Ginkgo A SPARSE LINEAR ALGEBRA LIBRARY FOR HPC

Karlsruher Institut für Technologie
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Hartwig Anzt, Natalie Beams, Terry Cojean, Fritz Göbel, Thomas Grützmacher, Aditya Kashi, Pratik Nayak, Tobias Ribizel, Yuhsiang M. Tsai

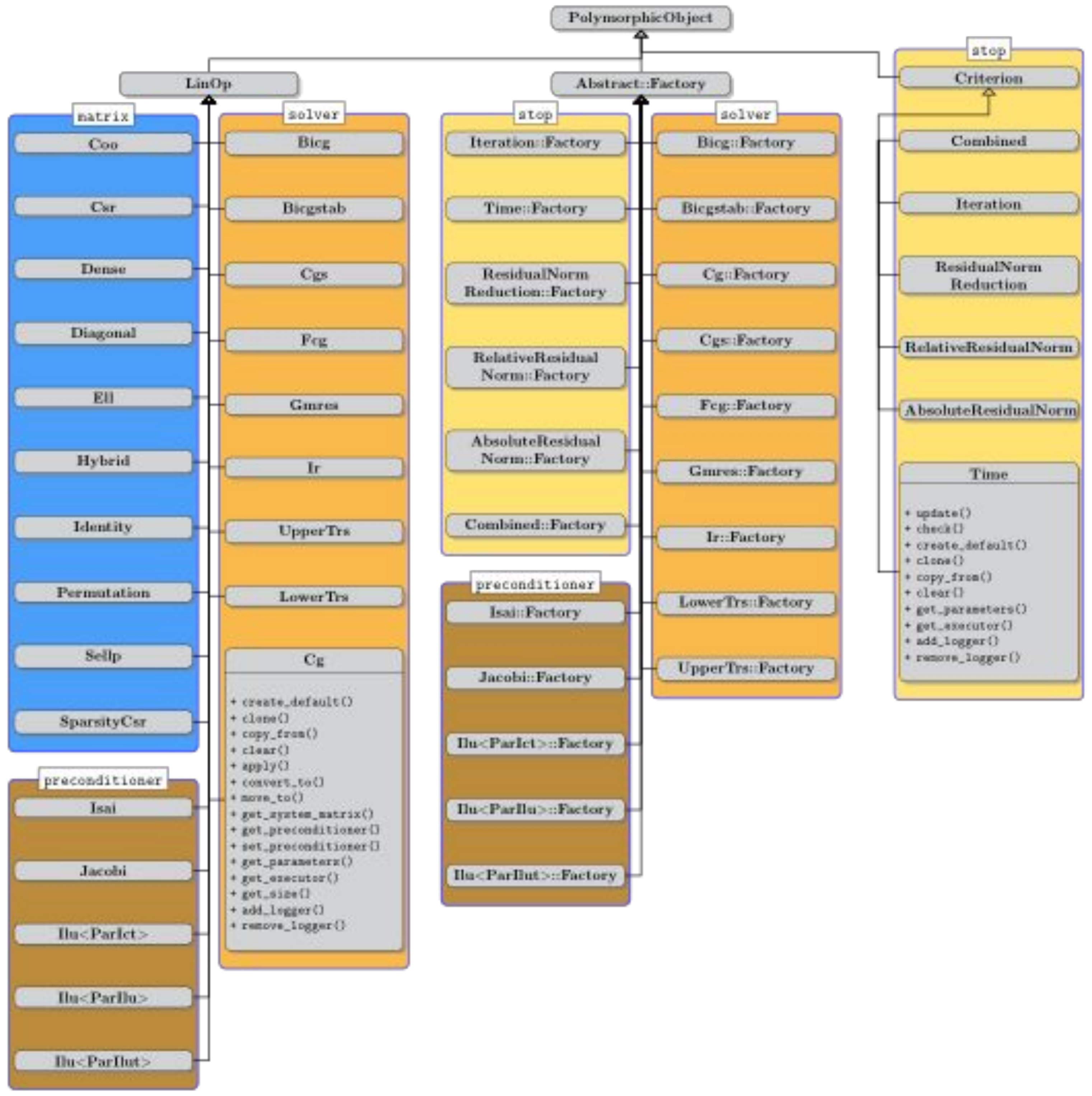
DESIGN

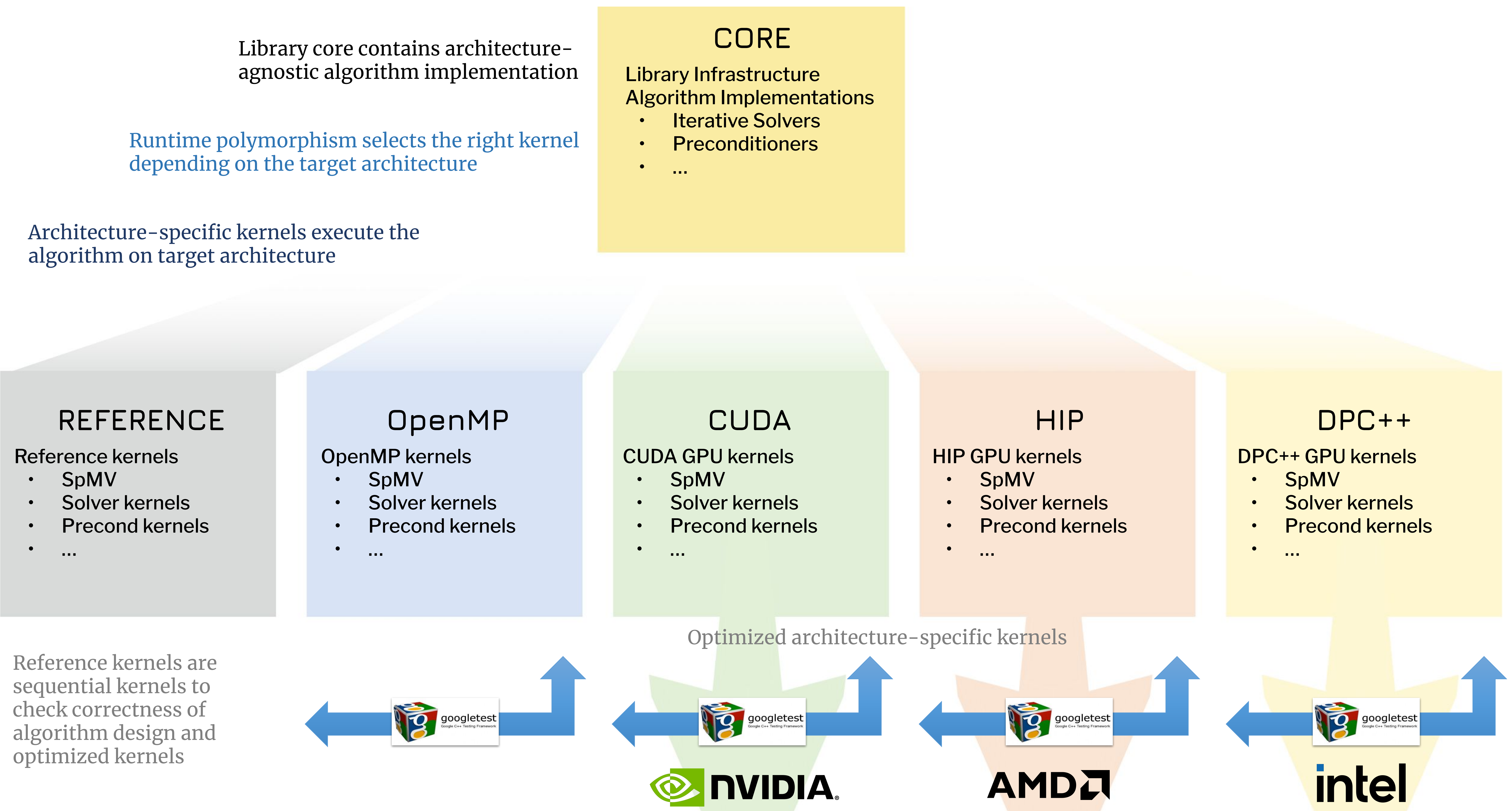
Ginkgo[1] is a C++ framework for sparse linear algebra. Using a universal linear operator abstraction, Ginkgo provides basic building blocks like the sparse matrix vector product for a variety of matrix formats, iterative solvers, and preconditioners. Ginkgo targets multi- and many-core systems, and currently features back-ends for AMD GPUs, Intel GPUs, NVIDIA GPUs, and OpenMP-supporting architectures. Core functionality is separated from hardware-specific kernels for easy extension to other architectures, with runtime polymorphism selecting the proper kernels.

SUSTAINABLE SOFTWARE DEVELOPMENT

Ginkgo is part of the Extreme-scale Scientific Software Stack (E4S) and the extreme-scale Software Development Kit (xSDK), and adopts the xSDK community policies for sustainable software development and high software quality. The source code of the Ginkgo library can be accessed in a public git repository on GitHub. Code development in Ginkgo is realized in a Continuous Integration / Continuous Benchmarking framework. GitLab runners are used on a private server where Docker images are used to provide different execution environments. To test the correct execution, each functionality is complemented by unit tests. The unit testing is realized using the Google Test framework.

