ICL Booth Talk
Supercomputing 2019

The Ginkgo Sparse Linear Algebra Package

Ginkgo

Hartwig Anzt

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Overview

Ginkgo

- Open-source C++ framework for sparse linear algebra.
- Sparse linear solvers, preconditioners, SpMV etc.
- Generic algorithm implementation:
  + reference kernels for checking correctness;
  + architecture-specific highly optimized kernels.
- Focused on Multicore and Manycore accelerators;
- Software quality and sustainability efforts guided by xSDK community policies:
  - open source, BSD
  - Doxygen
  - CMake Cross-platform Make
  - googletest Google C++ Testing Framework, GitLab CI

https://xsdk.info/

- Static polymorphism for templating precisions
  - ValueType (By default: Z,C,D,S), Integer
- Smart pointers to avoid memory leaks
- Runtime polymorphism for operators and kernels
  - Kernels have the same signature for different architectures
  - **Executor** determines which kernel is used

    Determines where Data lives & operations are executed
  - **LinOp** class for any linear operator:
    - Matrices
    - Solvers
    - Preconditioners
    - …

    generate
    apply
    …

Core Concept: Separate Algorithm from Kernels

Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Core

Library Infrastructure

Algorithm Implementations

- Iterative Solvers
- Preconditioners
- …

Architecture-specific kernels execute the algorithm on target architecture;

Kernels

- SpMV
- Solver kernels
- Precond kernels
- …

Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Core

Library Infrastructure

Algorithm Implementations

- Iterative Solvers
- Preconditioners
- …

Architecture-specific kernels execute the algorithm on target architecture;

Kernels

Reference

Reference kernels

- SpMV
- Solver kernels
- Precond kernels
- …

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

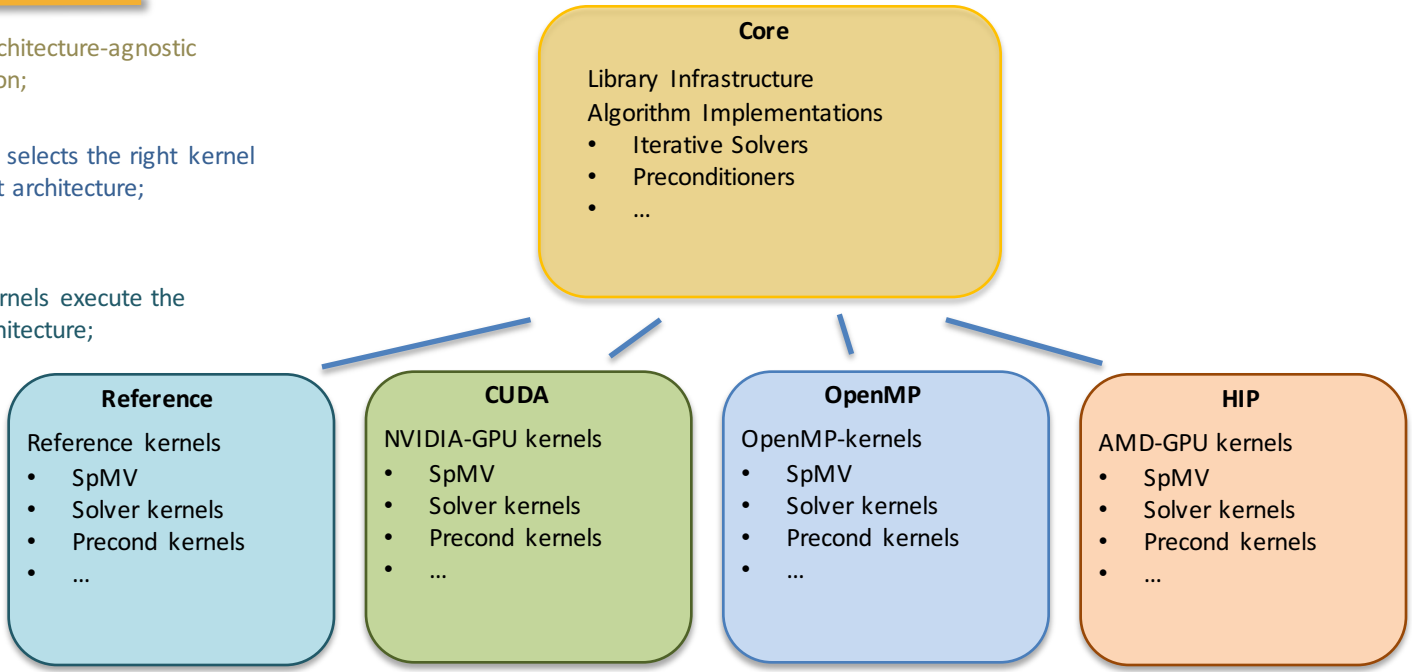
Core Concept: Separate Algorithm from Kernels

Ginkgo

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;



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

Optimized architecture-specific kernels;

Core Concept: Separate Algorithm from Kernels

Ginkgo

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
- ...

Kernels

Reference

Reference kernels

- SpMV
- Solver kernels
- Precond kernels
- ...

CUDA

NVIDIA-GPU kernels

- SpMV
- Solver kernels
- Precond kernels
- ...

OpenMP

OpenMP-kernels

- SpMV
- Solver kernels
- Precond kernels
- ...

HIP

AMD-GPU kernels

- SpMV
- Solver kernels
- Precond kernels
- ...

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

Optimized architecture-specific kernels;

Core Concept: Linear Operators

We express everything as Linear Operator and leverage C++ class inheritance.

Core Concept: Linear Operators

We express everything as Linear Operator and leverage C++ class inheritance.

Matrix-Vector Product

$$x := A \cdot b$$

Core Concept: Linear Operators

Ginkgo

We express everything as Linear Operator and leverage C++ class inheritance.

