack
    - module load netlib-scalapack
  - etc.

# Intel oneMKL

- Intel oneMKL (used to be MKL)
- Available in oneapi module (loaded by default, but not linked in by default)

- BLAS, LAPACK, FFT, Sparse Linear Algebra, Data fitting, Vector math, summary statistics, RNG, etc.

- C/C++/Fortran with 64- and 32-bit int interfaces
- Support for many data types, like FP16, BF16, TF32

- CPU APIs are unchanged from MKL
- GPU APIs for SYCL and OpenMP offload code

- https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
- https://www.intel.com/content/www/us/en/docs/onemkl/developer-reference-c/2025-0/overview.html

```
> ls -ltr $MKLROOT/lib
libmkl_vml_mc3.so.2
libmkl_vml_def.so.2
libmkl_vml_cmpt.so.2
libmkl_vml_avx512.so.2
libmkl_vml_avx2.so.2
libmkl_tbb_thread.so.2
libmkl_tbb_thread.so -> libmkl_tbb_thread.so.2
libmkl_tbb_thread.a
libmkl_sycl_vm.so.5
libmkl_sycl_vm.so -> libmkl_sycl_vm.so.5
libmkl_sycl_stats.so.5
libmkl_sycl_stats.so -> libmkl_sycl_stats.so.5
libmkl_sycl_sparse.so.5
libmkl_sycl_sparse.so -> libmkl_sycl_sparse.so.5
libmkl_sycl.so
libmkl_sycl_rng.so.5
libmkl_sycl_rng.so -> libmkl_sycl_rng.so.5
libmkl_sycl_lapack.so.5
libmkl_sycl_lapack.so -> libmkl_sycl_lapack.so.5
libmkl_sycl_dft.so.5
libmkl_sycl_dft.so -> libmkl_sycl_dft.so.5
libmkl_sycl_data_fitting.so.5
libmkl_sycl_data_fitting.so -> libmkl_sycl_data_fitting.so.5
libmkl_sycl_blas.so.5
libmkl_sycl_blas.so -> libmkl_sycl_blas.so.5
libmkl_sycl.a
libmkl_sequential.so.2
libmkl_sequential.so -> libmkl_sequential.so.2
...
```

# oneMKL examples

```
> ls -ltr $MKLROOT/share/doc/mkl/examples
-rw-r--r-- 1 root root   12601 Nov  7 13:37 README.txt
-rw-r--r-- 1 root root 1020843 Nov  7 13:37 examples_sycl.tgz
-rw-r--r-- 1 root root  120016 Nov  7 13:37 examples_offload_f.tgz
-rw-r--r-- 1 root root  129431 Nov  7 13:37 examples_offload_c.tgz
-rw-r--r-- 1 root root   34298 Nov  7 13:37 examples_f95.tgz
-rw-r--r-- 1 root root 1456998 Nov  7 13:37 examples_core_f.tgz
-rw-r--r-- 1 root root 1429409 Nov  7 13:37 examples_core_c.tgz
-rw-r--r-- 1 root root   41605 Nov  7 13:37 examples_cluster_f.tgz
-rw-r--r-- 1 root root   33844 Nov  7 13:37 examples_cluster_c.tgz
drwxr-xr-x 2 root root      79 Apr 26 22:30 cmake

> cp -r $MKLROOT/share/doc/mkl/examples .
> cd examples
> tar -xzvf examples_offload_f.tgz
> ls -ltr f_offload/lapack/source/
-rw-r--r-- 1 bertoni users 3989 Oct 31 13:54 zheevd.f90
-rw-r--r-- 1 bertoni users 3421 Oct 31 13:54 zgels_batch_strided.f90
-rw-r--r-- 1 bertoni users 3076 Oct 31 13:54 ssyevd.f90
-rw-r--r-- 1 bertoni users 3352 Oct 31 13:54 sgels_batch_strided.f90
-rw-r--r-- 1 bertoni users 3112 Oct 31 13:54 dsyevd.f90
-rw-r--r-- 1 bertoni users 2686 Oct 31 13:54 dpotrs.f90
-rw-r--r-- 1 bertoni users 2494 Oct 31 13:54 dpotri.f90
-rw-r--r-- 1 bertoni users 2360 Oct 31 13:54 dpotrf.f90
-rw-r--r-- 1 bertoni users 5098 Oct 31 13:54 dgetri_oop_batch_strided.f90
-rw-r--r-- 1 bertoni users 3392 Oct 31 13:54 dgels_batch_strided.f90
-rw-r--r-- 1 bertoni users 3937 Oct 31 13:54 cheevd.f90
-rw-r--r-- 1 bertoni users 3399 Oct 31 13:54 cgels_batch_strided.f90
```

