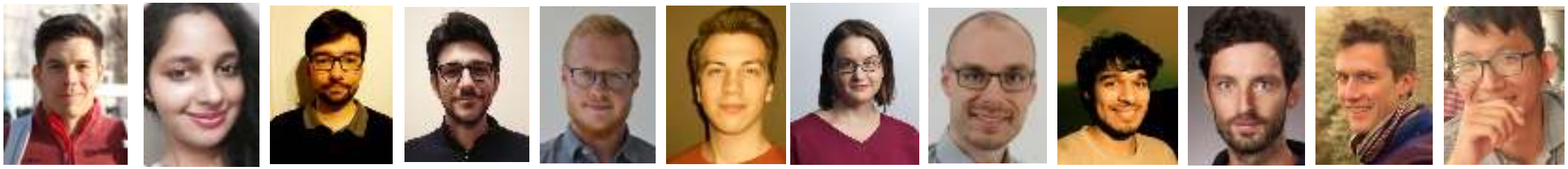
35th Workshop on Sustained Simulation Performance, HLRS, Stuttgart

Sustaining Simulation Performance in the US Exascale Computing Project – A tale from the trenches

Approved for public release

Hartwig Anzt, University of Tennessee



U.S. DEPARTMENT OF ENERGY | Office of Science

Designing an ECP library for sustaining simulation performance

MAGMA SPARSE

MAGMA-sparse as a “child” of MAGMA explores the development of sparse linear algebra for NVIDIA GPUs.

Limitations:

- *C code with hand-written build system*
- *Sparse unit testing*
- *Focus on NVIDIA GPUs*
- *Design-specific limitations (flexibility/extensibility)*

Designing an ECP library for sustaining simulation performance

MAGMA SPARSE

MAGMA-sparse as a “child” of MAGMA explores the development of sparse linear algebra for NVIDIA GPUs.

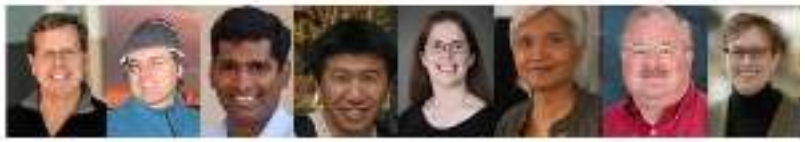
Design considerations for Ginkgo

- Platform Portability
- Performance
- Rapid integration of new algorithms
- xSDK / E4S Community Policies
- BSSw expertise / experience
- Modern C++
- CI/CD and unit testing
- Open source & permissive licensing

IDEAS productivity

better scientific software

Ginkgo



Designing an ECP library for sustaining simulation performance

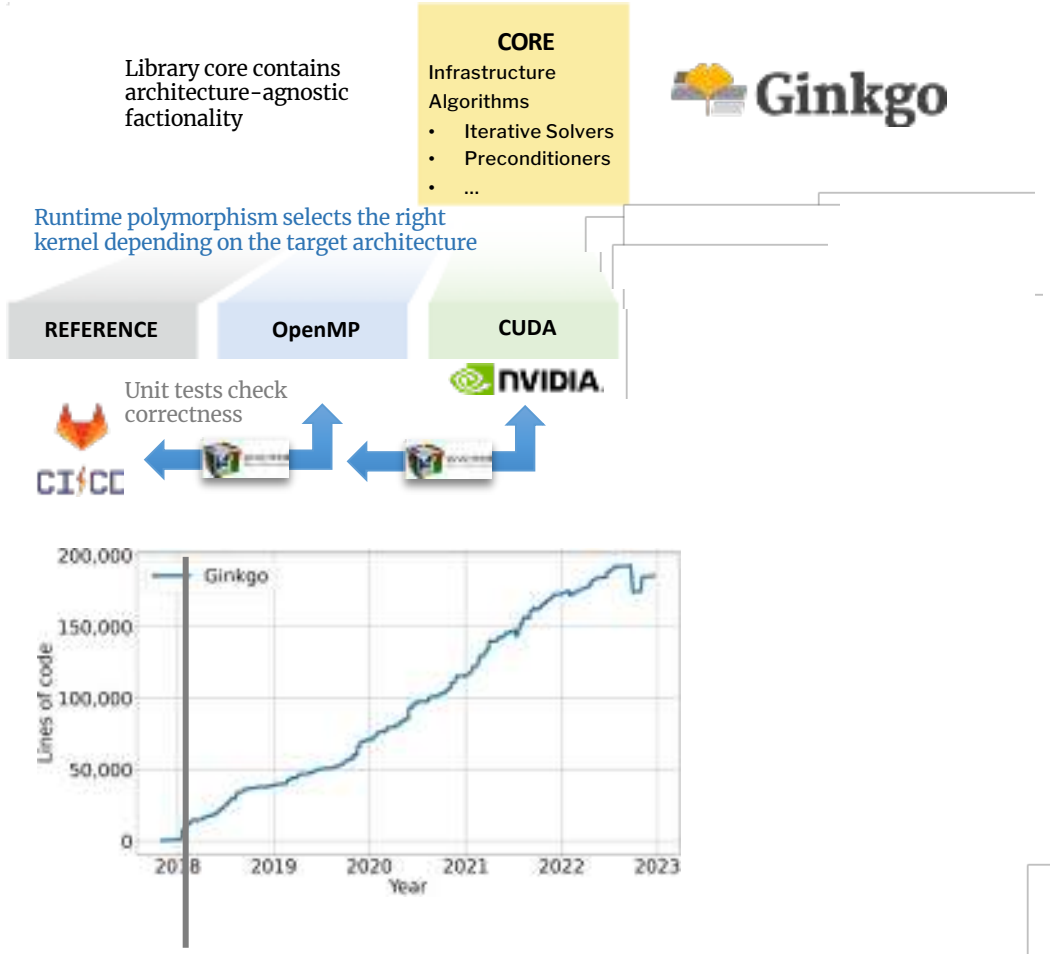
MAGMA SPARSE

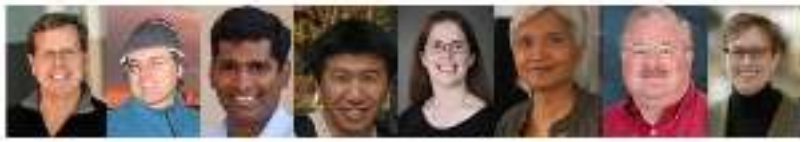
MAGMA-sparse as a “child” of MAGMA explores the development of sparse linear algebra for NVIDIA GPUs.

Design considerations for Ginkgo

- Platform Portability
- Performance
- Rapid integration of new algorithms
- xSDK / E4S Community Policies
- BSSw expertise / experience
- Modern C++
- CI/CD and unit testing
- Open source & permissive licensing

Before the first line of code is written, we spend a year on whiteboard discussions.





The LinOp abstraction

Linear Operator Interface

We express everything as Linear Operator

- Internally we leverage C++ class inheritance.
- Applications can apply any functionality as a linear operator

Matrix Vector Product | Preconditioner (for matrix A) | Solver (for system $Ax = b$)

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

$M := A^{-1}$ | $S := A^{-1}$

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

All of them can be expressed as

Application of a linear operator (LinOp) $L : \mathcal{V}^n \to \mathcal{V}^m$



Library core contains architecture-agnostic factionality

CORE

Infrastructure

Algorithms

- Iterative Solvers
- Preconditioners
- ...