Matrix-Vector Product

$$x := A \cdot b$$

Preconditioner (for matrix A)

$$x := M^{-1} \cdot b$$

$$M^{-1} \approx A^{-1}$$

Core Concept: Linear Operators

Ginkgo

We express everything as Linear Operator and leverage C++ class inheritance.

Matrix-Vector Product

$$x := A \cdot b$$

Preconditioner (for matrix A)

$$x := M^{-1} \cdot b$$

$$M^{-1} \approx A^{-1}$$

Solver (for system $Ax = b$)

$$x := S \cdot b$$

$$S \approx A^{-1}$$

Core Concept: Linear Operators

Ginkgo

We express everything as Linear Operator and leverage C++ class inheritance.

Matrix-Vector Product

$$x := A \cdot b$$

Preconditioner (for matrix A)

$$x := M^{-1} \cdot b$$

$$M^{-1} \approx A^{-1}$$

$$M^{-1} = \Pi(A)$$

Solver (for system $Ax = b$)

$$x := S \cdot b$$

$$S \approx A^{-1}$$

$$S = \Sigma(A)$$

Core Concept: Linear Operators

Ginkgo

We express everything as Linear Operator and leverage C++ class inheritance.

Matrix-Vector Product	Preconditioner (for matrix A)	Solver (for system $Ax = b$)
$x := A \cdot b$	$x := M^{-1} \cdot b$	$x := S \cdot b$
	$M^{-1} \approx A^{-1}$	$S \approx A^{-1}$
	$M^{-1} = \Pi(A)$	$S = \Sigma(A)$

All of them can be expressed as

- Application of a linear operator* (LinOp) $L : \mathbb{F}^m \to \mathbb{F}^m$

Core Concept: Linear Operators

Ginkgo

We express everything as Linear Operator and leverage C++ class inheritance.

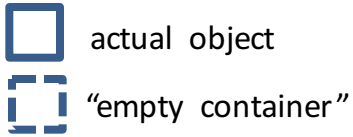
Matrix-Vector Product	Preconditioner (for matrix A)	Solver (for system $Ax = b$)
$x := A \cdot b$	$x := M^{-1} \cdot b$	$x := S \cdot b$
	$M^{-1} \approx A^{-1}$	$S \approx A^{-1}$
	$M^{-1} = \Pi(A)$	$S = \Sigma(A)$

All of them can be expressed as

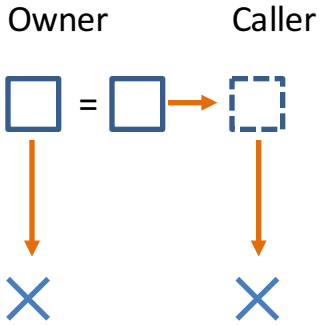
- Application of a linear operator* (LinOp) $L : \mathbb{F}^m \to \mathbb{F}^m$
- (non-linear) transformation applied to linear operator (Factory) $\Phi : \mathbb{L}^{mn}(\mathbb{F}) \to \mathbb{L}^{kl}(\mathbb{F})$

Memory Management: Smart Pointers

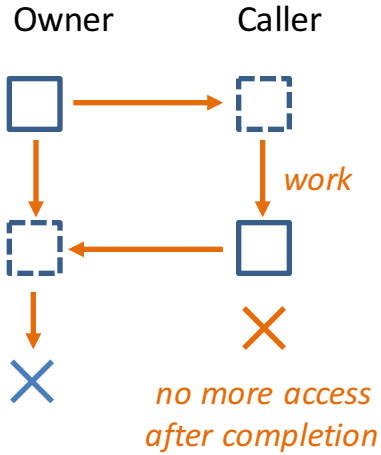
Ginkgo



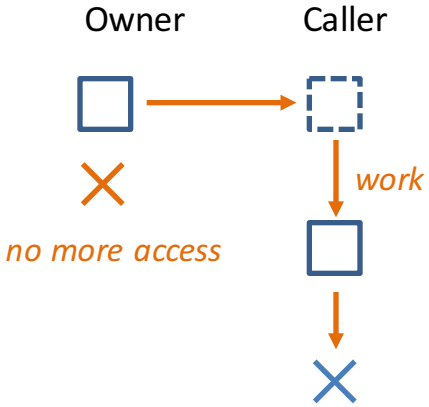
Clone



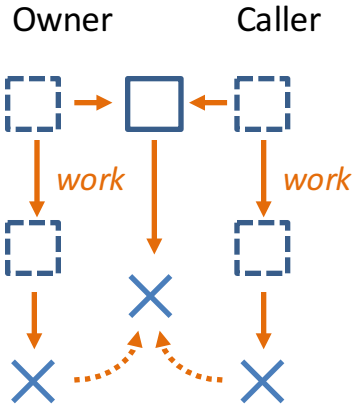
Lend



Give



Share



Ginkgo Functionality (version 1.1.0)

Ginkgo

Matrix Formats
- Dense
- CSR
- COO
- ELL
- SELLP
- hybrid

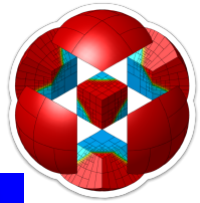
Krylov Solvers
- BiCGSTAB
- CG
- CGS
- Flexible CG
- GMRES

Preconditioners
- (Block-) Jacobi
- Adaptive Precision Block-Jacobi
- ILU(0)

Hardware Backends
- Reference (sequential, CPU)
- OpenMP
- CUDA
- HIP

Supported OS
- Linux
- MacOS
- Windows

Software Integration
- MFEM
- deal.II



deal.II

Ginkgo Functionality (version 1.1.0)

