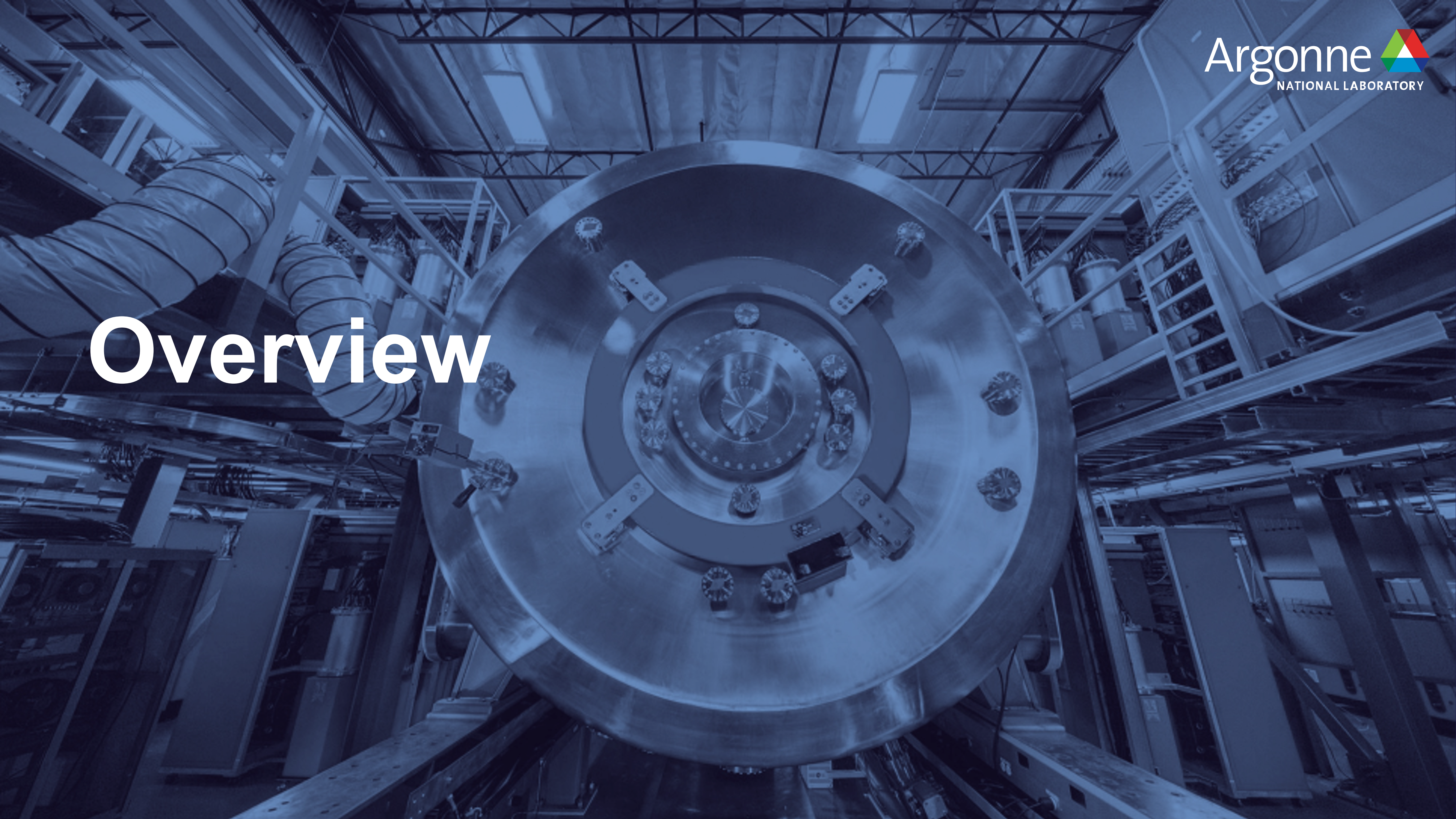
Bringing Performance Portability to the Exascale Era with C++ and SYCL

Distributed and Heterogeneous Programming in C++ (DHPCC++22)

Kevin Harms
Argonne National Laboratory

Overview



Summary of DOE Activities - ECP

- Exascale Computing Projects (ECP) applications and software using SYCL
  - https://www.exascaleproject.org
  - Kokkos – Portable GPU programming model
    - SYCL backend
  - RAJA – Portable GPU programming model
    - SYCL backend
  - AMReX – Portable programming model for block structured AMR
    - SYCL backend
  - NWChemEx – will support a broad range of chemistry research important to DOE BER and DOE Basic Energy Sciences on computing systems that range from terascale workstations and petascale servers to exascale computers.
    - SYCL used in various components
  - NekRS - a GPU-oriented thermal-fluids simulation code based on the spectral element method (SEM)
    - OCCA portability layer has SYCL backend

Summary of DOE Activities - Facilities

- Argonne Leadership Computing Facility (ALCF)
  - https://alcf.anl.gov
  - Preparing Aurora Exascale computer for launch (Intel PVC)
    - SYCL is our primary programming model for direct GPU programming
  - Intel DPC++ is our primary SYCL implementation
  - Have tested ComputeCPP and hipSYCL
  - Active in Khronos Specification process
- Oakridge Leadership Computing Facility (OLCF)
  - https://olcf.ornl.gov
  - Collaboration on creating DPC++ plugin implementation for AMD
  - Potential support SYCL as alternate programming model for Frontier (AMD MI-250X)
- National Energy Research Scientific Computing Center (NERSC)
  - https://nersc.gov
  - Collaboration on enhancing initial Nvidia DPC++ implementation
    - Targeting support for Perlmutter (Nvidia A100)
  - Support for SYCL as alternate programming model on Perlmutter