Ginkgo

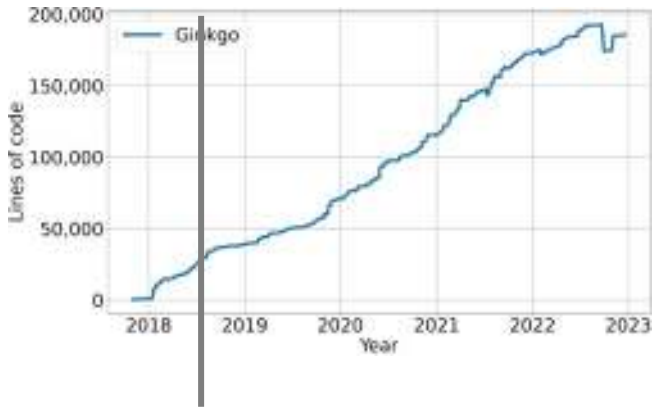
Runtime polymorphism selects the right kernel depending on the target architecture

REFERENCE | **OpenMP** | **CUDA**

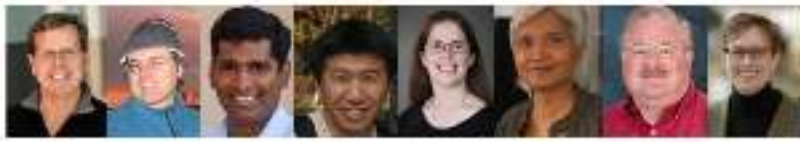
NVIDIA

Unit tests check correctness

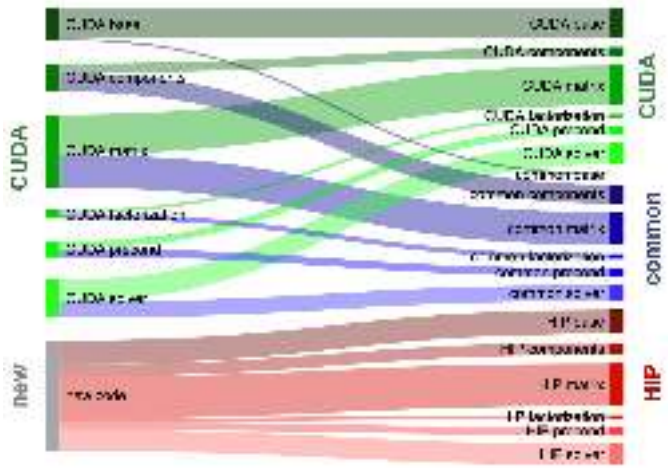
CI/CD



	Functionality	OMP	CUDA
Basic	SpMV	✓	✓
	SpMM	✓	✓
	SpGeMM	✓	✓
Krylov solvers	BiCG	✓	✓
	BiCGSTAB	✓	✓
	CG	✓	✓
	CGS	✓	✓
	GMRES	✓	✓
	IDR	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓
	ILU/IC		✓
	Parallel ILU/IC	✓	✓
	Parallel ILUT/ICT	✓	✓
	Sparse Approximate Inverse	✓	✓