# oneMKL Linking

- Link line adviser (Essential if you're compiling on command line or using a Makefile)
  - https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-link-line-advisor.html
  - Often just `-qmkl` will work ok though!

- Cmake
  - https://www.intel.com/content/www/us/en/docs/onemkl/developer-guide-windows/2025-1/cmake-config-for-onemkl.html
  - $MKLROOT/lib/cmake/mkl/MKLConfig.cmake

```
# Below INTERFACE targets provide full link-lines for direct use.
# Example:
#    target_link_options(<my_linkable_target> PUBLIC MKL::MKL)
#
# MKL::MKL
#    Link line for C and Fortran API
# MKL::MKL_SYCL
#    Link line for SYCL API
# MKL::MKL_SCALAPACK
#    Link line for ScaLAPACK and PBLAS API
# MKL::MKL_BLACS
#    Link line for BLACS and CPARDISO API (includes MKL::MKL)
```

| | |
|---|---|
| Select compiler: | Intel(R) oneAPI DPC++/C++ |
| Select architecture: | Intel(R) 64 |
| Select dynamic or static linking: | Dynamic |
| Select interface layer: | DPC++ API |
| Select threading layer: | OpenMP threading |
| Select OpenMP library: | Intel(R) (libiomp5) |
| Enable OpenMP offload feature to GPU: | ☐ |
| Select cluster library: | ☐ Parallel Direct Sparse Solver for Clusters (BLACS required)<br>☐ Cluster Discrete Fast Fourier Transform (BLACS required)<br>☐ ScaLAPACK (BLACS required)<br>☐ BLACS |
| Select MPI library: | <Select MPI> |
| Select the Fortran 95 interfaces: | ☐ BLAS95<br>☐ LAPACK95 |
| Select SYCL domain library: | Sparse |
| Link with Intel® oneMKL libraries explicitly: | ☐ |
| Link with DPC++ debug runtime compatible libraries: | ☐ |