Ginkgo

Include Ginkgo functionality

Control Ginkgo solver

Generate deal.ii problem

Generate Ginkgo executor

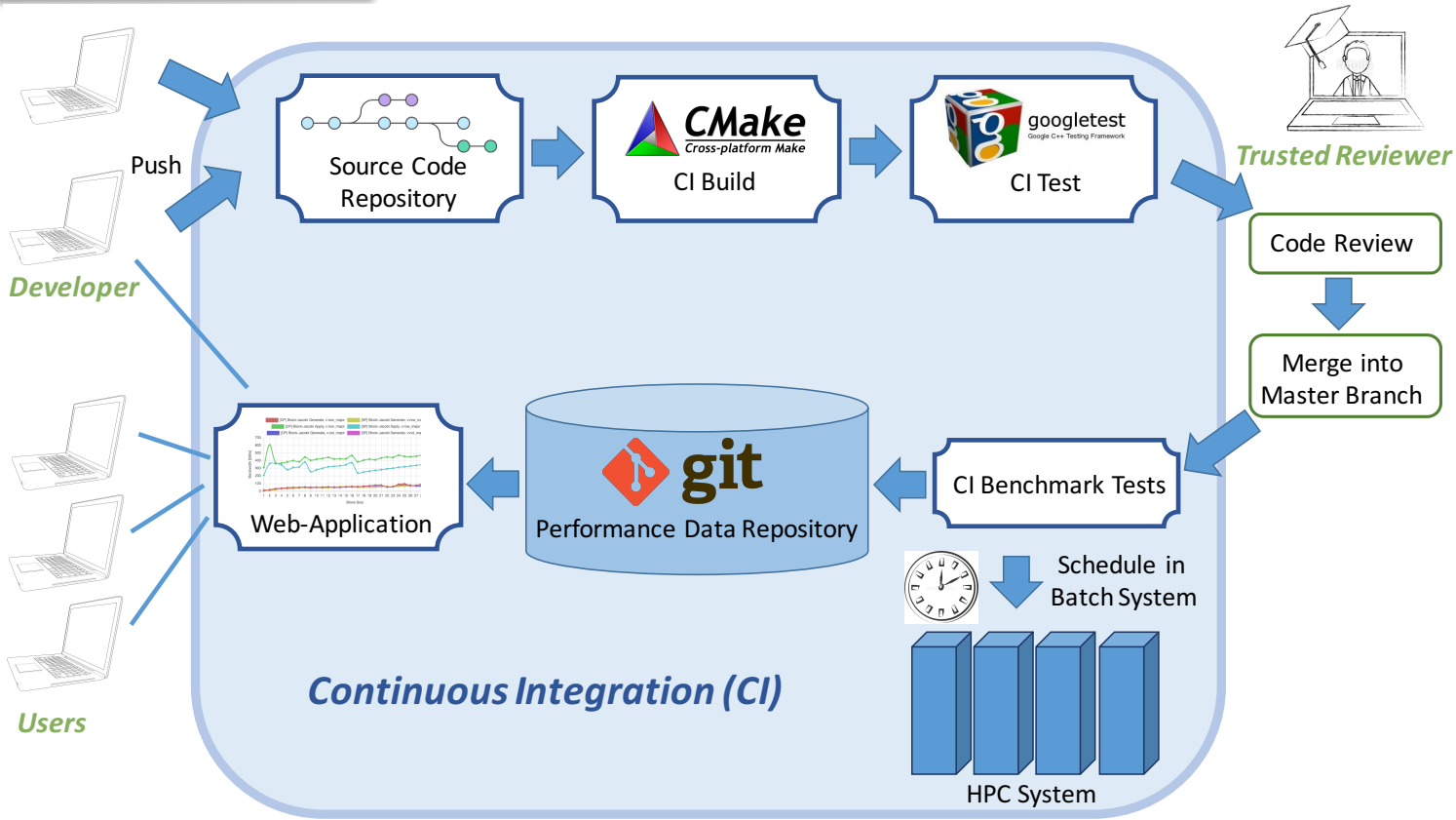
Generate Ginkgo solver

Solve linear system

```
#include <deal.II/lac/ginkgo_solver.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/vector.h>
#include <deal.II/lac/vector_memory.h>

#include "../testmatrix.h"
#include "../tests.h"

#include <iostream>
#include <typeinfo>

int main(int argc, char **argv)
{
    // Set solver parameters
    SolverControl control(200, 1e-6);

    const unsigned int size = 32;
    unsigned int       dim  = (size - 1) * (size - 1);

    // Setup a simple matrix
    FDMatrix        testproblem(size, size);
    SparsityPattern structure(dim, dim, 5);
    testproblem.five_point_structure(structure);
    structure.compress();
    SparseMatrix<double> A(structure);
    testproblem.five_point(A);

    Vector<double> f(dim);
    f = 1.;
    Vector<double> u(dim);
    u = 0.;

    // Instantiate a CUDA executor.
    auto gpu = gko::CudaExecutor::create(0, gko::OmpExecutor::create());
    // Use ginkgo to solve the system on the gpu using the CG solver.
    GinkgoWrappers::SolverCG                    solver(control, gpu );
    // Solves the system and copies the data back to deal.ii's solution variable.
    solver.solve(A, u, f);
}
```