Extending to AMD GPUs



Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

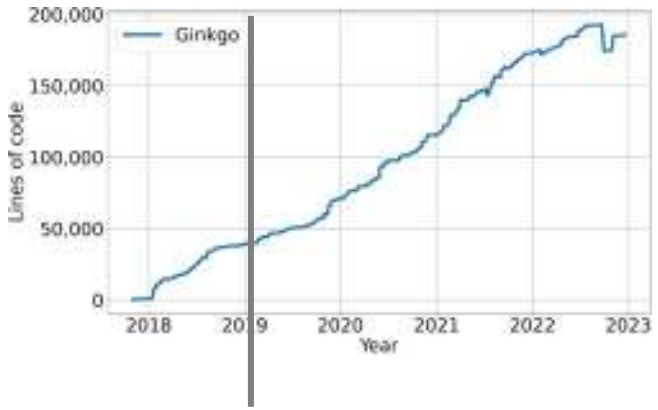
Architecture-optimized kernels

REFERENCE	OpenMP	CUDA	HIP

NVIDIA AMD

Unit tests check correctness

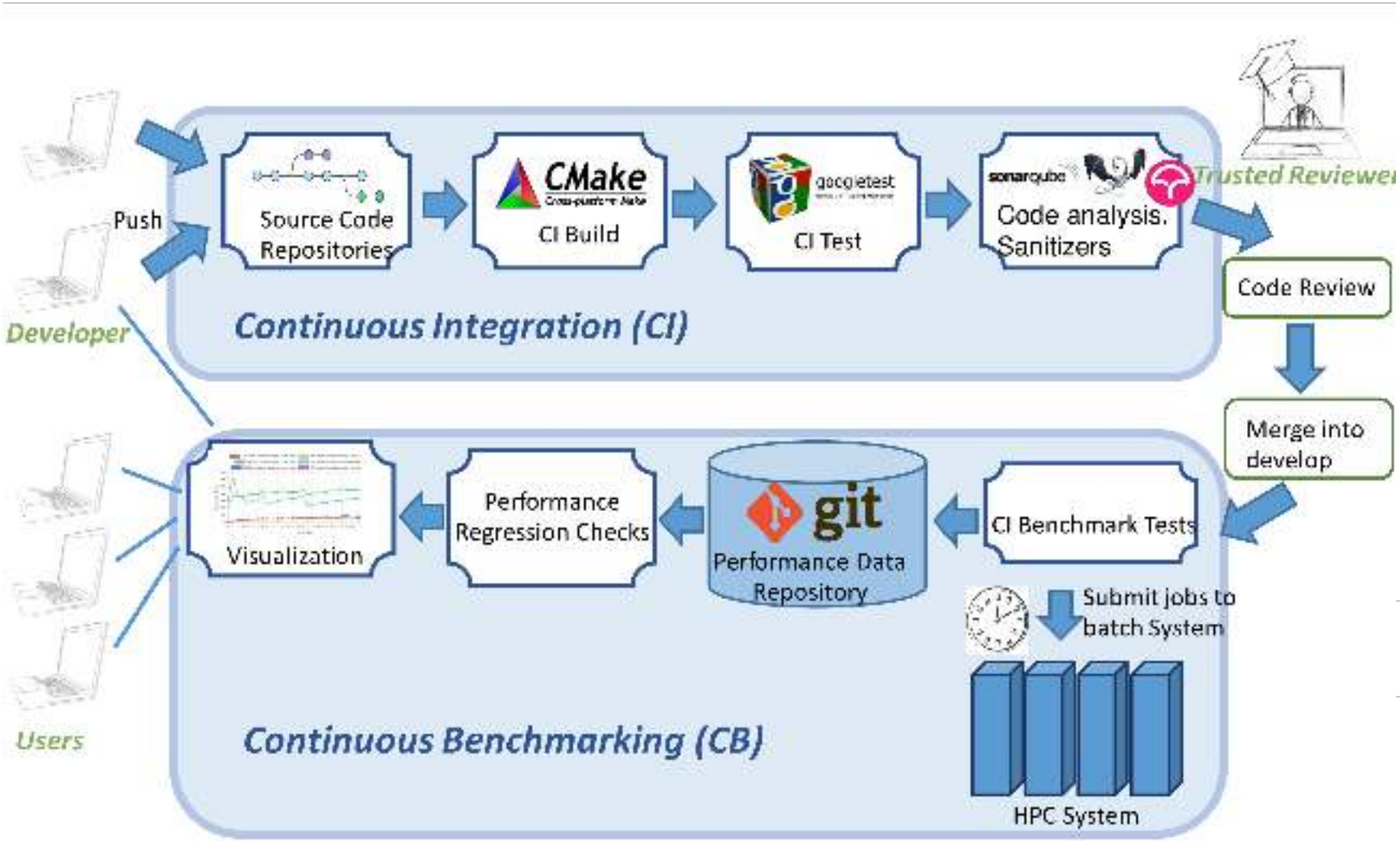
CI/CD



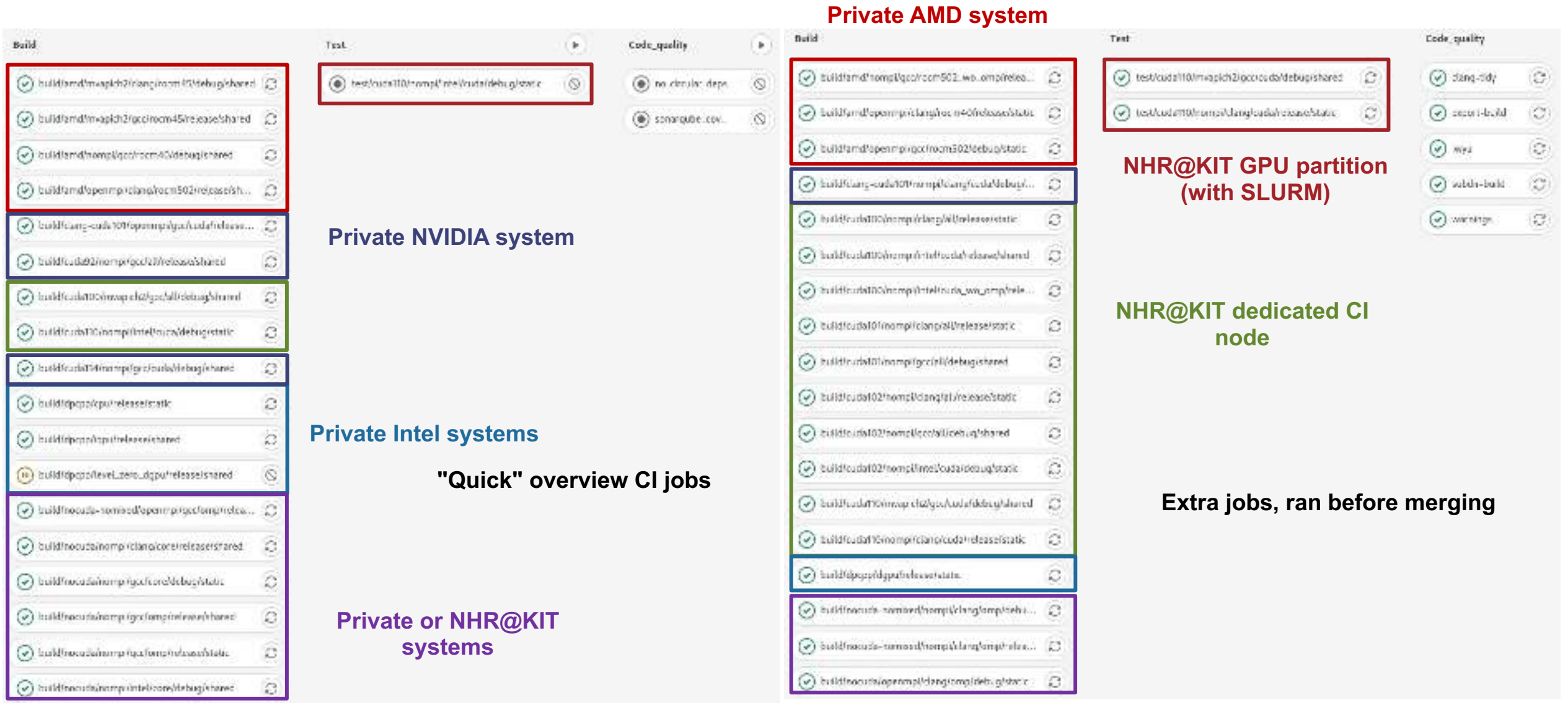
	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓

Spack xSDK E4S

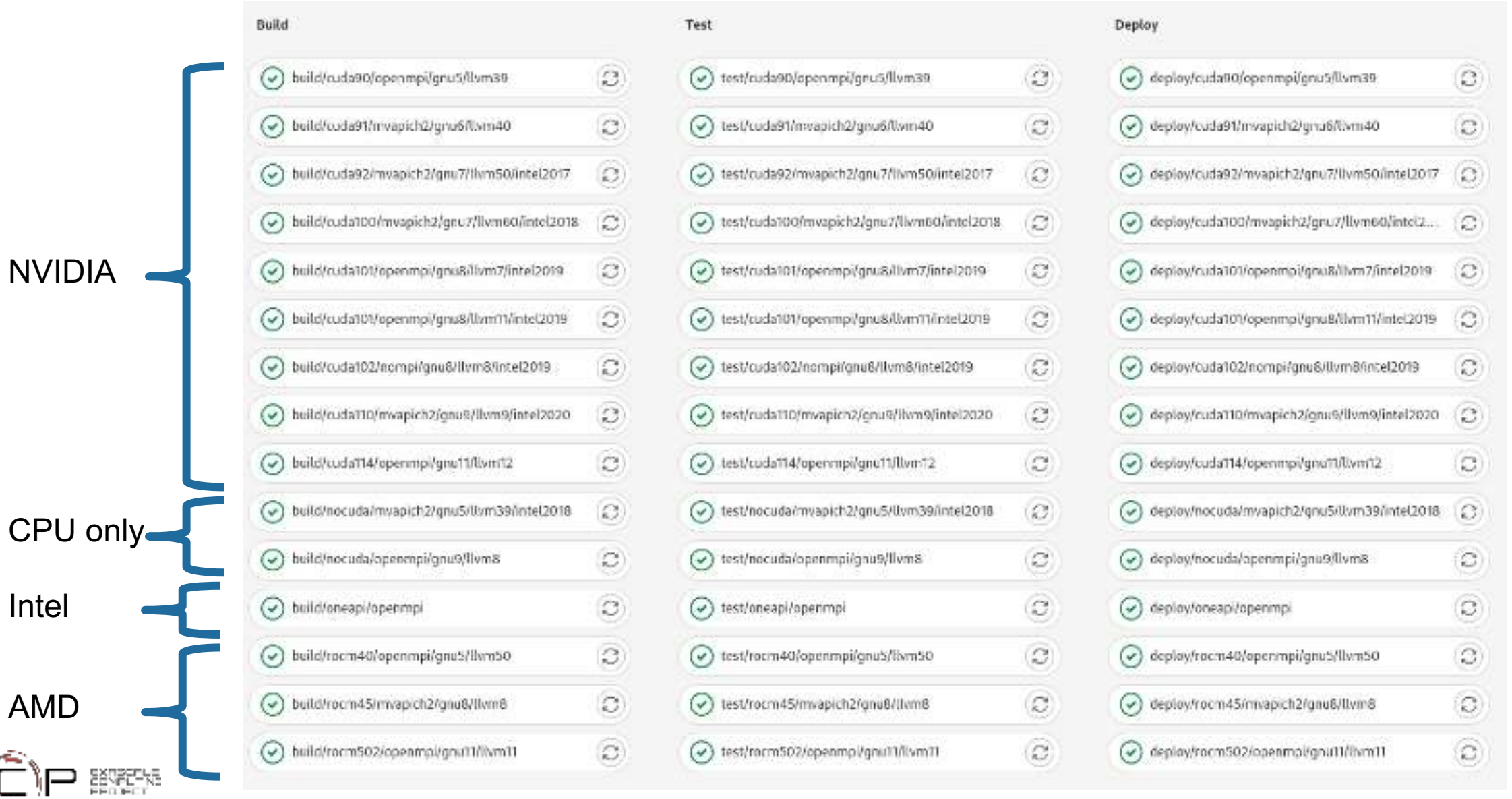
Sustainable software development & CI/CD



Sustainable software development & CI/CD



Sustainable software development & CI/CD



Input from the “first customer”

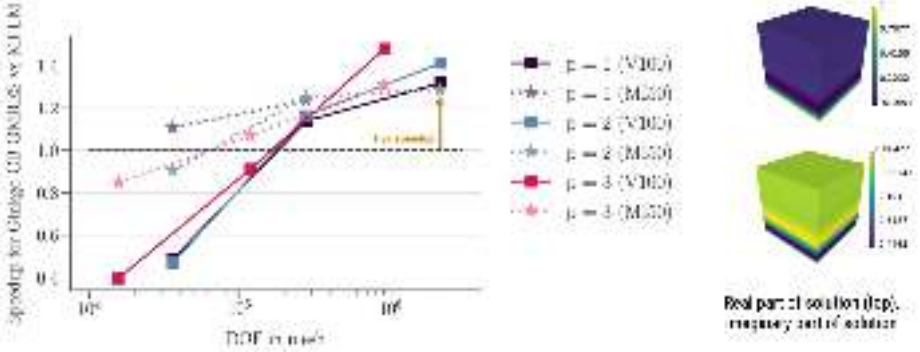
MFEM is a *free*, *lightweight*, *scalable* C++ library for finite element methods.

