Ginkgo 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

Usage example:
```c++
#include <ginkgo.hpp>

int main()
{
    // Instantiate a QD executor
    gko::exec::Executor& exe = gko::exec::create<0, gko::omp::Executor>::create();
    auto solver = gko::solver::Amg<
        gko::dist::Array<
            gko::dist::Matrix<
                gko::dist::SparseMatrix<
                    gko::dist::Array<
                        gko::dist::Device<float>,
                        gko::dist::Device<float>>>,
                gko::dist::Device<float>>>,
                gko::dist::Device<float>>,
                gko::dist::Device<float>>>::create(exe);
    // Create the solver
    solver->build();
    solver->apply(gko::solver::Amg<
        gko::dist::Array<
            gko::dist::Matrix<
                gko::dist::SparseMatrix<
                    gko::dist::Array<
                        gko::dist::Device<float>,
                        gko::dist::Device<float>>>,
                gko::dist::Device<float>>>,
                gko::dist::Device<float>>,
                gko::dist::Device<float>>>::create(exe));
    // Solve problem
    solver->apply(gko::solver::Amg<
        gko::dist::Array<
            gko::dist::Matrix<
                gko::dist::SparseMatrix<
                    gko::dist::Array<
                        gko::dist::Device<float>,
                        gko::dist::Device<float>>>,
                gko::dist::Device<float>>>,
                gko::dist::Device<float>>,
                gko::dist::Device<float>>>::create(exe));
    return 0;
}
```

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.