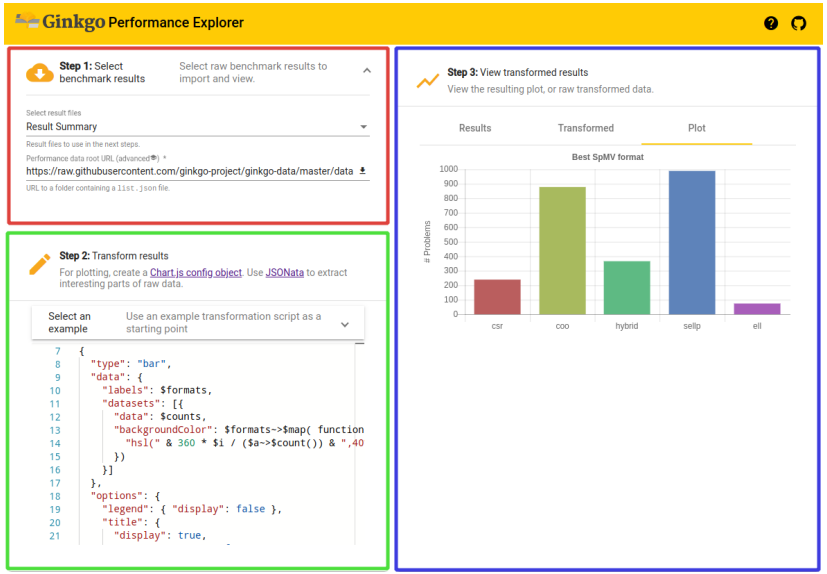
Continuous Integration & Continuous Benchmarking

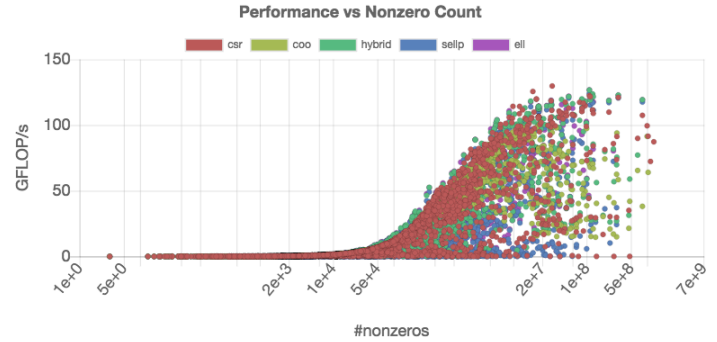
Ginkgo

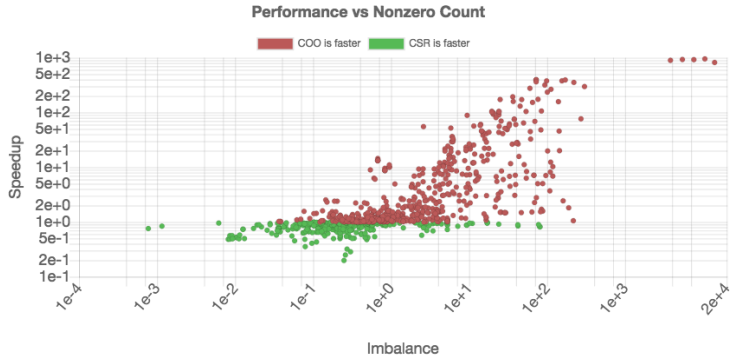


Ginkgo Performance Explorer (GPE)

Allows anyone to access & visualize Ginkgo Performance data in a browser.