Speeding up MFEM's "example 22" on GPUs

Example 22 of the MFEM finite element library solves harmonic oscillation problems, with a forced oscillation imposed at the boundary. In this test, we use variant 1:

$$-\nabla \cdot (a\nabla u) - \omega^2 bu + i\omega cu = 0$$

with a = 1, b = 1, ω = 10, c = 20



Speedup of Ginkgo's Compressed Basis-GMRES solver vs MFEM's GMRES solver for three different orders of basis functions (p), using MFEM matrix-free operators and the Ginkgo-MFEM integration wrappers in MFEM. CUDA 10.1/V100 and ROcm 4.0/MI50.

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

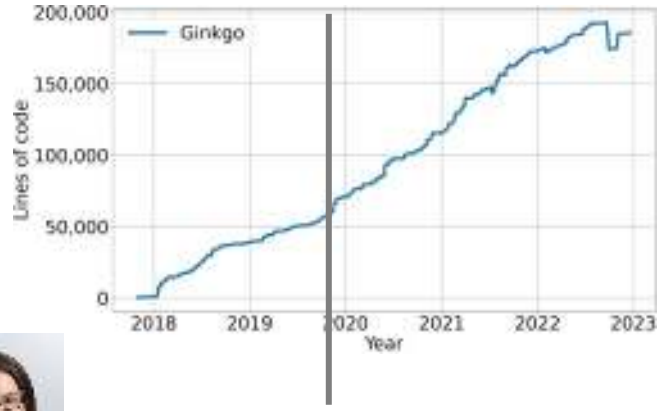
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP

NVIDIA AMD

Unit tests check correctness

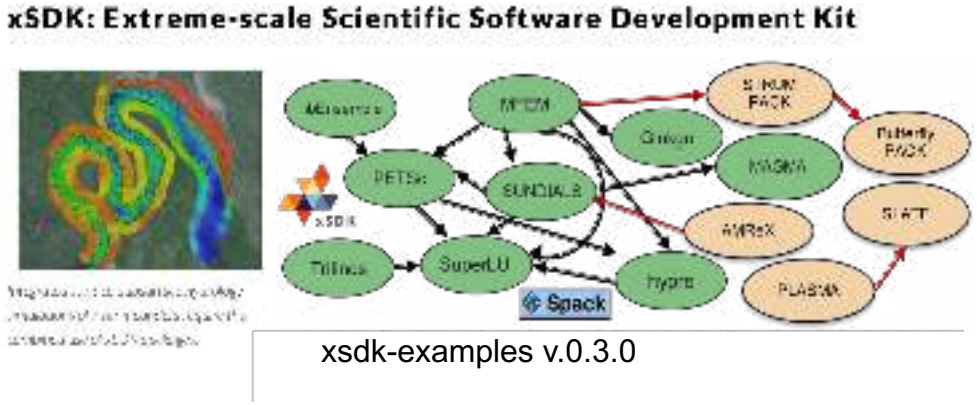
CI/CD



Spack xSDK E4S

	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓
	MC64/RCM reordering	✓		
	Wrapping user data		✓	

Part of the xSDK effort

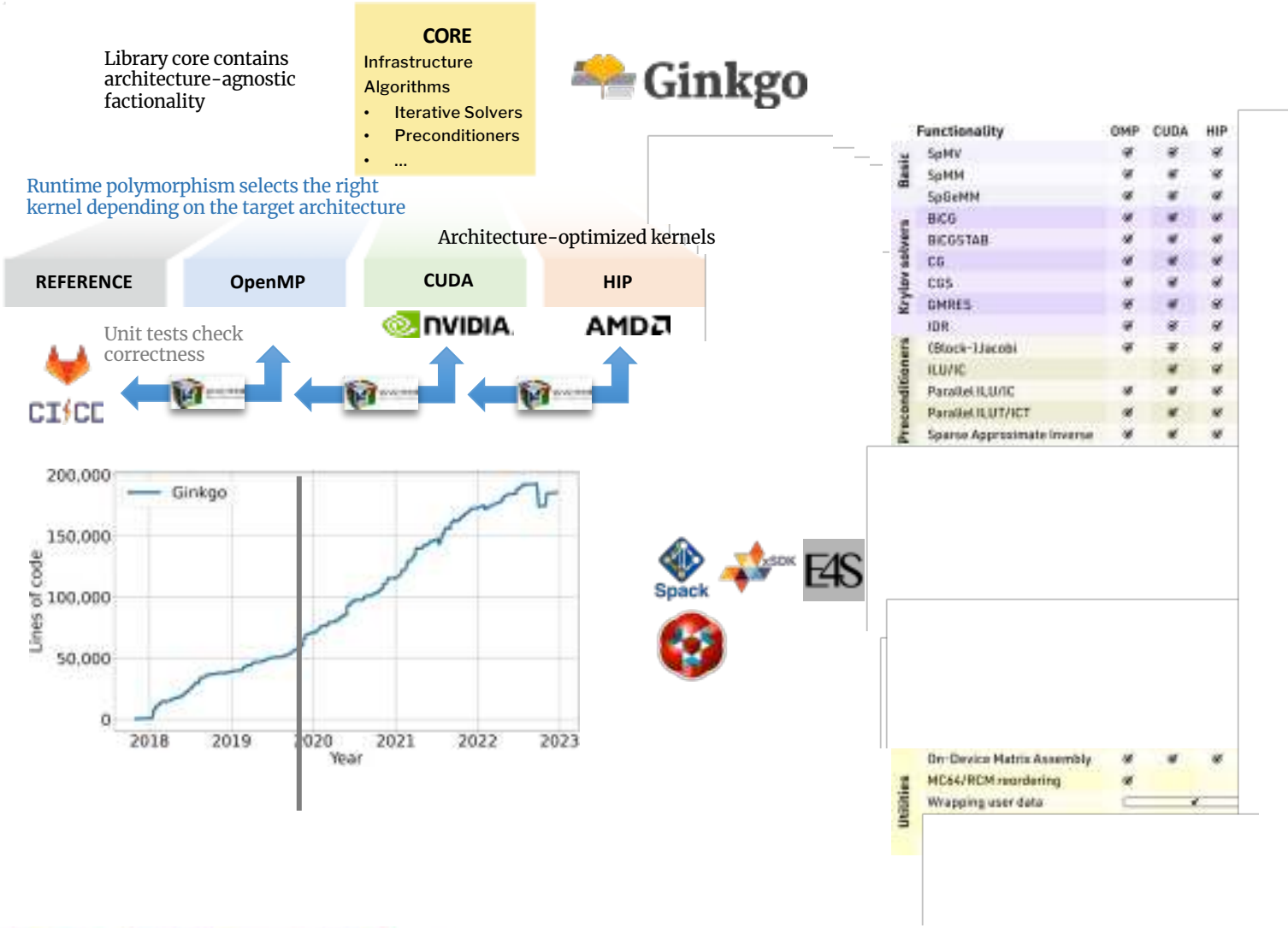


xsdk-examples v.0.3.0

The xSDK provides infrastructure for and interoperability of a **collection of related and complementary software elements**—developed by diverse, independent teams throughout the high-performance computing (HPC) community—that provide the building blocks, tools, models, processes, and related artifacts for rapid and efficient development of high-quality applications.