Use this link line:

```
-qmkl=parallel  -fsycl -qmkl-sycl-impl=sparse
```

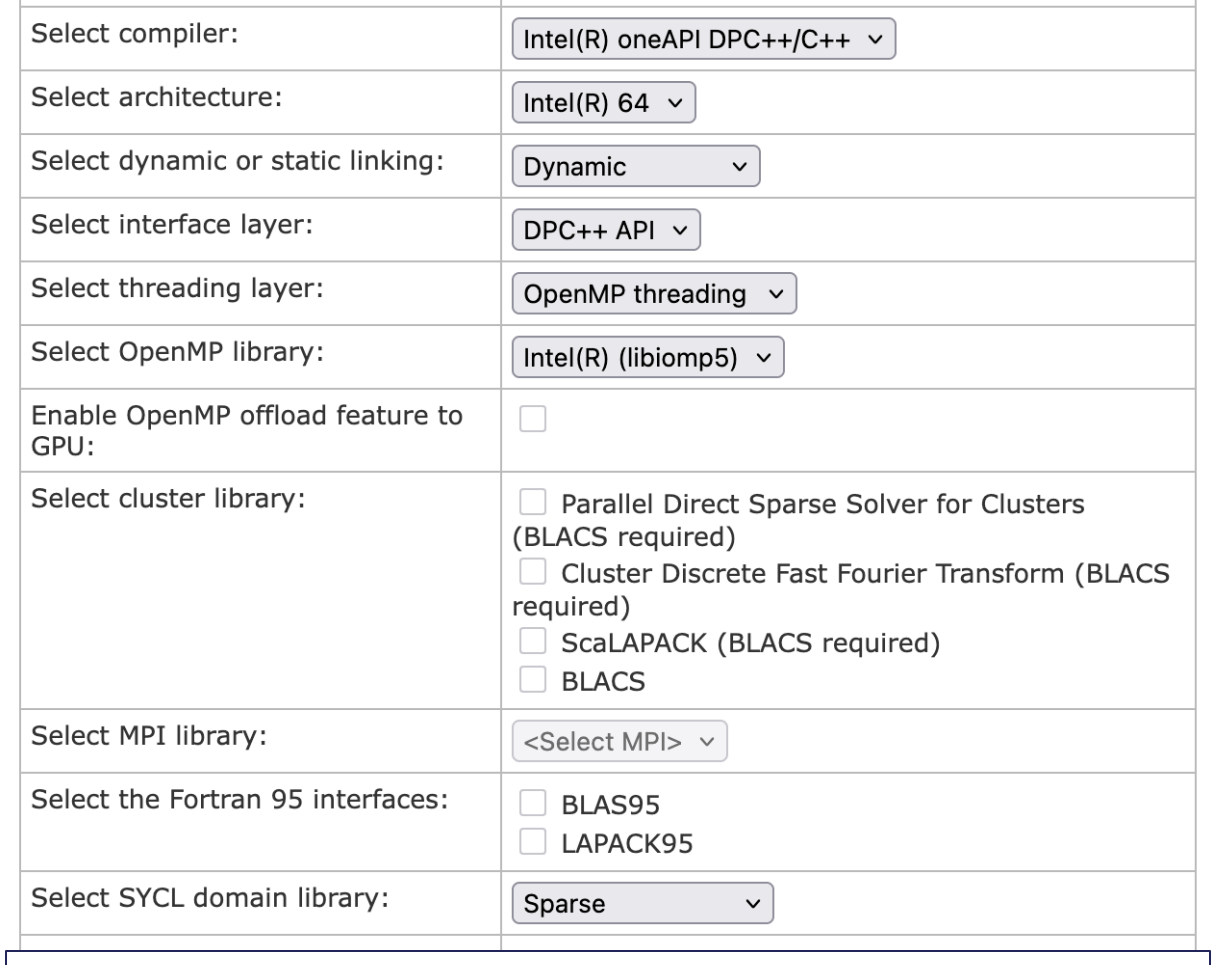
Compiler options:

```
-fsycl -DMKL_ILP64  -qmkl=parallel
```

# oneMKL Linking

- Link line adviser (Essential if you're hand-compiling or using a Makefile)
  - https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-link-line-advisor.html
  - Often just `-qmkl` will work ok though!

- Cmake
  - https://www.intel.com/content/www/us/en/docs/onemkl/developer-guide-windows/2025-1/cmake-config-for-onemkl.html
  - $MKLROOT/lib/cmake/mkl/MKLConfig.cmake




```
# Below INTERFACE targets provide full link-lines for direct use.
# Example:
#    target_link_options(<my_linkable_target> PUBLIC MKL::MKL)
#
# MKL::MKL
#    Link line for C and Fortran API
# MKL::MKL_SYCL
#    Link line for SYCL API
# MKL::MKL_SCALAPACK
#    Link line for ScaLAPACK and PBLAS API
# MKL::MKL_BLACS
#    Link line for BLACS and CPARDISO API (includes MKL::MKL)
```

```
> cat CMakelists.txt
cmake_minimum_required(VERSION 3.13)
project(oneMKL_Example LANGUAGES C)
find_package(MKL CONFIG REQUIRED PATHS $ENV{MKLROOT})
add_executable(myapp app.c)
target_link_libraries(myapp PUBLIC MKL::MKL)
```