DOE Funded Work – Nvidia Support

- ALCF joint project with NERSC to develop/enhance PI_CUDA
  - —Led by Brandon Cook (NERSC)
  - —Contracted with Codeplay to execute the work
  - —Support A100 / SM_80 architecture
  - —Optimizations for A100
    - PTX 7.0 builtins
  - —Joint Matrix proposal to support Nvidia Tensor Core
    - Optimize matrix-matrix operations
    - Intel working on support of VNNI/AMX instructions
  - —Device to Device memory transfer
  - —Year of maintenance (currently ongoing)

DOE Funded Work – AMD Support

- ALCF and OLCF joint project with Codeplay to develop PI_HIP
  - Led by Kevin Harms and David Bernholdt
  - Develop a plugin based on HIP
  - Reuse existing clang support for generating HIP device code
  - Target MI-50 and MI-100 AMD GPUs
  - Prototype project has been completed
  - Develop sufficient support to execute five benchmark applications
    - Approximate 50% completion of plugin interface
    - Some builtins missing
    - Interop missing

oneAPI

- Intel initiative to create open specifications for software components compatible with SYCL
- Creating an ecosystem around the SYCL programming model
- Examples
  - —oneDPL implemented in SYCL and can be built for non-Intel backends
  - —oneMKL has initial support for Nvidia and AMD
    - Utilizes optimized cuBLAS and rocBLAS

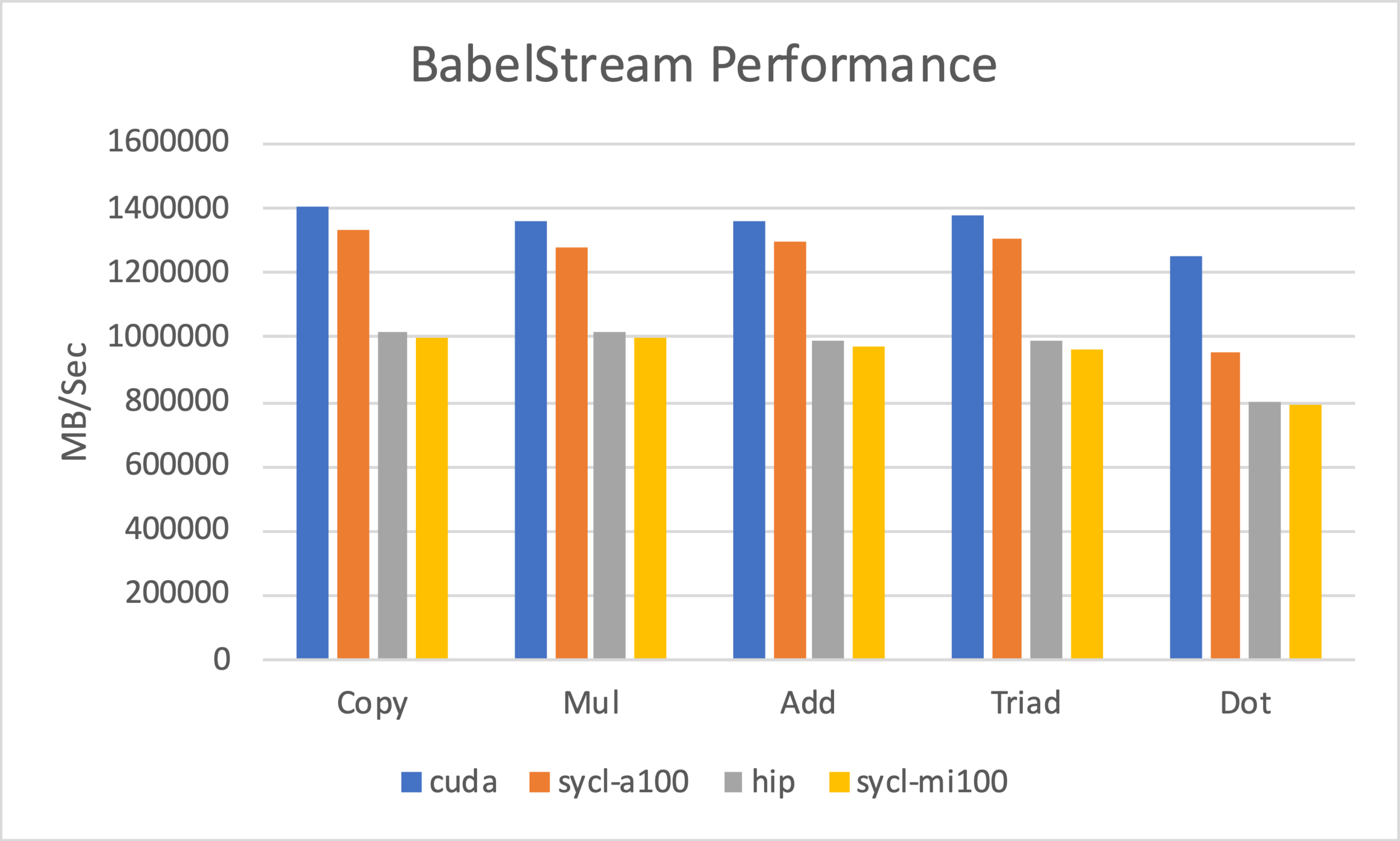
Performance

Performance Evaluation

- Study of five benchmark / miniapps that were focus of work for initial AMD support
  - —BabelStream
  - —LULESH
  - —RSbench
  - —SYCLDSlash (implementation of Dslash operator used in QCD codes)
  - —SYCL reduction benchmarks