COMPONENTS





Library core contains architecture-agnostic algorithm implementation

Runtime polymorphism selects the right kernel depending on the target architecture

Architecture-specific kernels execute the algorithm on target architecture

CORE

Library Infrastructure
Algorithm Implementations
- Iterative Solvers
- Preconditioners
- …

REFERENCE

Reference kernels
- SpMV
- Solver kernels
- Precond kernels
- …

OpenMP

OpenMP kernels
- SpMV
- Solver kernels
- Precond kernels
- …

CUDA

CUDA GPU kernels
- SpMV
- Solver kernels
- Precond kernels
- …

HIP

HIP GPU kernels
- SpMV
- Solver kernels
- Precond kernels
- …

DPC++

DPC++ GPU kernels
- SpMV
- Solver kernels
- Precond kernels
- …

Optimized architecture-specific kernels

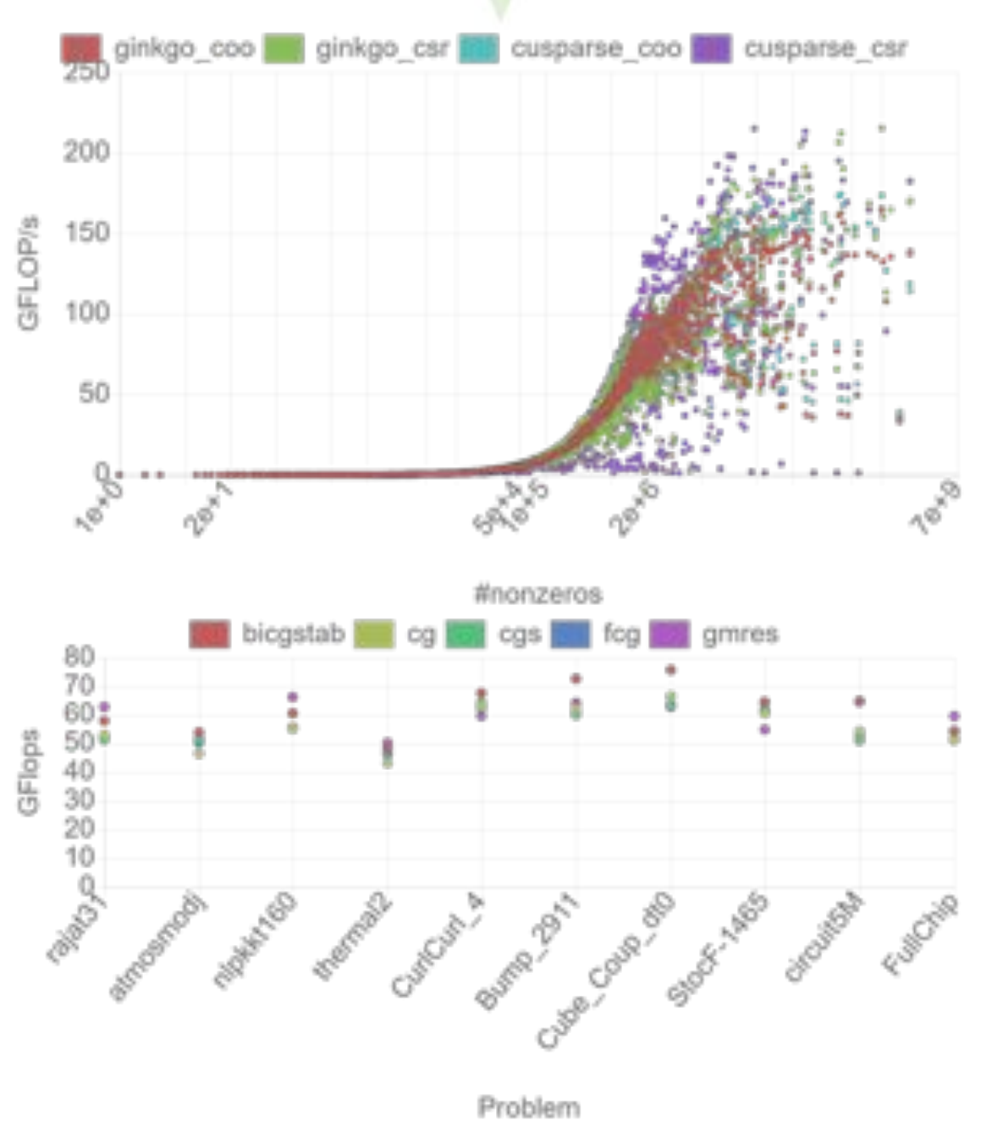
Reference kernels are sequential kernels to check correctness of algorithm design and optimized kernels