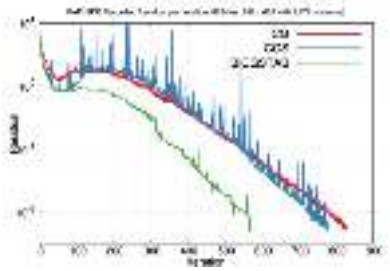
November 2022
- 26 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

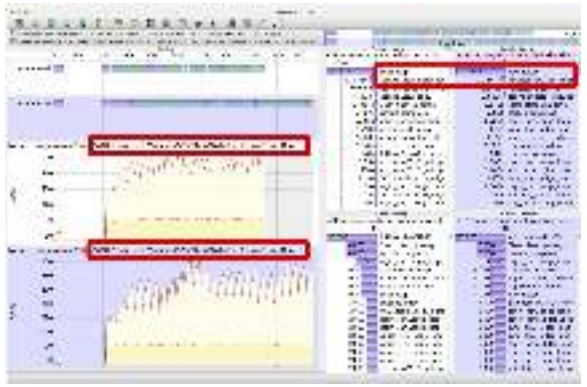
xSDK community policies:
- 16 mandatory policies,
- 8 recommended policies,
- 4 Spack variant guidelines
- Available on Github https://xsdk.info/policies/



Adding profiling functionality

Exa-PAPI





Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

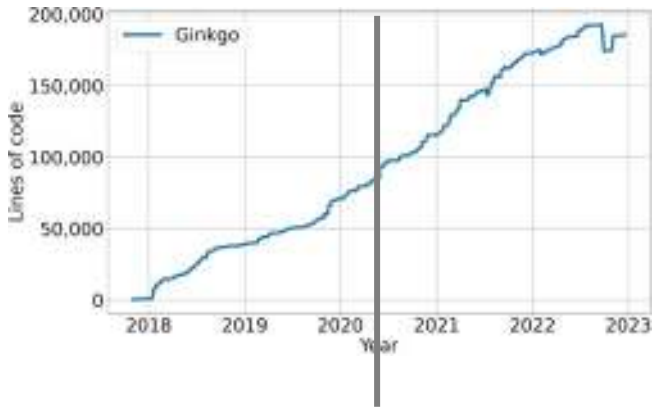
Architecture-optimized kernels

REFERENCE	OpenMP	CUDA	HIP

NVIDIA AMD

Unit tests check correctness

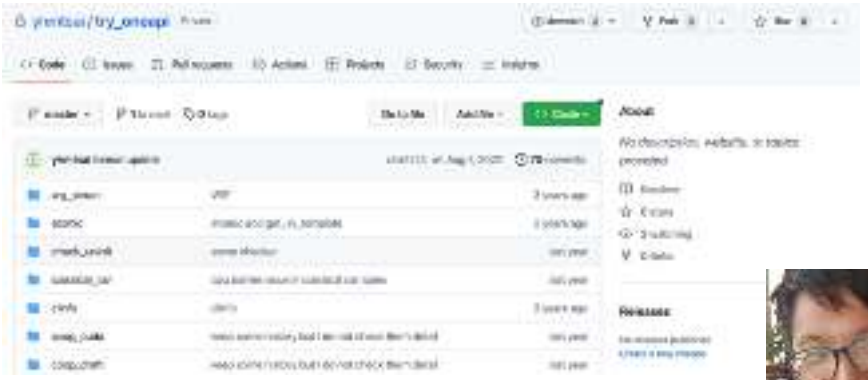
CI/CD



Spack xSDK E4S

Functionality	OMP	CUDA	HIP
Basic: SpMV	✓	✓	✓
Basic: SpMM	✓	✓	✓
Basic: SpGeMM	✓	✓	✓
Krylov solvers: BiCG	✓	✓	✓
Krylov solvers: BiCGSTAB	✓	✓	✓
Krylov solvers: CG	✓	✓	✓
Krylov solvers: CGS	✓	✓	✓
Krylov solvers: GMRES	✓	✓	✓
Krylov solvers: IDR	✓	✓	✓
Preconditioners: (Block-)Jacobi	✓	✓	✓
Preconditioners: ILU/IC		✓	✓
Preconditioners: Parallel ILU/IC	✓	✓	✓
Preconditioners: Parallel ILUT/ICT	✓	✓	✓
Preconditioners: Sparse Approximate Inverse	✓	✓	✓
Utilities: On-Device Matrix Assembly	✓	✓	✓
Utilities: MC64/RCM reordering	✓		
Utilities: Wrapping user data		✓	
Utilities: Logging		✓	
Utilities: PAPI counters		✓	

Extending to Intel GPUs



Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

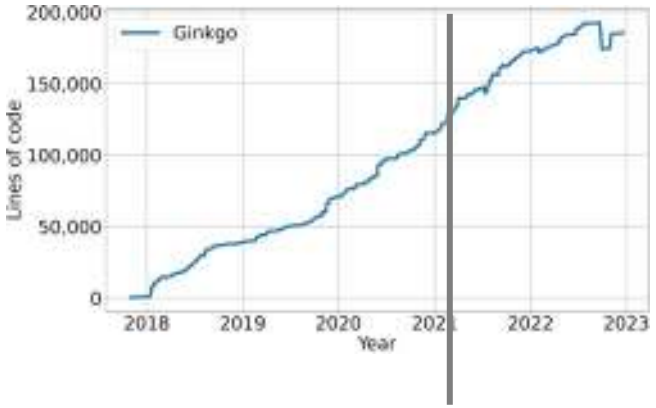
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

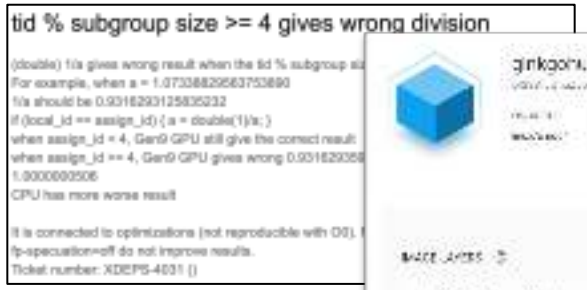
Industry Collaboration with bi-weekly meetings

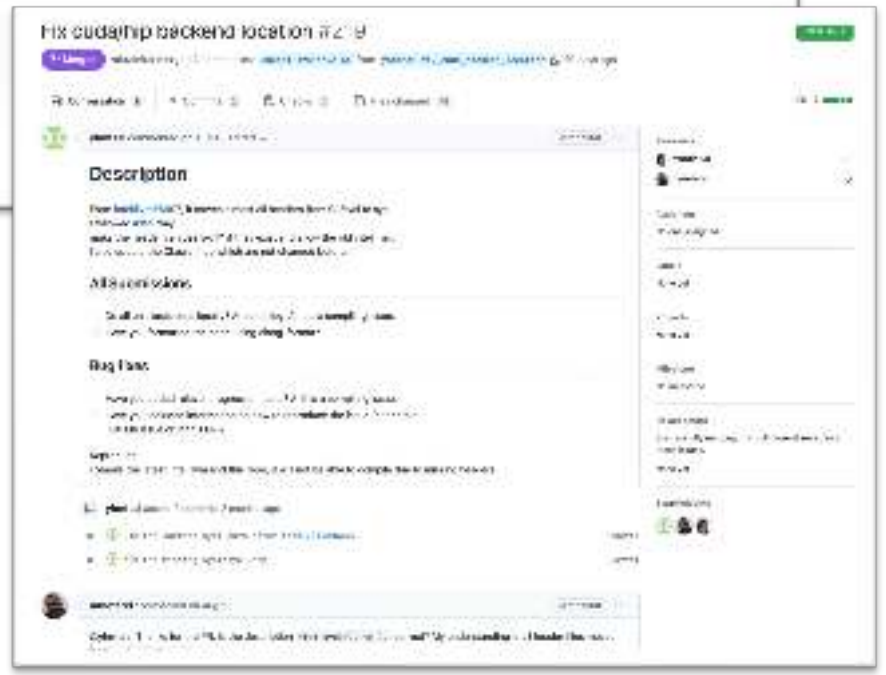
	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

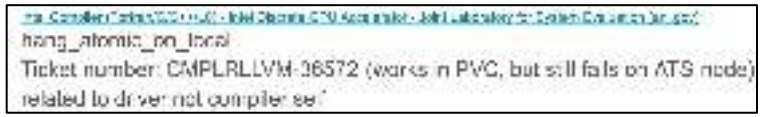
Extending to Intel GPUs

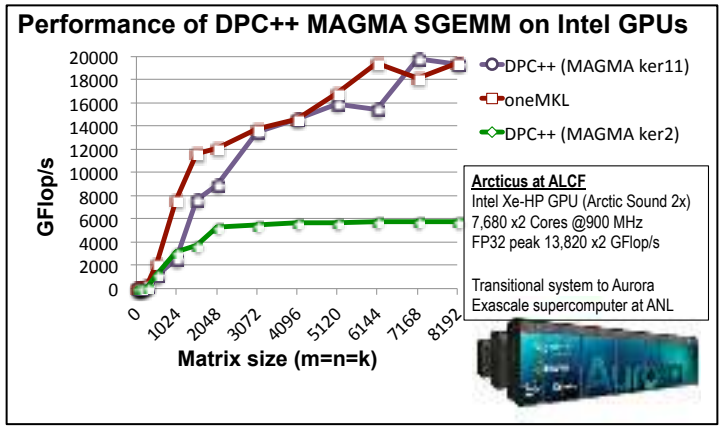
- Bi-Weekly technical meetings with Intel
- Long list of bug reports, feature requests, performance data discussions, documentation improvements …

… but also docker image contributions and bug fixes!