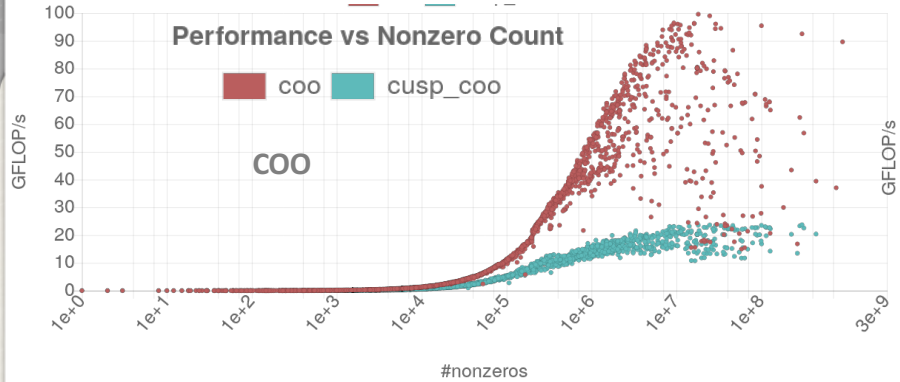


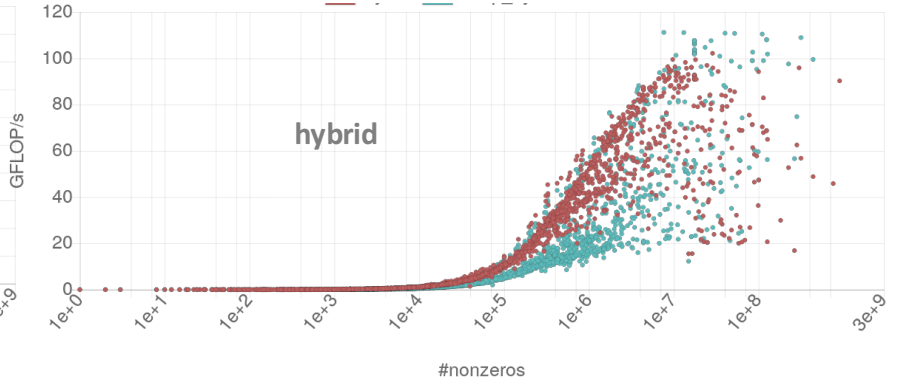


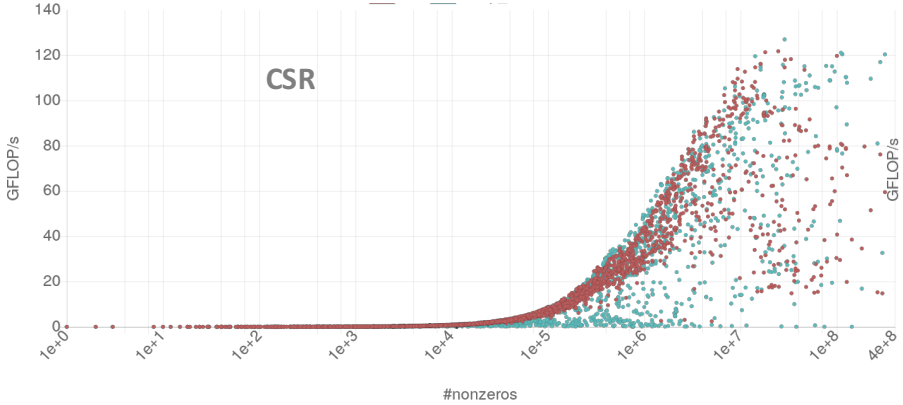


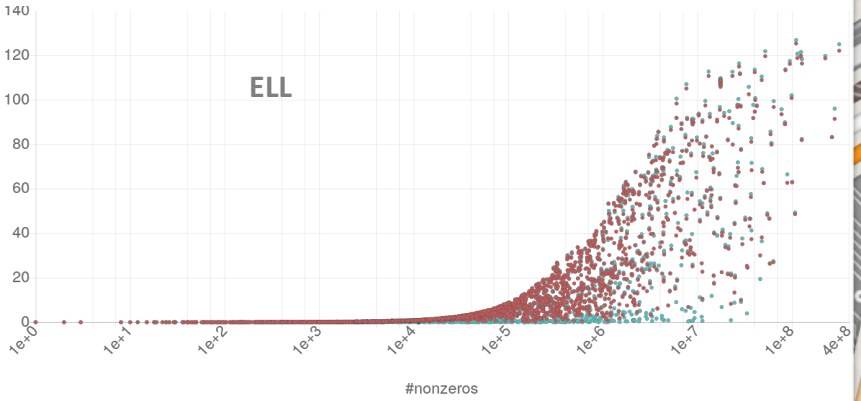
https://ginkgo-project.github.io/gpe/

Ginkgo SpMV Performance CUDA on V100

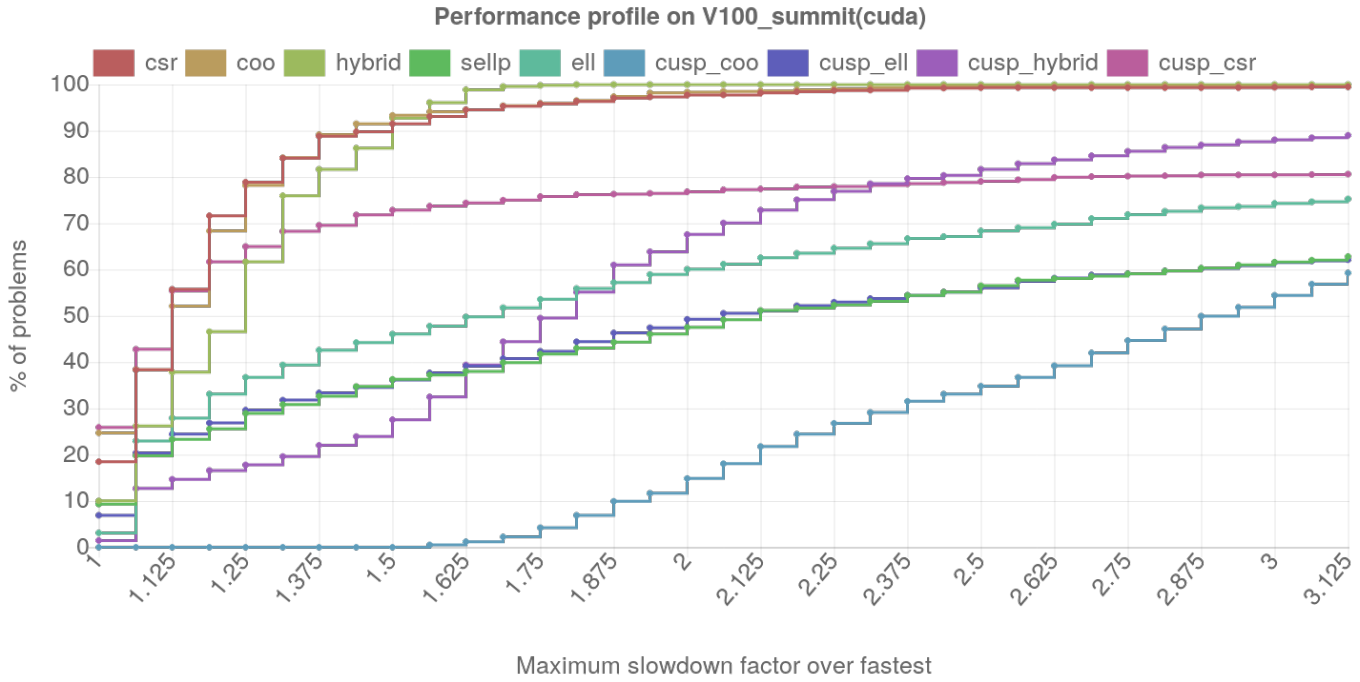






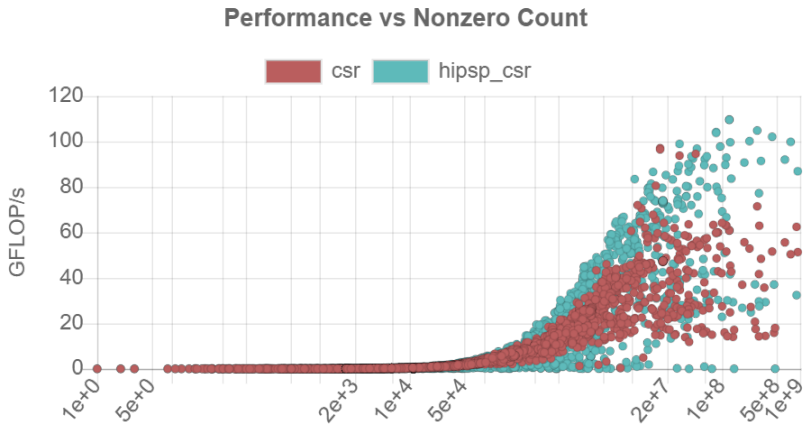