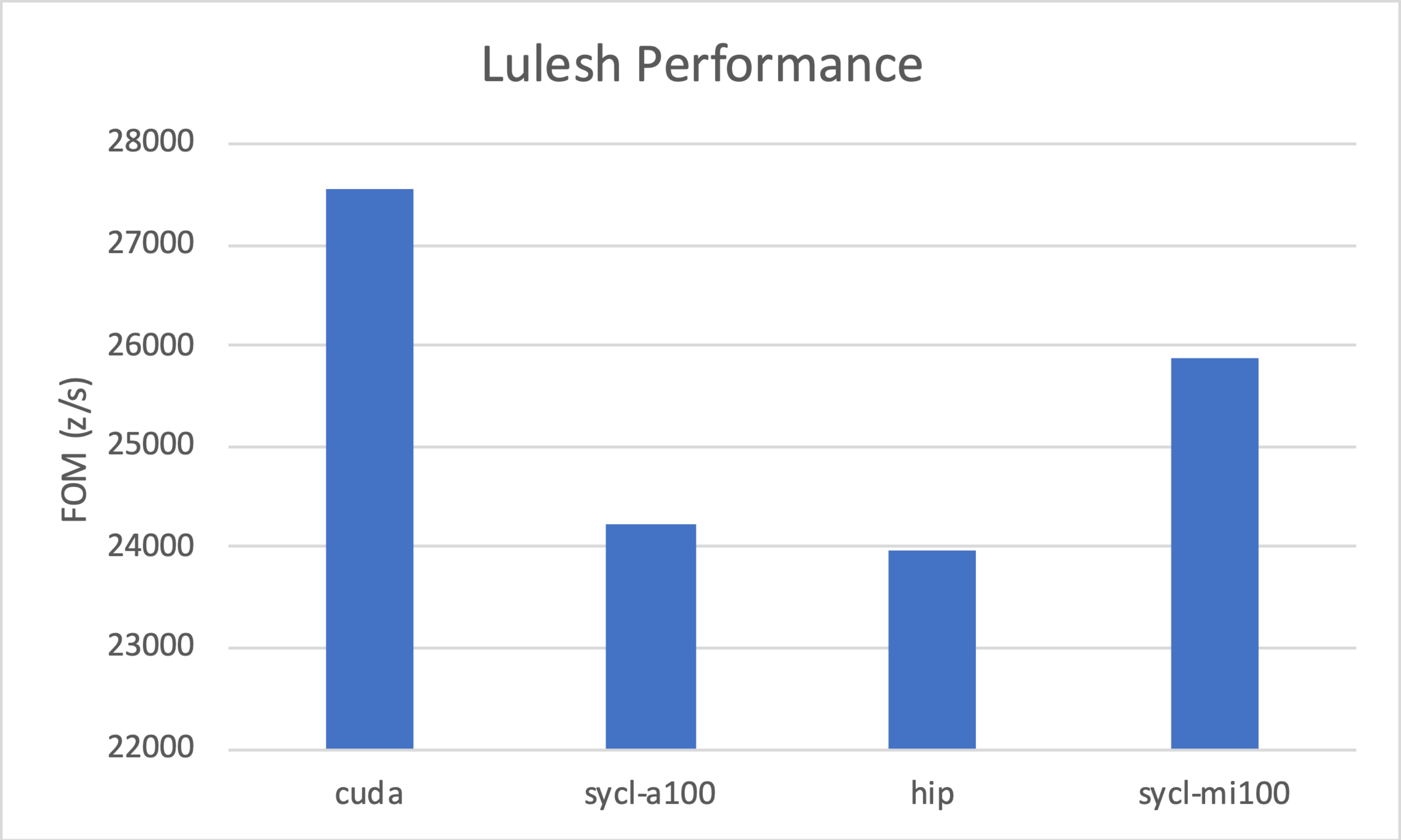
- Study was done by Codeplay using public hardware available in Argonne’s Joint Laboratory for System Evaluation (JLSE)
  - —Nvidia A100
  - —AMD MI-50
  - —AMD MI-100