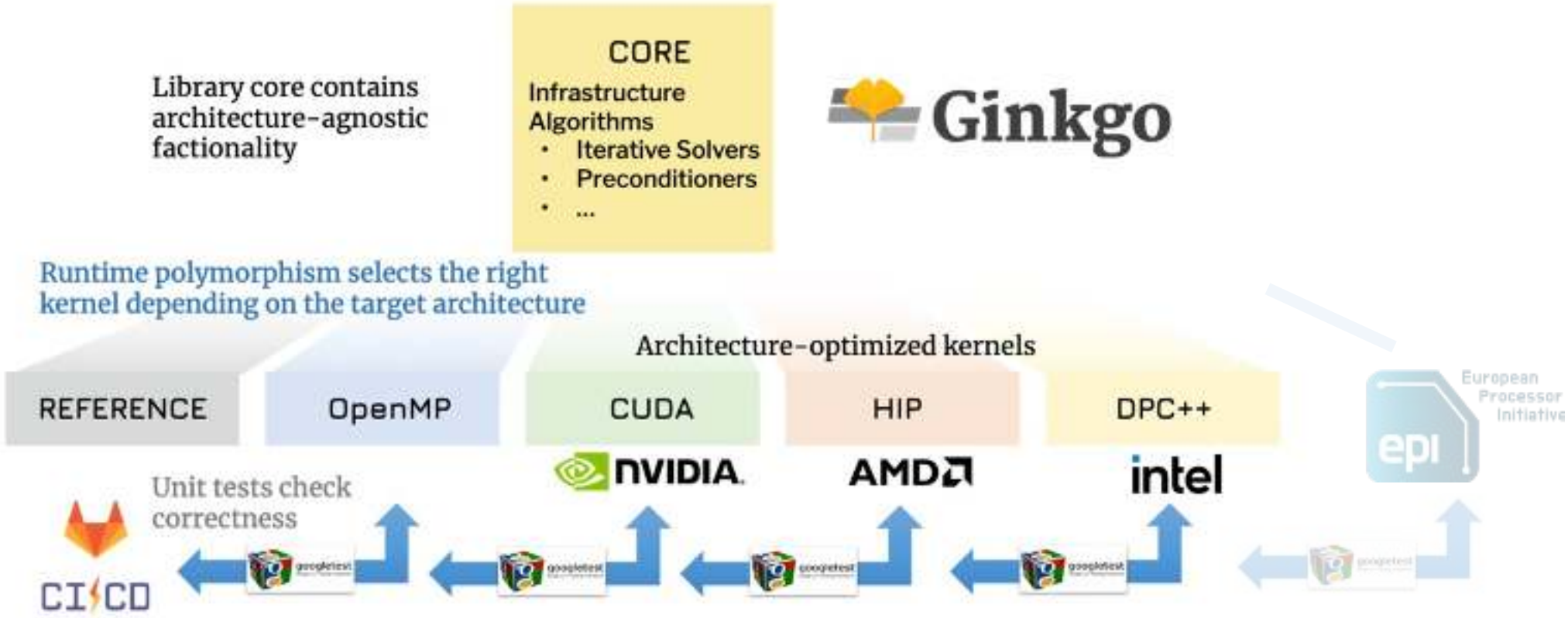








Portability as central design principle



Focus efforts as lightweight tool in ECP to address challenges

Focus efforts
- Mixed precision
- batched

- *Address recent hardware trends (tensor cores, etc.)*
- *Address hardware requirements*

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

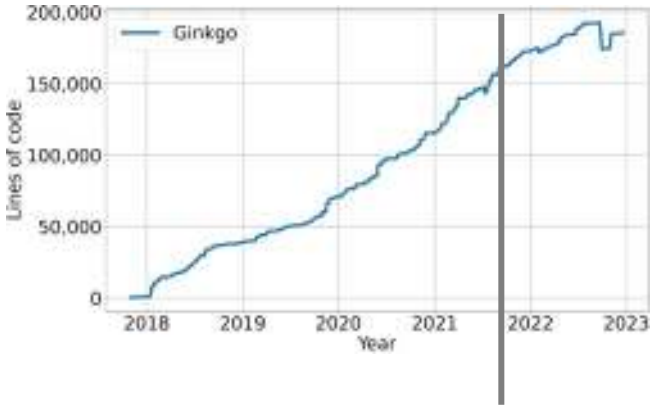
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

Spack xSDK E4S

Exa-PAPI

oneAPI

Industry Collaboration with bi-weekly meetings

Mixed precision focus effort

Focus efforts
- Mixed precision
- batched

- *Address recent hardware trends (tensor cores, etc.)*
- *Address hardware requirements*

Advances in Mixed Precision Algorithms: 2021 Edition

by the ECP Multiprecision Effort Team (Lead: Hartwig Anzt)

Ahmad Abdelfattah, Hartwig Anzt, Alan Ayala, Erik G. Boman, Erin Carson, Sebastien Cayrols, Terry Cojean, Jack Dongarra, Rob Falgout, Mark Gates, Thomas Grützmacher, Nicholas J. Higham, Scott B. Kruger, Sherry Li, Neil Lindquist, Yang Liu, Jennifer Loe, Piotr Luszczek, Pratik Nayak, Daniel Osei-Kuffuor, Sri Pranesh, Sivasankaran Rajamanickam, Tobias Ribizel, Barry Smith, Kasia Swirydowicz, Stephen Thomas, Stanimire Tomov, Yaohung M. Tsai, Ichi Yamazaki, Urike Meier Yang

Available access Research article First published online March 18, 2021

A survey of numerical linear algebra methods utilizing mixed-precision arithmetic

Ahmad Abdelfattah, Hartwig Anzt, [redacted] and Ulrike Meier Yang View all authors and affiliations

Volume 35, Issue 4 https://doi.org/10.1177/10943420211003313

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

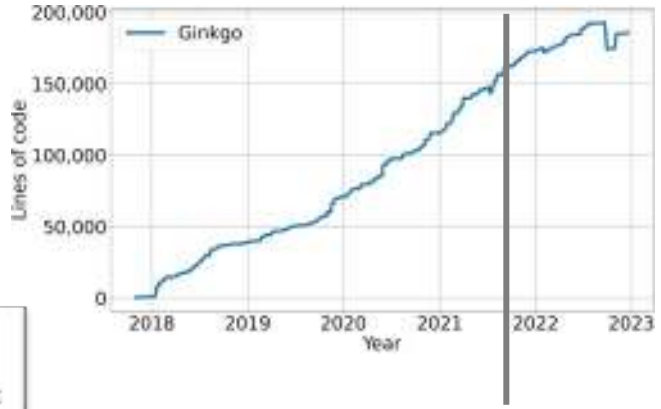
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