# Sanity Checking

# Sanity checking

- **Dynamic linking**
- 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.link.x
        linux-vdso.so.1 (0x00007ffc1e379000)
        libmkl_sycl_blas.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_blas.so.5 (0x00007f313b5a2000)
        libmkl_sycl_lapack.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_lapack.so.5
        libmkl_sycl_sparse.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_sparse.so.5
        libmkl_sycl_dft.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_dft.so.5 (0x00007f312e49f000)
        libmkl_sycl_vm.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_vm.so.5 (0x00007f3124ec2000)
        libmkl_sycl_rng.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_rng.so.5 (0x00007f311f859000)
        libmkl_sycl_stats.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_stats.so.5 (0x00007f311d99e000)
        libmkl_sycl_data_fitting.so.5 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sycl_data_fitting.so.5
        libmkl_intel_ilp64.so.2 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_intel_ilp64.so.2
        libmkl_sequential.so.2 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_sequential.so.2 (0x00007f311ade8000)
        libmkl_core.so.2 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_core.so.2 (0x00007f3116e37000)
        libsycl.so.8 => /opt/aurora/24.347.0/oneapi/compiler/latest/lib/libsycl.so.8 (0x00007f3116aa6000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007f3116a6b000)
        libimf.so => /opt/aurora/24.347.0/oneapi/compiler/latest/lib/libimf.so (0x00007f311665f000)
        libm.so.6 => /lib64/libm.so.6 (0x00007f3116512000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007f311650d000)
        libmpi.so.12 => /opt/aurora/24.347.0/spack/unified/0.9.2/install/linux-sles15-x86_64/oneapi-2025.0.5/mpich-develop-git.6037a7a-sxnhr7p/lib/libmpi.so.12
(0x00007f3114763000)
        ...
```

# 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>"

```
ifx -Wl,-ydgemm_ -o gamess.00.21.9.x -i8 -fiopenmp -fp-model=precise -fopenmp-targets=spir64_gen -mcmodel=large -g -
I/home/bertoni/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 -fsycl -L/opt/aurora/24.347.0/oneapi/mkl/latest/\
lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lsycl -lpthread ...
...
ld: /var/tmp/pbs.4574093.aurora-pbs-0001.anl.gov/ifxU5oHFFccaux.o: reference to dgemm_
ld: /var/tmp/pbs.4574093.aurora-pbs-0001.anl.gov/ifxK7zwfFvxx.o: reference to dgemm_
...
ld: /opt/aurora/24.347.0/oneapi/mkl/latest/lib/intel64/libmkl_intel_ilp64.so: definition of dgemm_
```

# Sanity Checking

Switched to using LP64 instead of ILP64

- 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>"

ifx **-Wl,-ydgemm_** -o gamess.00.21.9.x -i8 -fiopenmp -fp-model=precise -fopenmp-targets=spir64_gen -mcmodel=large -g -I/home/bertoni/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 -fsycl -L/opt/aurora/24.347.0/oneapi/mkl/latest/\
lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lsycl -lpthread ...
...
ld: /var/tmp/pbs.4574093.aurora-pbs-0001.anl.gov/ifxU5oHFFccaux.o: reference to dgemm_
ld: /var/tmp/pbs.4574093.aurora-pbs-0001.anl.gov/ifxK7zwfFvxx.o: reference to dgemm_
...
ld: /opt/aurora/24.347.0/oneapi/mkl/latest/lib/intel64/libmkl_intel_lp64.so: definition of dgemm_

# Sanity Checking

- You can use MKL_VERBOSE=2 to see a trace of MKL calls and sizes
- https://www.intel.com/content/www/us/en/developer/articles/technical/verbose-mode-supported-in-onemkl-112.html
  - Some restrictions on which calls show up
  - Summary tool: https://github.com/TApplencourt/mkl-verbose-toolkit