BabelStream



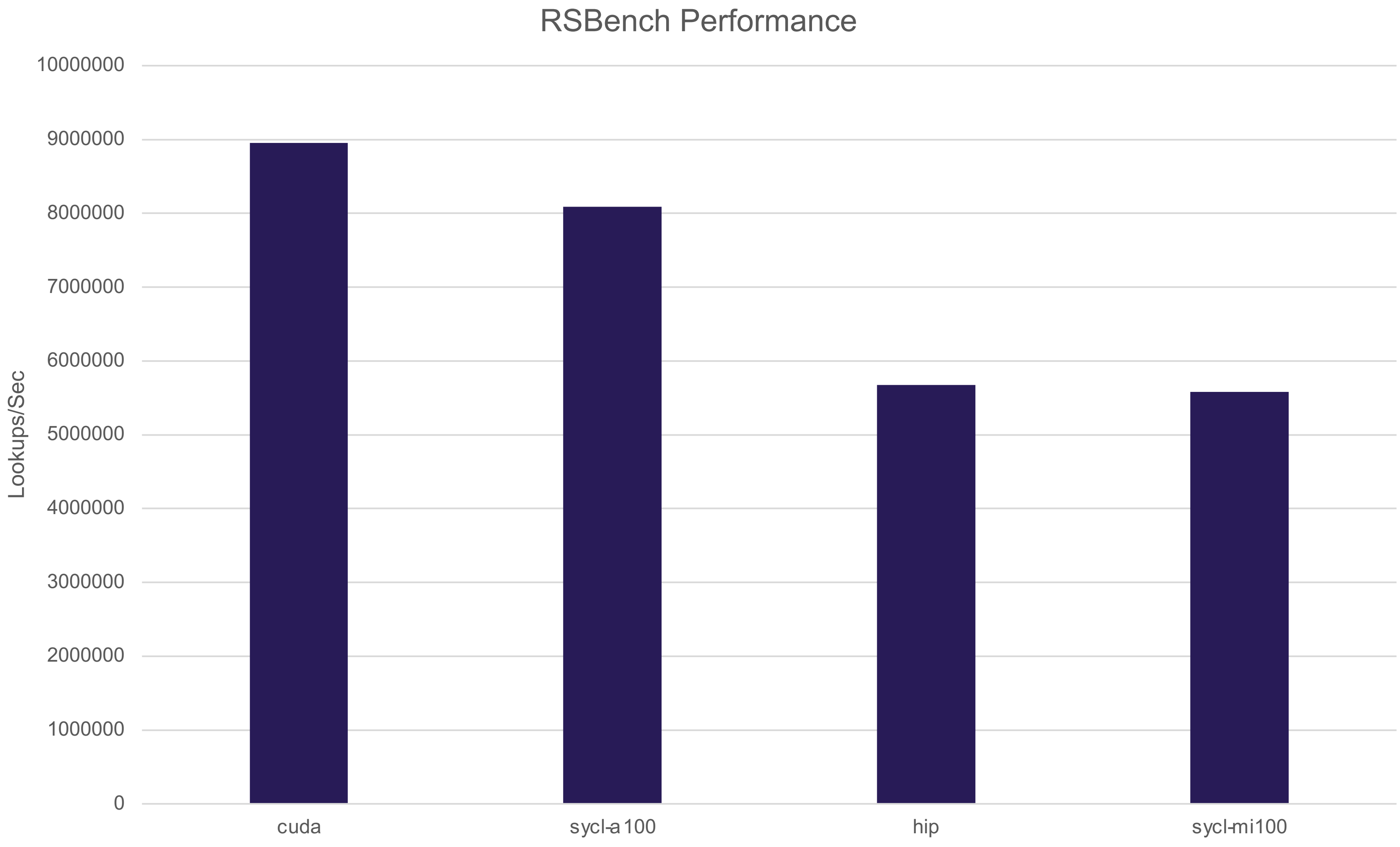
Credit Codeplay Software

LULESH



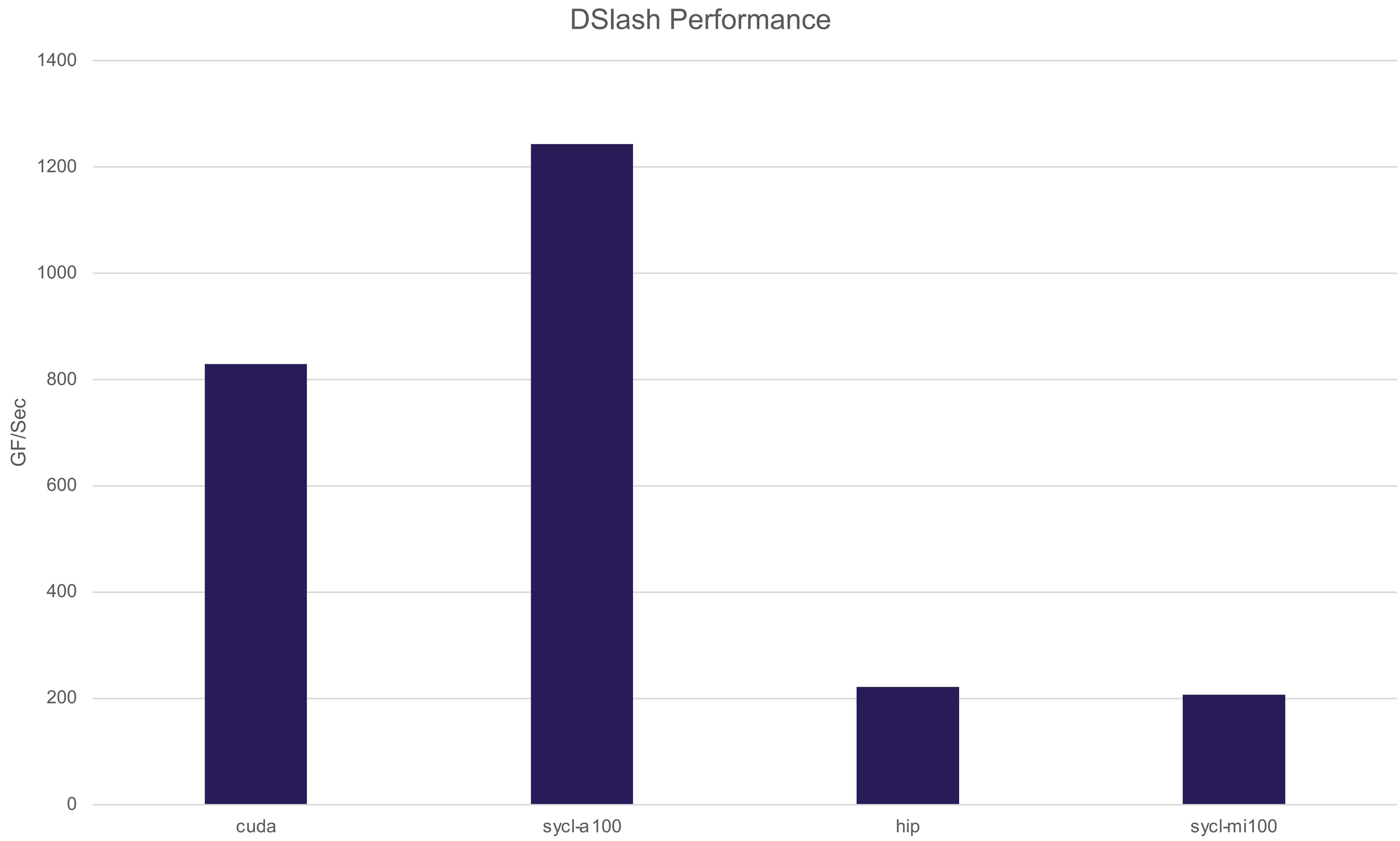
Credit Codeplay Software

RSBENCH



Credit Codeplay Software

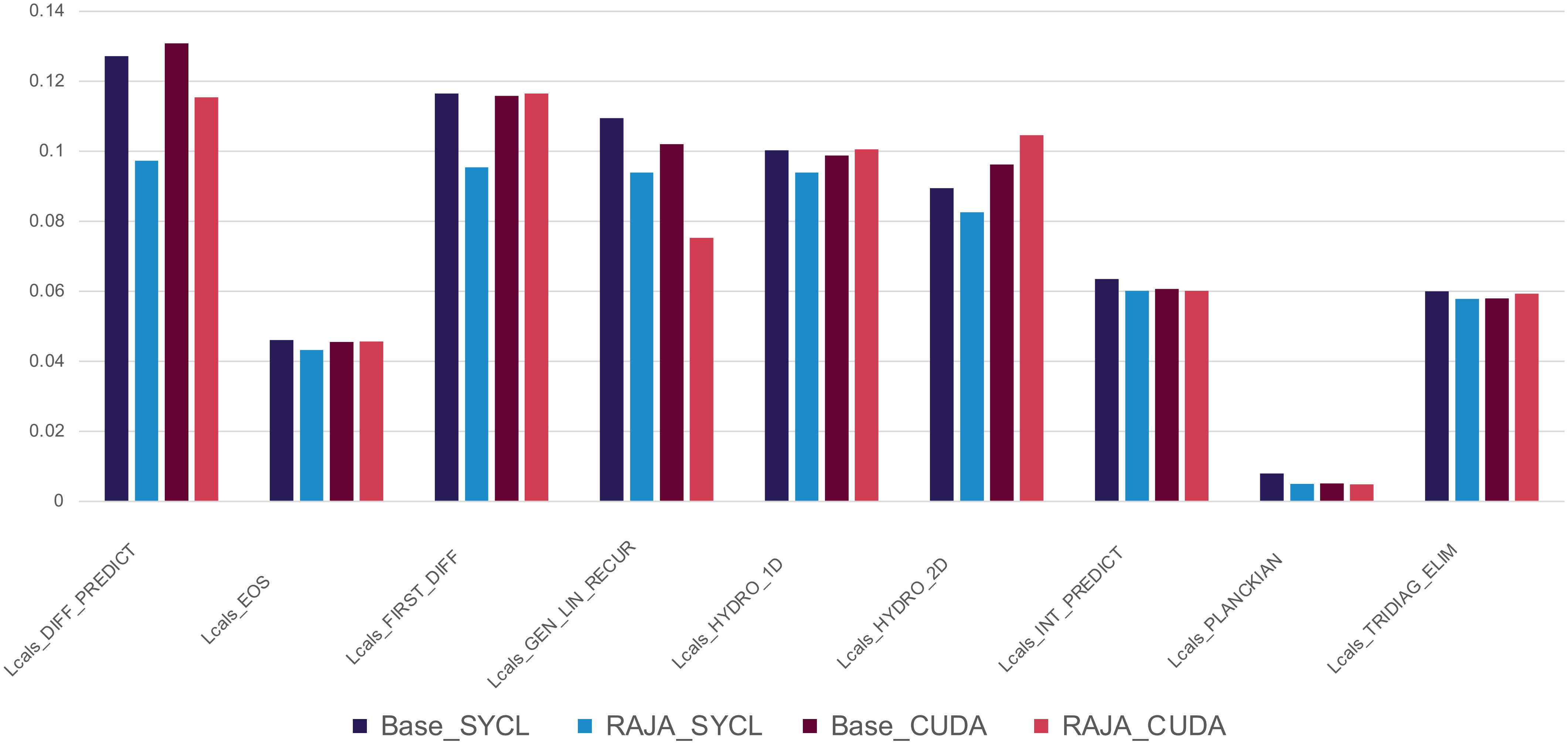
SYCLDSlash



Credit Codeplay Software

RAJAPerf – LCALS kernel execution time (lower is better)

Credit Brian Homerding



Features

SYCL Complex

- Complex numbers are widely used in HPC
- Currently, SYCL doesn't have support for complex number in device code within the specification
  - —A DPCPP implementation existed, but only implemented for L0/OpenCL backend
- Argonne and Codeplay implemented a "header only" library which enables sycl::complex and associated math function in device code
  - —A PR has been created to merge it into DPCPP
  - —Derivative/port of LLVM complex

Credit

Thomas Applencourt (Argonne)
Brice Videau (Argonne)
Aidan Belton-Schure (Codeplay)
Gordon Brown (Codeplay)