Industry Collaboration with bi-weekly meetings

	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓

Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

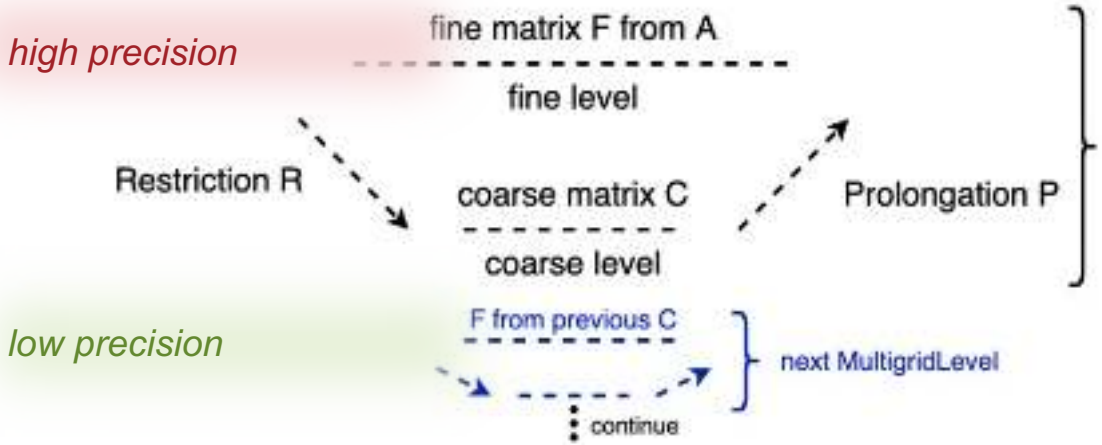
ECP EXASCALE COMPUTING PROJECT

Mixed precision AMG on GPUs

- **Preconditioning iterative solvers**
  - Idea: Approximate inverse of system matrix to make the system “easier to solve”: $P^{-1} \approx A^{-1}$

    and solve $Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b}$

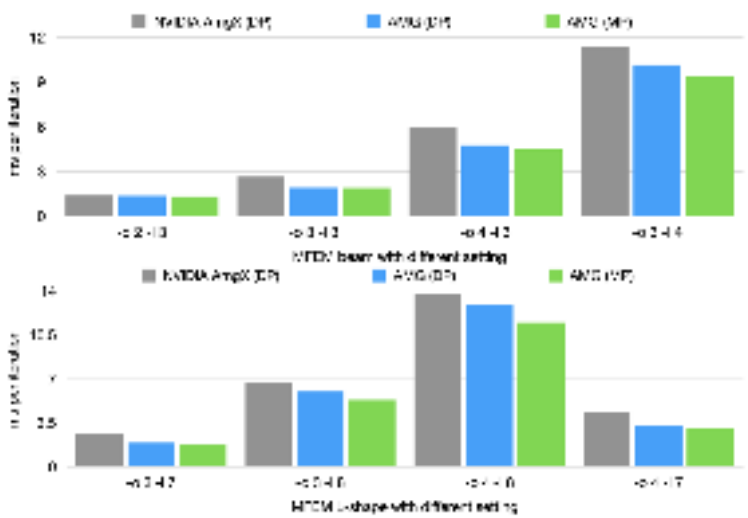
- **Mixed Precision Multigrid Preconditioner**




```
multigrid::build()
    .with_max_levels(10u) // equal to NVIDIA/AMGX 11 max levels
    .with_min_coarse_row(64u)
    .with_pre_smoother(sm, sm_f)
    .with_mg_level(pgm, pgm_f)
    .with_level_selector(
        [](const size_type level, const LinOp*) -> size_type {
            // Only the first level is generated by MultigridLevel(double).
            // The subsequent levels are generated by MultigridLevel(float)
            return level >= 1 ? 1 : 0;
        })
    .with_coarest_solver(coarest_solver_f)
```

Ginkgo

Mike Tsai



Stephen F. McCormick, Joseph Benzaken, Rasmus Tamstorf: Algebraic error analysis for mixed-precision multigrid solvers, https://arxiv.org/abs/2007.06614

Mixed precision AMG on GPUs

Focus efforts
- Mixed precision
- batched


Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

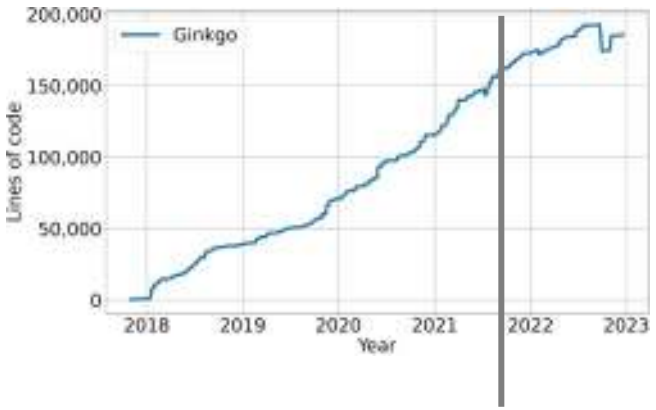
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

Industry Collaboration with bi-weekly meetings

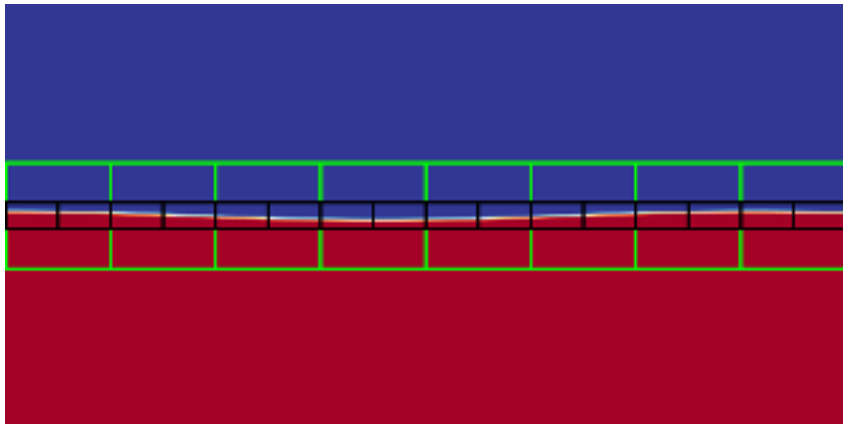
	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
AMG	AMG preconditioner	✓	✓	✓	✓
	AMG solver	✓	✓	✓	✓
	Parallel Graph Match	✓	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

Batched focus effort – Combustion Simulations

Batched iterative solvers for SUNDIALS / PeleLM

PeleLM is a parallel, adaptive mesh refinement (AMR) code that solves the reacting Navier-Stokes equations in the low Mach number regime. The core libraries for managing the subcycling AMR grids and communication are found in the AMReX source code.

https://amrex-combustion.github.io/PeleLM/overview.html



Problem	Size	Non-zeros (A)	Non-zeros (L+U)
[illegible]	54	2,332 (80%)	2,734 (94%)
[illegible]	22	438 (90%)	442 (91%)
[illegible]	33	978 (90%)	1,018 (93%)
[illegible]	54	2,560 (88%)	2,860 (98%)
[illegible]	144	6,135 (30%)	20,307 (98%)
[illegible]	10	91 (91%)	91 (91%)

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

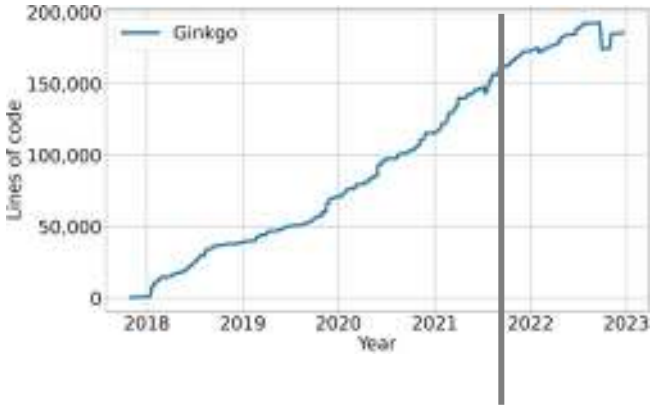
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

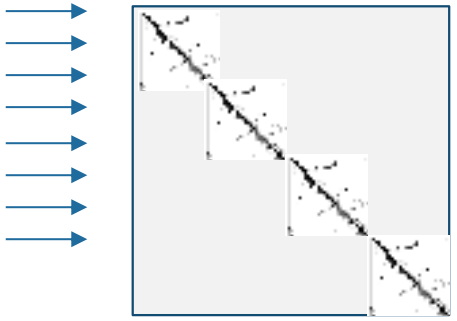
Industry Collaboration with bi-weekly meetings

	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
AMG	AMG preconditioner	✓	✓	✓	✓
	AMG solver	✓	✓	✓	✓
	Parallel Graph Match	✓	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

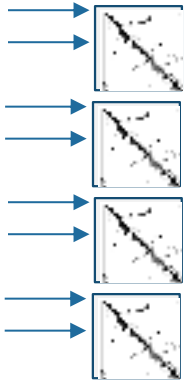
Batched focus effort – Combustion Simulations