USAGE EXAMPLE

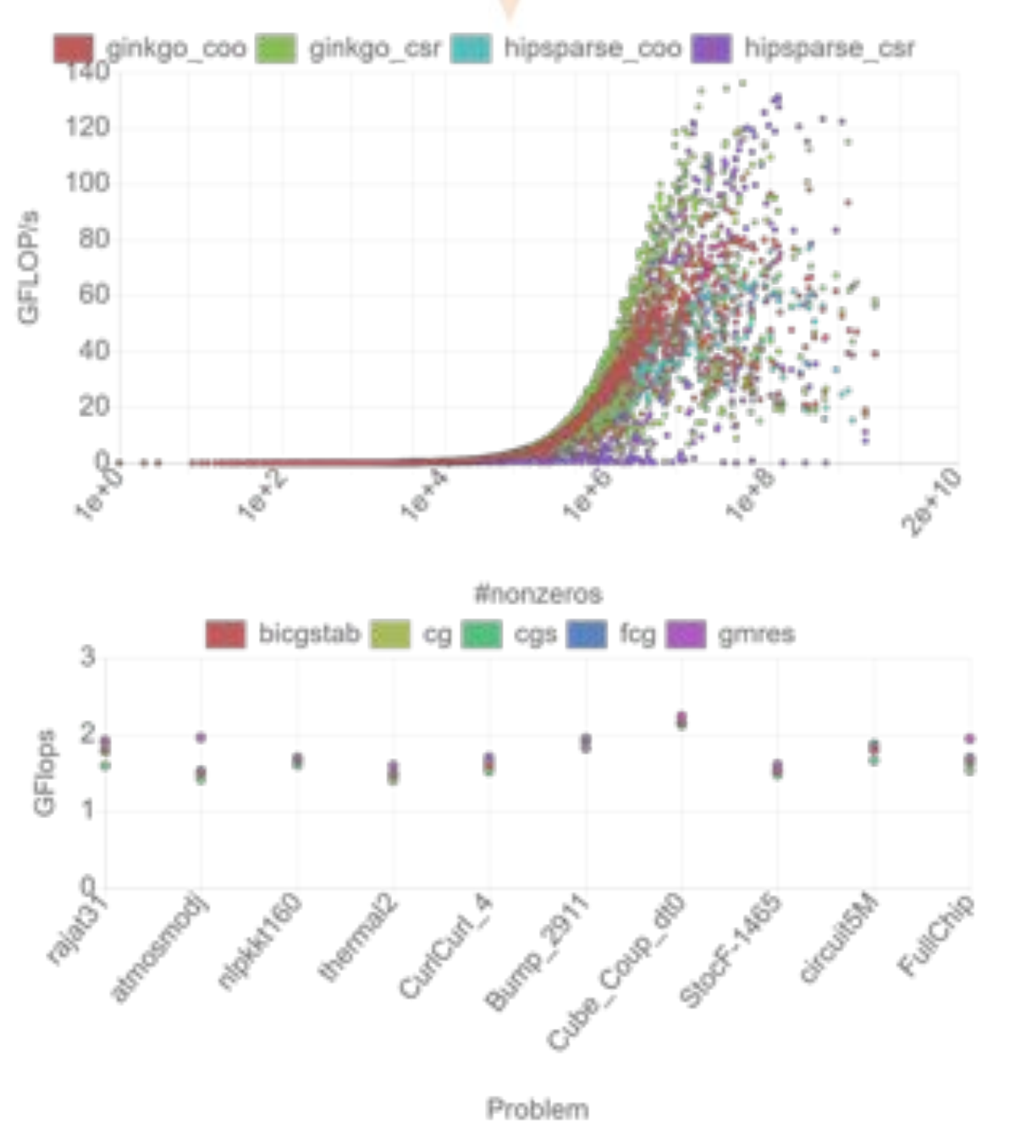
```cpp
#include <ginkgo.hpp>
#include <iostream>

int main()
{
    // Instantiate a CUDA executor
    auto gpu = gko::CudaExecutor::create(0, gko::OmpExecutor::create());
    // Read data
    auto A = gko::read<gko::matrix::Csr<>>(std::cin, gpu);
    auto b = gko::read<gko::matrix::Dense<>>(std::cin, gpu);
    auto x = gko::read<gko::matrix::Dense<>>(std::cin, gpu);
    // Create the solver
    auto solver =
        gko::solver::Cg<>::build()
            .with_preconditioner(gko::preconditioner::Jacobi<>::build().on(gpu))
            .with_criteria(
                gko::stop::Iteration::build().with_max_iters(20u).on(gpu),
                gko::stop::ResidualNormReduction<>::build()
                    .with_reduction_factor(1e-15)
                    .on(gpu))
            .on(gpu);
    // Solve system
    solver->generate(give(A))->apply(lend(b), lend(x));
    // Write result
    write(std::cout, lend(x));
}
```