Example

https://github.com/argonne-lcf/SyclCPLX

```
#include "sycl_ext_complex.hpp”

sycl::queue Q(sycl::gpu_selector{});
std::complex<double> i00{0.2,0.5};
std::complex<double> i01{0.2,0.3};
gpu_result =
sycl::malloc_shared<sycl::ext::cplx::complex<double>>(1,Q);
Q.single_task([=]() {
       //Using implicit cast from std::complex ->
sycl::ext::cplx::complex
     gpu_result[0] = sycl::ext::cplx::pow<double>(i00, i01);
     }
  ).wait();
```

Application Studies

NWChemEx: Portability of SYCL on ALCF Polaris

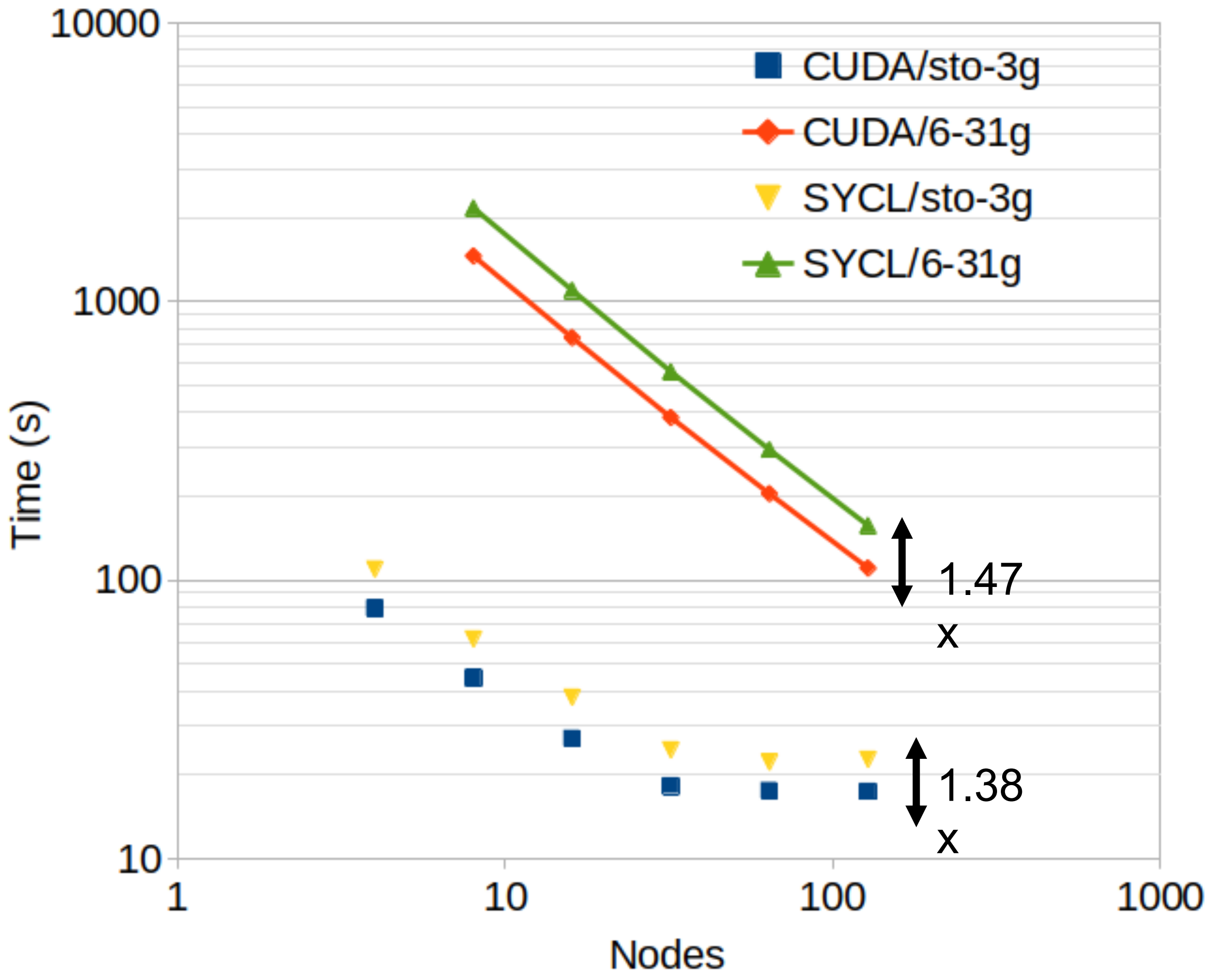
Credit Abhishek Bagusetty

Science

- **Application**: NWChemEx/TAMM, Coupled Cluster Singles, Doubles & Triples methods, CCSD(T) = CCSD + (T)
- **Code**: C++, CUDA, HIP, SYCL
- **Libraries**: cublas, rocblas, oneMKL
- CCSD energies contribution in CCSD(T) is mostly governed by vendor GEMM
- **(T) energies contribution forms the bottleneck in scaling a CCSD(T) computations**

Portability Performance Highlights

- SYCL performance for Nvidia devices is almost feature complete
- SYCL performance is about 1.38 – 1.47x slower in comparison to native CUDA on Nvidia A100
- This is a significant improvement over the past couple of months where the performance gap was about 8.5x
- Note: Performance characteristics of CCSD contribution were not included. Portable oneMKL (gemm) uses cublas for computations yielding similar performance as cublas APIs