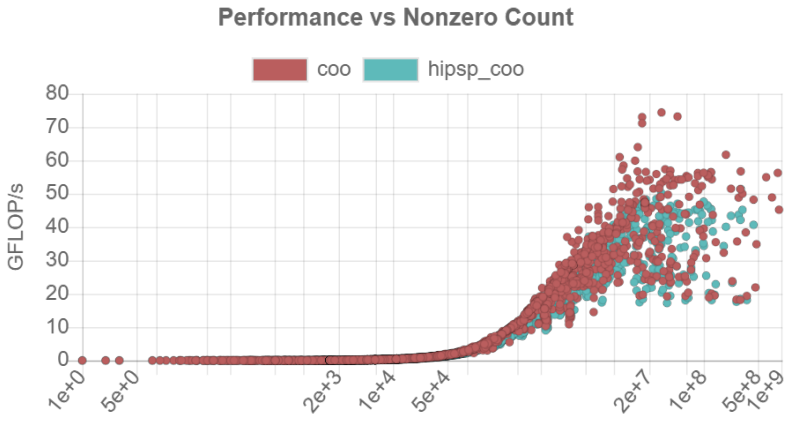
Ginkgo SpMV Performance CUDA on V100

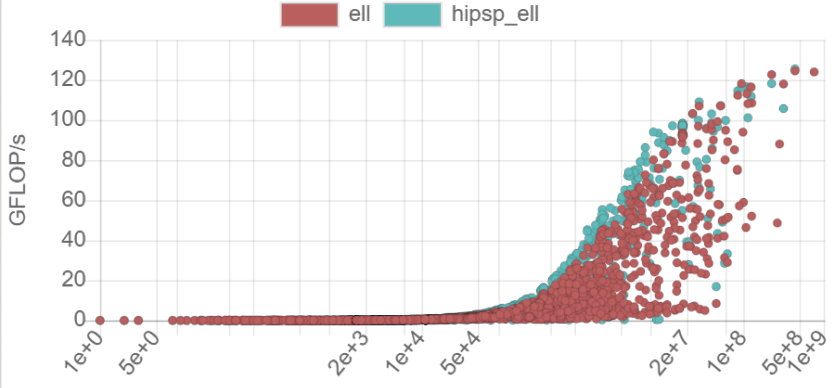


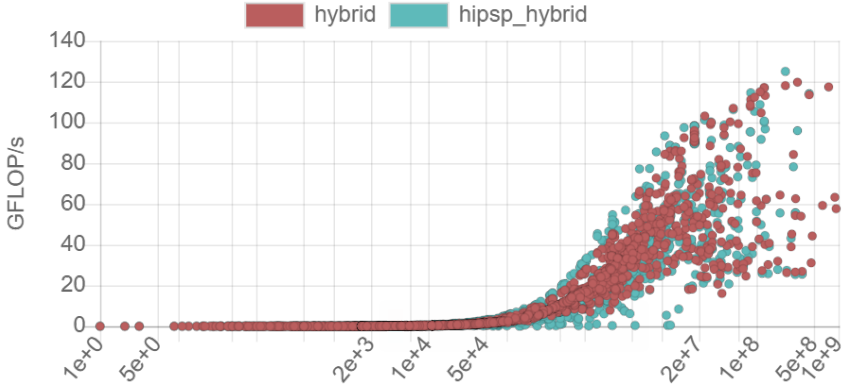
Anzt, Cojean, Chen, Dongarra, Flegar, Nayak, Tomov, Tsai, Wang: **Load-balancing Sparse Matrix Vector Product Kernels on GPUs**, TOPC, 2019.