```
Running tests on GPU.
Running with double precision real data type:
MKL_VERBOSE oneMKL 2025 Patch 1 Product build 20241031 for Intel(R) 64 architecture Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) with support for INT8, BF16, FP16 (limited) instructions, and Intel\
(R) Advanced Matrix Extensions (Intel(R) AMX) with INT8 and BF16, Lnx 2.00GHz ilp64 intel_thread
MKL_VERBOSE Detected GPU0 Intel(R)_Xe_HPC Backend:Level_Zero VE:896 Stack:2 maxWGsize:1024
...
MKL_VERBOSE
oneapi::mkl::blas::row/column_major::dot[double](0x7ffe31da1c78,20000000,0x14940e600000,1,0x149404c00000,1,0x14942ca70000,Vector<sycl::event>OfSize:0) host:210.80ms device:N/A GPU0
MKL_VERBOSE
oneapi::mkl::blas::row/column_major::dot[double](0x7ffe31da1c78,20000000,0x14940e600000,1,0x149404c00000,1,0x14942ca70000,Vector<sycl::event>OfSize:0) host:654.42us device:N/A GPU0
MKL_VERBOSE
oneapi::mkl::blas::row/column_major::dot[double](0x7ffe31da1c78,20000000,0x14940e600000,1,0x149404c00000,1,0x14942ca70000,Vector<sycl::event>OfSize:0) host:536.71us device:N/A GPU0
```

# 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
...
/home/bertoni/gamess/object/mthlib.f:(.ltext+0x309b): undefined reference to `dgemm_'
ld: /tmp/ifx05813697443gHnu1/ifxbZzwG0mthlib.o: in function `mtarbr_':
/home/bertoni/gamess/object/mthlib.f:(.ltext+0x3a3e): undefined reference to `dsymm_'
ld: /tmp/ifx05813697443gHnu1/ifxbZzwG0mthlib.o: in function `dmspmm_':
/home/bertoni/gamess/object/mthlib.f:(.ltext+0x3da7): undefined reference to `dspmv_'
ld: /home/bertoni/projects/p62.restart_gamess/gamess/object/mthlib.f:(.ltext+0x3dc9): undefined reference to `dcopy_'
ld: /home/bertoni/gamess/object/mthlib.f:(.ltext+0x3e11): undefined reference to `dgemv_'
ld: /tmp/ifx05813697443gHnu1/ifxbZzwG0mthlib.o: in function `onvmgs_':
/home/bertoni/gamess/object/mthlib.f:1258:(.ltext+0x3f7e): undefined reference to `ddot_'
...
```

# Debugging Game #1

- Issue: Usually this means that there's a library missing at link time
  - Either a –L${LIBRARY_PATH} is missing, a module is not loaded, or a flag is missing
- Solution:
  - In this case we were missing a blas library, and it can be resolved by rebuilding with
    - " -fsycl  -L${MKLROOT}/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core -lsycl -lpthread -lm -ldl" (from Link Line Adviser) or
    - "-qmkl"

# Debugging Game #1

- Issue: Usually this means that there's
  - Either a –L${LIBRARY_PATH} is
    is missing
- Solution:
  - In this case we were missing a bla
    rebuilding with
    - " -fsycl  -L${MKLROOT}/lib/inte
      lmkl_sequential -lmkl_core -lsy
      Adviser) or
    - "-qmkl"