Strong scaling – Contribution of timings in seconds for (T) kernel on ALCF Polaris, each node with 4 Nvidia A100 40 GB.
Nodes: 4, 8, 16, 32, 64, 128

sto-3g = 231 basis functions
6-31g = 424 basis functions

NWChemEx: Portability of SYCL on OLCF Crusher

Credit Abhishek Bagusetty

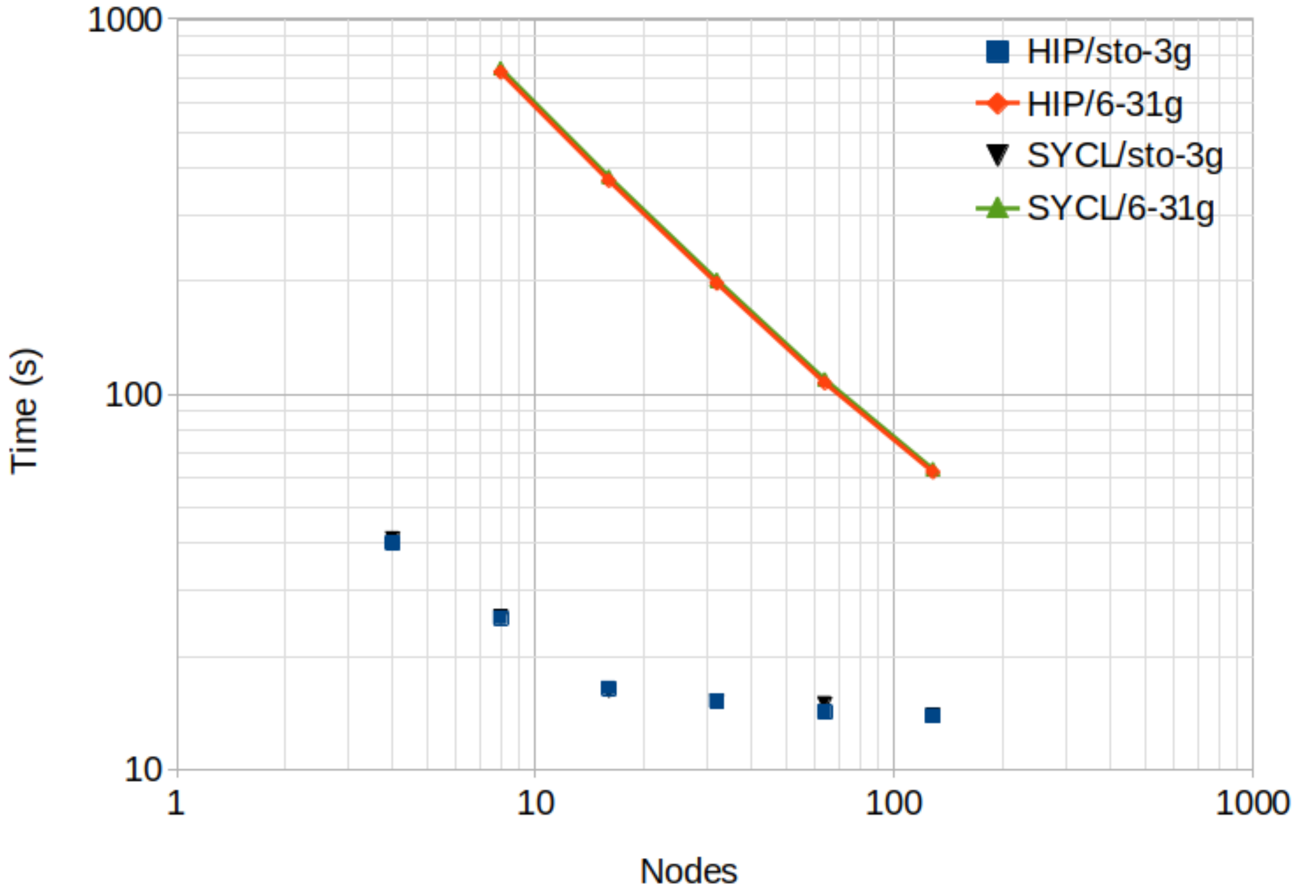
Portability Performance Highlights

- HIP plugin in LLVM is not as feature complete as CUDA plugin
- Though SYCL portability performance on AMD MI-250x GPU is almost similar to native HIP performance

- SYCL portability performance is about **98-99%** of native HIP on AMD MI-250x
- A significant improvement in merging the gap from 90-93% as measured in May 2022

- Timings are shown for the computationally bottle neck computation of triples (T) kernel of CCSD(T) calculation

- Note: Unlike CUDA, ROCM and SYCL ecosystem leverages LLVM providing better portability with optimizations.
-
- Note: Performance characteristics of CCSD contribution were not included. Portable oneMKL (gemm) uses rocblas for computations yielding similar performance with SYCL portable libraries for DGEMM



Strong scaling – Contribution of timings in seconds for (T) kernel on pre-production OLCF Crusher each node with 4 AMD MI-250x (i.e., 6 GCD). Nodes: 4, 8, 16, 32, 64, 128
sto-3g = 231 basis functions
6-31g = 424 basis functions

*SYCL built with rocm-5.1.0 : Dated Aug 18, 2022

Acknowledgment: Thanks to Codeplay Software Ltd., for providing the necessary SYCL plugins to CUDA and HIP backends, more importantly performance optimizations

Credit Abhishek Bagusetty

SuperLU

Status: In-development
Supports MPI, GPU (CUDA, HIP, SYCL)

- SuperLU consists:
  (a) hand-written device CUDA/HIP/SYCL kernels
  (b) Vendor BLAS: sgemm/dgemm/zgemm
- SYCL backend primary goal was initially for Aurora architecture
- Complete: Portable SYCL backend builds on OLCF Crusher, Perlmutter
- In progress: Unit testing, performance benchmarks