Ginkgo SpMV Performance HIP on Radeon 7

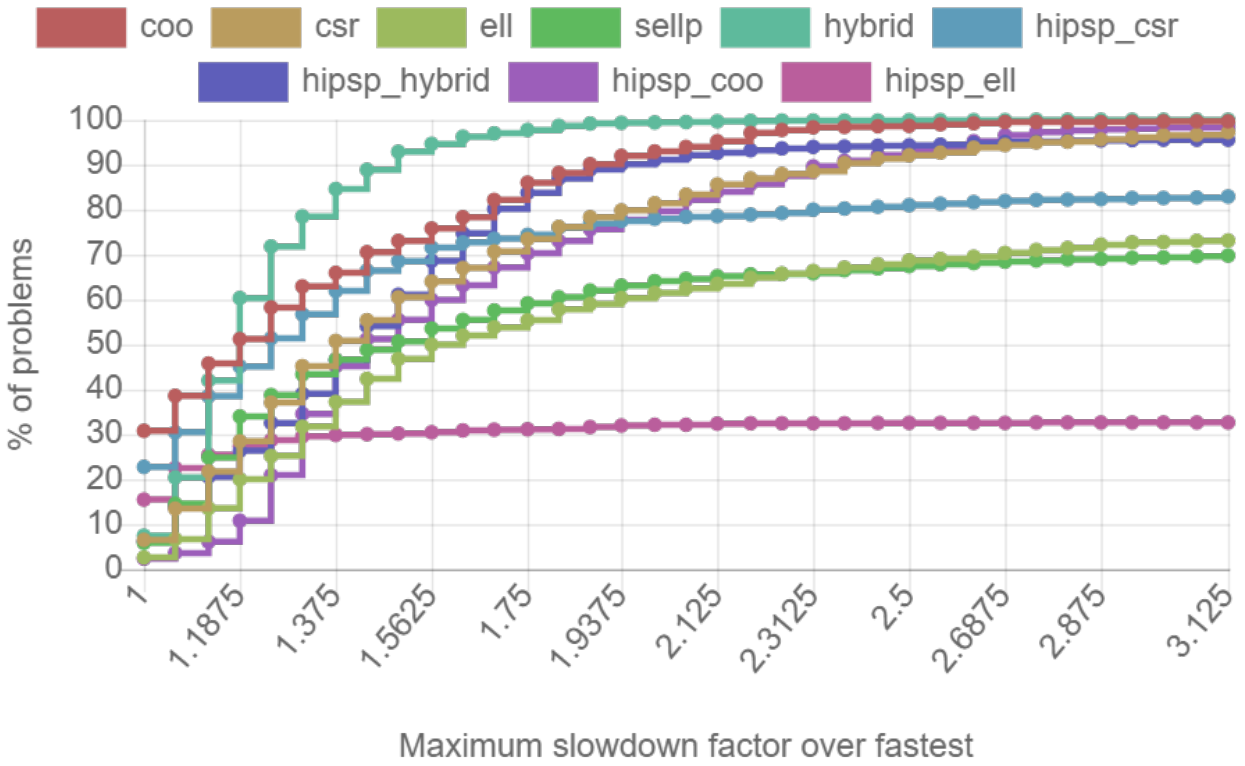








Ginkgo SpMV Performance HIP on Radeon 7

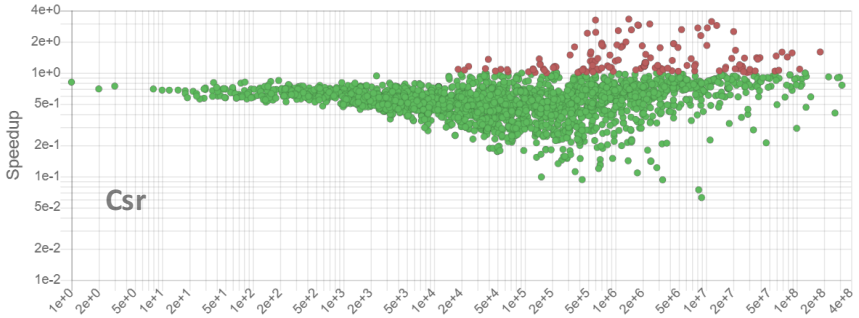


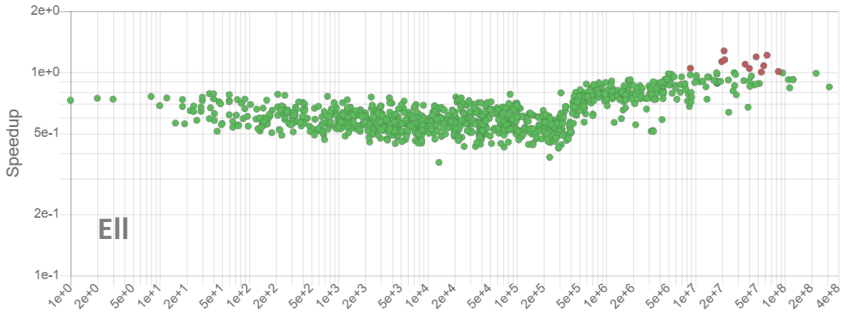
SpMV Performance Comparison V100 vs. Radeon 7