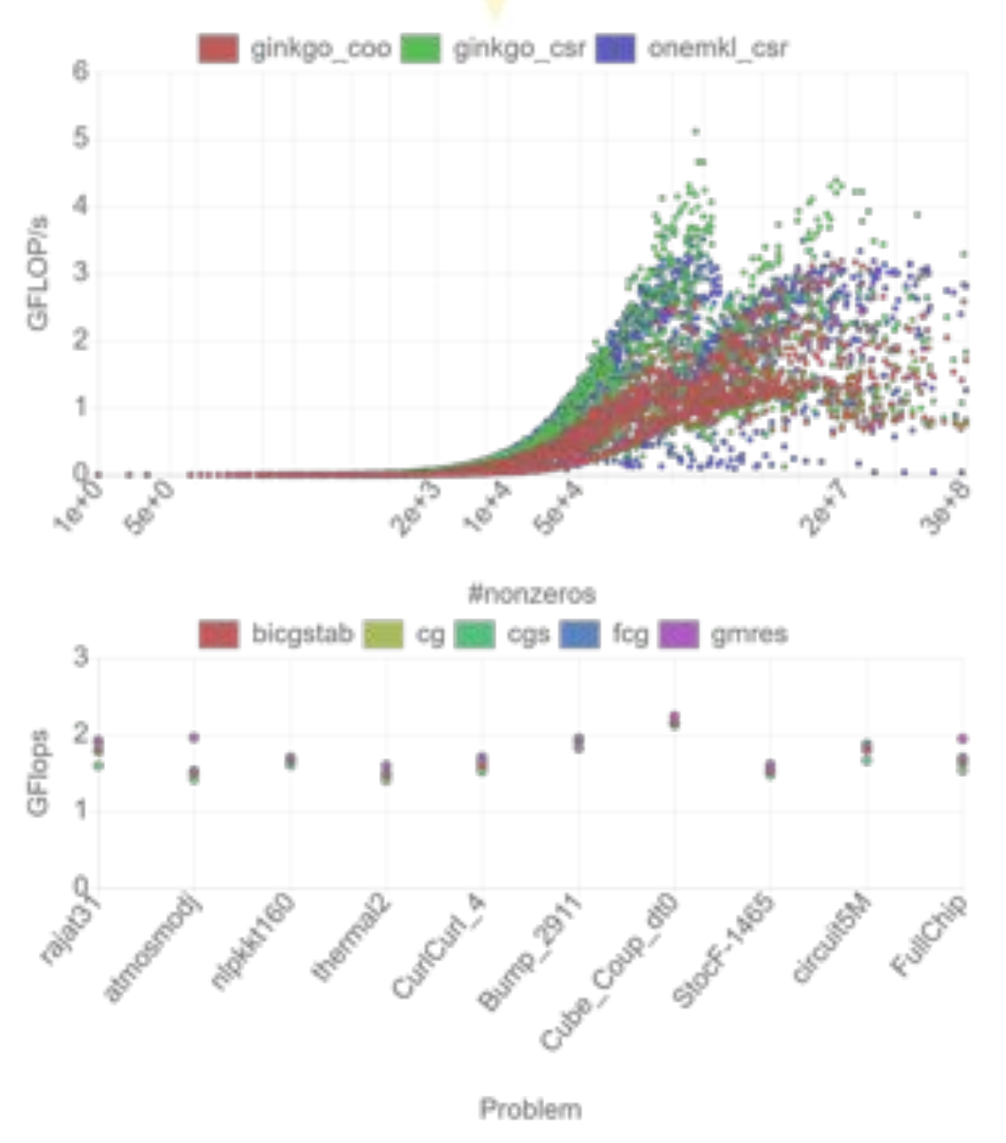
NVIDIA A100 GPU



AMD MI100 GPU



INTEL GEN.9 GPU



Ginkgo
https://ginkgo-project.github.io

xSDK

EXASCALE COMPUTING PROJECT

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.