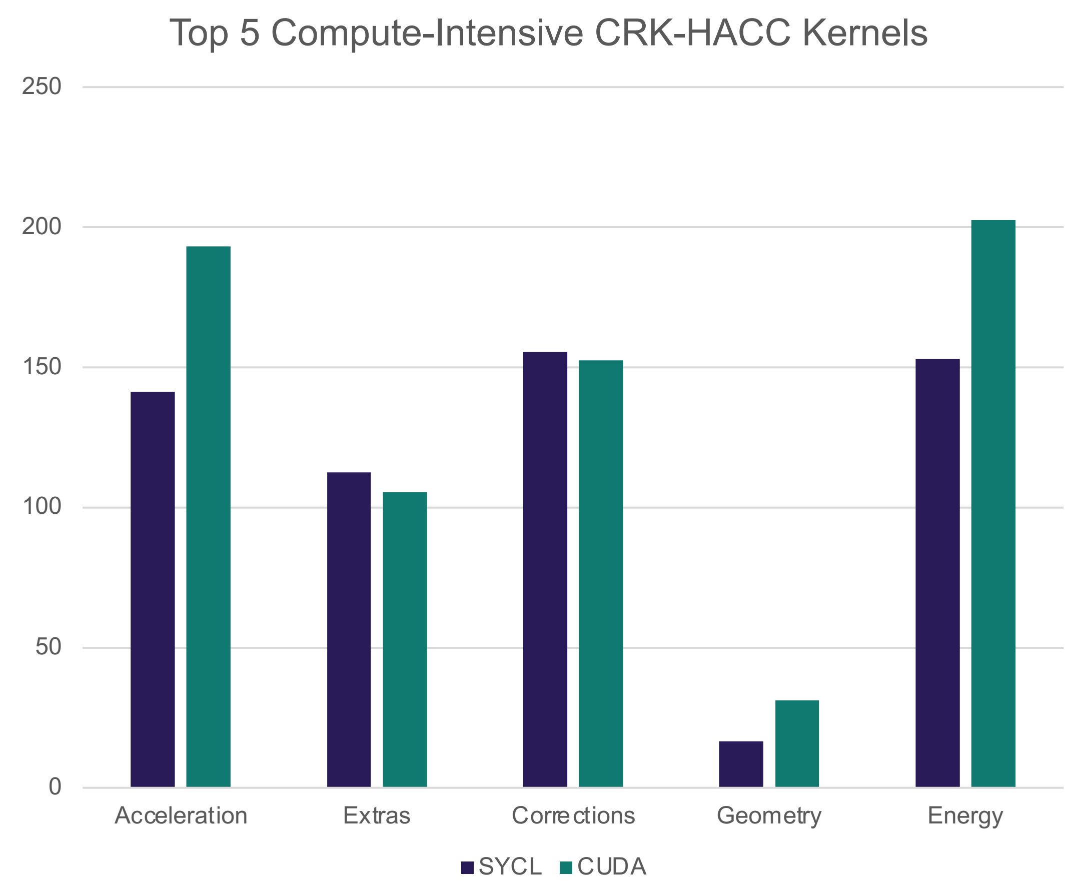
<u>Open-source, oneMKL Library</u>

	NVIDIA	AMD	Intel
BLAS	cuBLAS	rocBLAS	oneMKL
Linear Solvers	cuSOLVER	in-works (rocSOLVER)	oneMKL
Random Numbers	cuRAND	rocRAND	oneMKL
FFT	in-works (cuFFT)	in-works (rocFFT)	in-works (onemkl::dft)

CRK-HACC Results on A100: SYCL, CUDA

Credit Steve Rangel

- Original CUDA code used as the source for porting to SYCL using the Intel DPC++ compatibility tool, DPCT, with some manual tuning and rewriting (~10%).
- CUDA (11) compiled with nvcc.
- SYCL compiled with public Intel DPC++ compiler using CUDA backend.
- Kernels replayed through a testing harness with data taken from a simulation with 256^3 particles (~2GB of data on the GPU).
- Kernels are single-precision (FP32) and largely compute-bound with some use of FP32 atomic operations.
- Timing results obtained with CUDA events API and SYCL event profiling.



Argonne
NATIONAL LABORATORY

Closing

Conclusion

- SYCL has demonstrated performance portability between Intel, Nvidia and AMD hardware
  - Intel results when GPUs are no longer NDA

- Interested in feedback from applications developers on SYCL and Aurora
  - Contact Kevin Harms (harms@alcf.anl.gov) and Scott Parker (sparker@alcf.anl.gov)

- Interested in SYCL on Frontier
  - Contact David Bernholdt (bernholdtde@ornl.gov) and Balint Joo (joob@ornl.gov)

- Interested in SYCL on Perlmutter
  - Contact Brandon Cook (bgcook@lbl.gov)

- Labs are looking for interested applications that want to explore SYCL on Aurora, Frontier and Perlmutter

Acknowledgements

- ALCF – OLCF work (David Bernholdt, Balint Joo)
- NERSC – ALCF work (Brandon Cook)
- SYCL Complex (Thomas Applencourt, Brice Videau, Adian Belton-Schure, Gordon Brown)
- RAJA Perf Suite (Brian Homerding)
- NWChemEx / SuperLU (Abhishek Bagusetty)
- CRK-HACC (Steve Rangel)

- Codeplay Software Ltd. (Gordon Brown et. al.)

- *This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.*

- *This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.*

- *This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231*