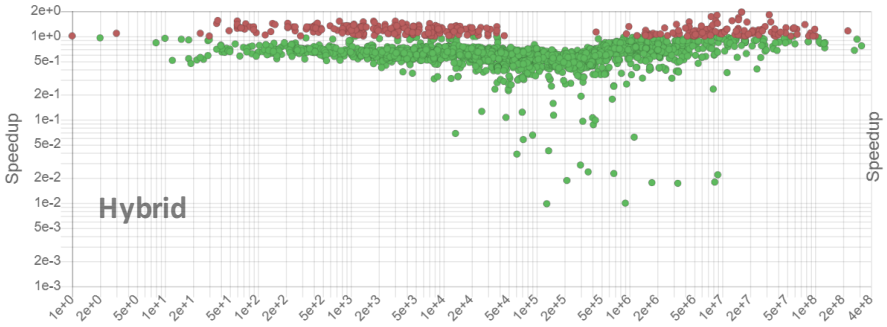
Ginkgo

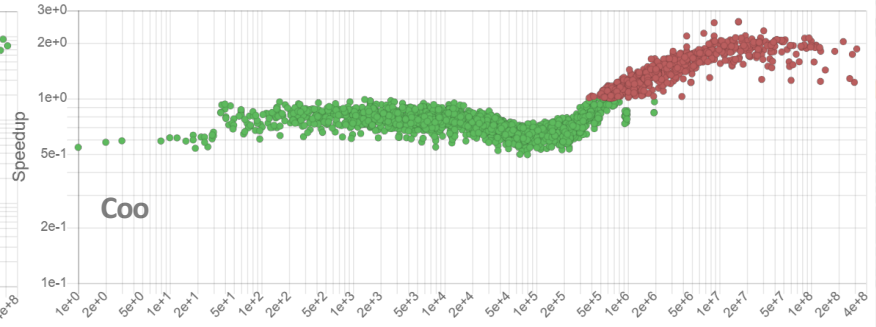
cuSPARSE on V100 is faster

HIPSparse on Radeon 7 is faster







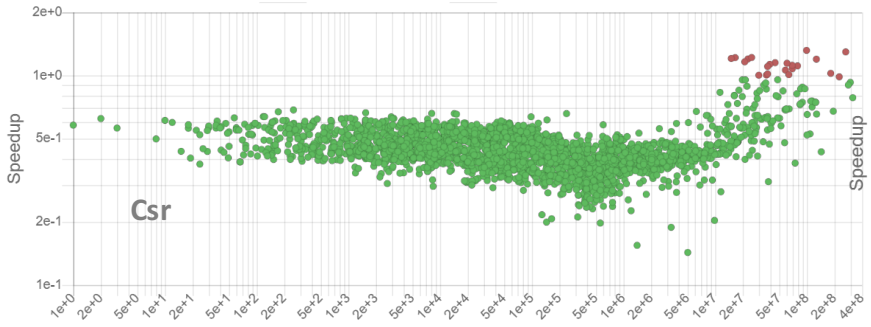


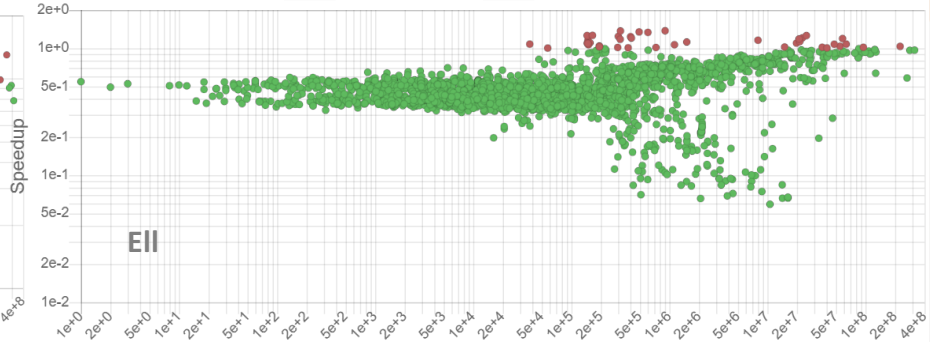
SpMV Performance Comparison V100 vs. Radeon 7

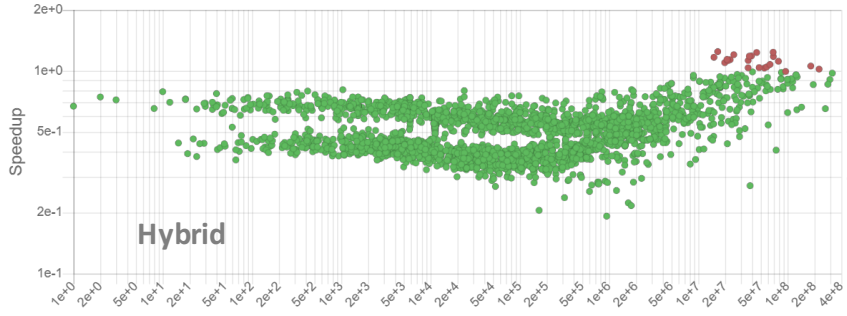
Ginkgo

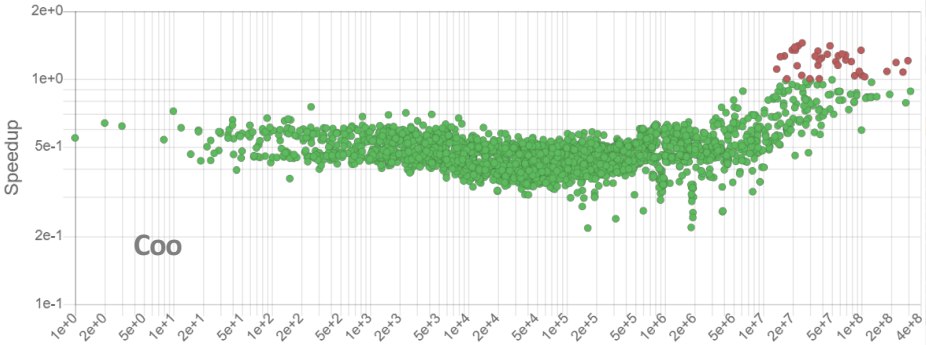
Ginkgo on V100 is faster

Ginkgo on Radeon 7 is faster









ICL

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Past, Present, Future…

Ginkgo

05/2017	Start library design discussions
05/2017	Initial C++ usage tests
10/2017	*"First bits in the bucket"* https://github.com/ginkgo-project/ginkgo
11/2017	Ginkgo name decision
02/2018	Continuous Integration (CI)
07/2018	Ginkgo Performance Explorer (GPE) https://ginkgo-project.github.io/gpe/
09/2018	Continuous Benchmarking (CB)
09/2018	deal.ii Finite Element Software integration example
12/2018	Ginkgo Project Webpage https://ginkgo-project.github.io/
01/2019	PAPI SDE integration
05/2019	Ginkgo v.1.0.0 release
07/2019	Ginkgo approved for xSDK release v.0.5.0
10/2019	Ginkgo v.1.1.0 release
10/2019	MFEM Finite Element Software integration example
11/2019	*HIP executor*
01/2020	*PETSc integration*
01/2020	*Trilinos integration*
06/2020	*FEAST sparse eigensolver*
06/2020	*Multi-GPU executor?*

NVIDIA

deal.II

AMD

Past, Present, Future…

Ginkgo

05/2017	Start library design discussions
05/2017	Initial C++ usage tests
10/2017	*"First bits in the bucket"* https://github.com/ginkgo-project/ginkgo
11/2017	Ginkgo name decision
02/2018	Continuous Integration (CI)
07/2018	Ginkgo Performance Explorer
09/2018	Continuous Benchmarking (CI
09/2018	deal.ii Finite Element Softwar
12/2018	Ginkgo Project Webpage http
01/2019	PAPI SDE integration
05/2019	Ginkgo v.1.0.0 release
07/2019	Ginkgo approved for xSDK rel
10/2019	Ginkgo v.1.1.0 release
10/2019	MFEM Finite Element Softwa
11/2019	*HIP executor*
01/2020	*PETSc integration*
01/2020	*Trilinos integration*
06/2020	*FEAST sparse eigensolver*
06/2020	*Multi-GPU executor?*

Join the Pack!