| | |
|---|---|
| Select dynamic or static linking: | Dynamic |
| Select interface layer: | DPC++ API |
| Select threading layer: | OpenMP threading |
| Select OpenMP library: | Intel(R) (libiomp5) |
| Enable OpenMP offload feature to GPU: | ☐ |
| Select cluster library: | ☐ Parallel Direct Sparse Solver for Clusters (BLACS required)<br>☐ Cluster Discrete Fast Fourier Transform (BLACS required)<br>☐ ScaLAPACK (BLACS required)<br>☐ BLACS |
| Select MPI library: | <Select MPI> |
| Select the Fortran 95 interfaces: | ☐ BLAS95<br>☐ LAPACK95 |
| Select SYCL domain library: | Sparse |
| Link with Intel® oneMKL libraries explicitly: | ☐ |
| Link with DPC++ debug runtime compatible libraries: | ☐ |

Use this link line:

```
-qmkl=parallel  -fsycl -qmkl-sycl-impl=sparse
```

Compiler options:

```
-fsycl -DMKL_ILP64  -qmkl=parallel
```

# Debugging Game #2

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

```
> ./jobscript.sh
…

+ mpiexec -n 24 -ppn 24 --cpu-bind depth -d 8 gpu_tile_compact.sh gamess.00.link.x
gamess.00.link.x: error while loading shared libraries: libmpi.so.12: cannot open shared object file: No such file or directory
gamess.00.link.x: error while loading shared libraries: libmpi.so.12: cannot open shared object file: No such file or directory
x4408c3s3b0n0.hostmgmt2408.cm.aurora.alcf.anl.gov: rank 7 exited with code 127
```

# Debugging Game #2

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

```
> ldd gamess.00.link.x
    linux-vdso.so.1 (0x00007ffef3be3000)
    ...
    libmkl_core.so.2 => /opt/aurora/24.347.0/oneapi/mkl/latest/lib/libmkl_core.so.2 (0x000014f6270dd000)
    libsycl.so.8 => /opt/aurora/24.347.0/oneapi/compiler/latest/lib/libsycl.so.8 (0x000014f626d4c000)
    libpthread.so.0 => /lib64/libpthread.so.0 (0x000014f626d0f000)
    libimf.so => /opt/aurora/24.347.0/oneapi/compiler/latest/lib/libimf.so (0x000014f626903000)
    libm.so.6 => /lib64/libm.so.6 (0x000014f6267b8000)
    libdl.so.2 => /lib64/libdl.so.2 (0x000014f6267b3000)
    libmpi.so.12 => not found
    libmpifort.so.12 => not found
```

- Solution: module load the needed module for mpich, or `module restore` to get back to the original environment

# Debugging Game #3

- Problem: Code runs, but there's a strange runtime error about MKL parameters

```
> mpiexec –n … ./a.out
…

Intel oneMKL ERROR: Parameter 1 was incorrect on entry to DGESV .
```

# Debugging Game #3

- Issue: MKL thinks there's a problem with an input parameter
- We can use MKL_VERBOSE=2 to look at what the parameters are (if you can find the section)

```
> MKL_VERBOSE=2 ./a.out
…
Intel oneMKL ERROR: Parameter 1 was incorrect on entry to DGESV .
…
MKL_VERBOSE DGESV(-2213609290906730491, 21474836483, 0x7ffec80609c0, 12884901893, 0x7ffec80609a0,
0x7ffec8060920,25769803775,-1) 89.35us CNR:OFF Dyn:1 FastMM:1 TID:0  NThr:104
```

- From looking at the API, we know that the first parameter is an integer pointer, so why is this a problem?
- Solution: The 32-bit interface was needed but we linked with the 64-bit interface
  - By default, `-qmkl` links in the 64-bit version. If you are sending in pointers to 32-bit integers, be sure to link the correct MKL version

# Summary

- Aurora 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

# Extra Debugging Tool for Fortran

# Codee

- A “linter” for Fortran!
  - The Codee tool automatically analyzes your code line-by-line to identify and fix opportunities for correctness, modernization, security and optimization.

- module use /soft/modulefiles
- module load codee

- https://docs.codee.com/

# Thanks!

Questions?