- Many sparse problems of medium size have to be solved concurrently.
  - ~ 50 – 2,000 unknowns, < 50% dense;
  - All sparse systems may share the same sparsity pattern;
  - An approximate solution may be acceptable (e.g., inside a non-linear solver);

- One solution is to arrange the individual systems on the main diagonal of one large system.
  - Convergence determined by the “hardest” problem;
  - No reuse of sparsity pattern information;
  - Global synchronization points;



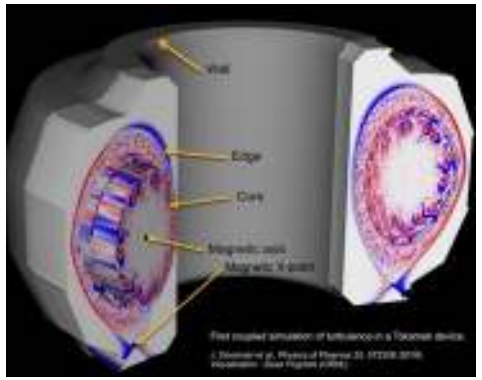
- Better approach: design batched iterative solve functionality that solves all problems concurrently.
  - Problem-dependent convergence accounted for;
  - No global synchronization;
  - Reuse of sparsity pattern information;

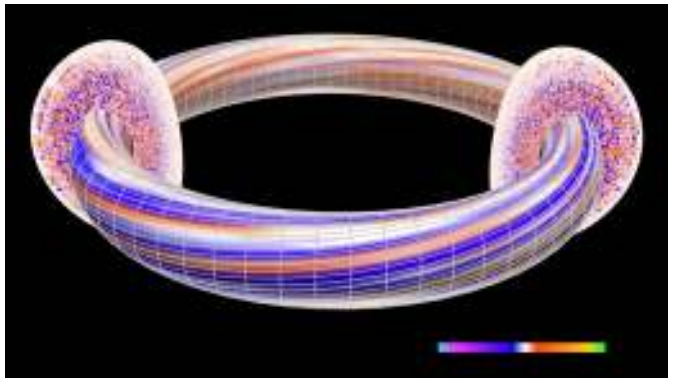


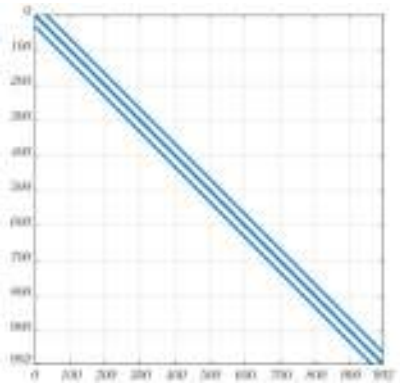
Batched focus effort – Fusion Plasma Simulations

XGC is a gyrokinetic particle-in-cell code, which specializes in the simulation of the edge region of magnetically confined thermonuclear fusion plasma. The simulation domain can include the magnetic separatrix, magnetic axis and the biased material wall. XGC can run in total-delta-f, and conventional delta-f mode. The ion species are always gyrokinetic except for ETG simulation. Electrons can be adiabatic, massless fluid, driftkinetic, or gyrokinetic.

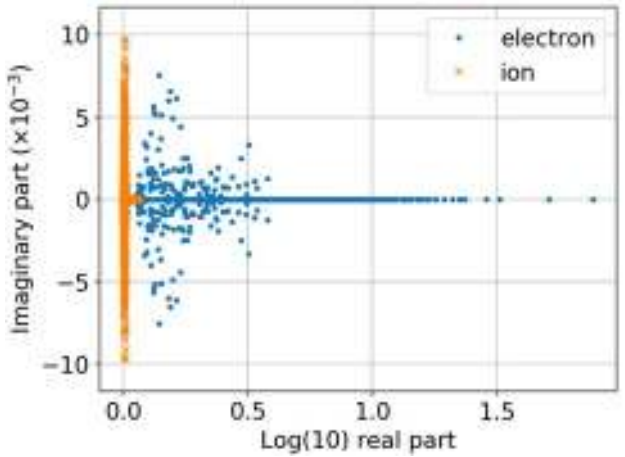
Source: https://xgc.pppl.gov/html/general_info.html





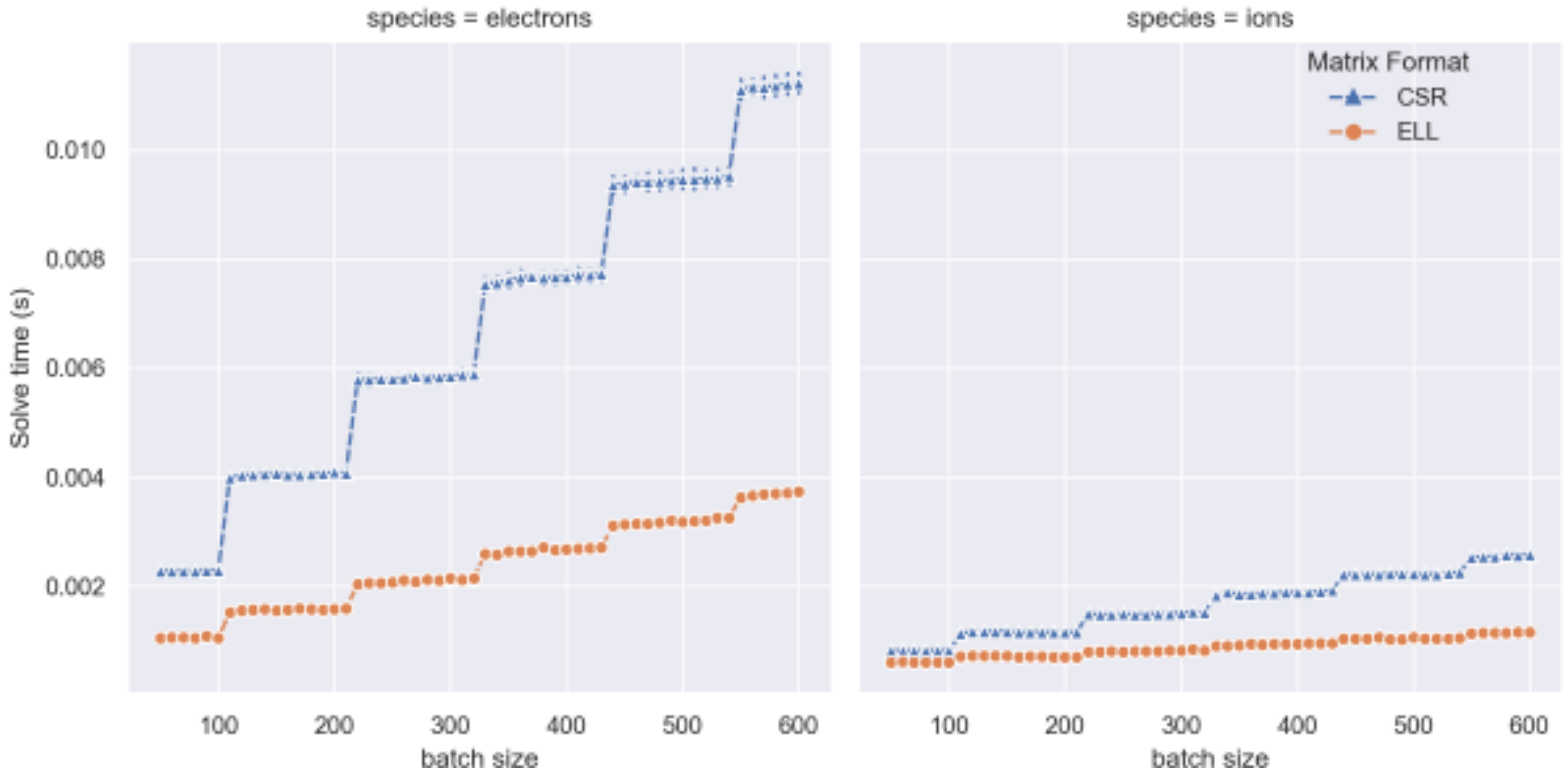


- Two species
- Ions easy to solve
- Electrons hard to solve
- Banded matrix structure
- Non-symmetric, need BiCGSTAB
- n = ~1,000
- nz = ~9,000



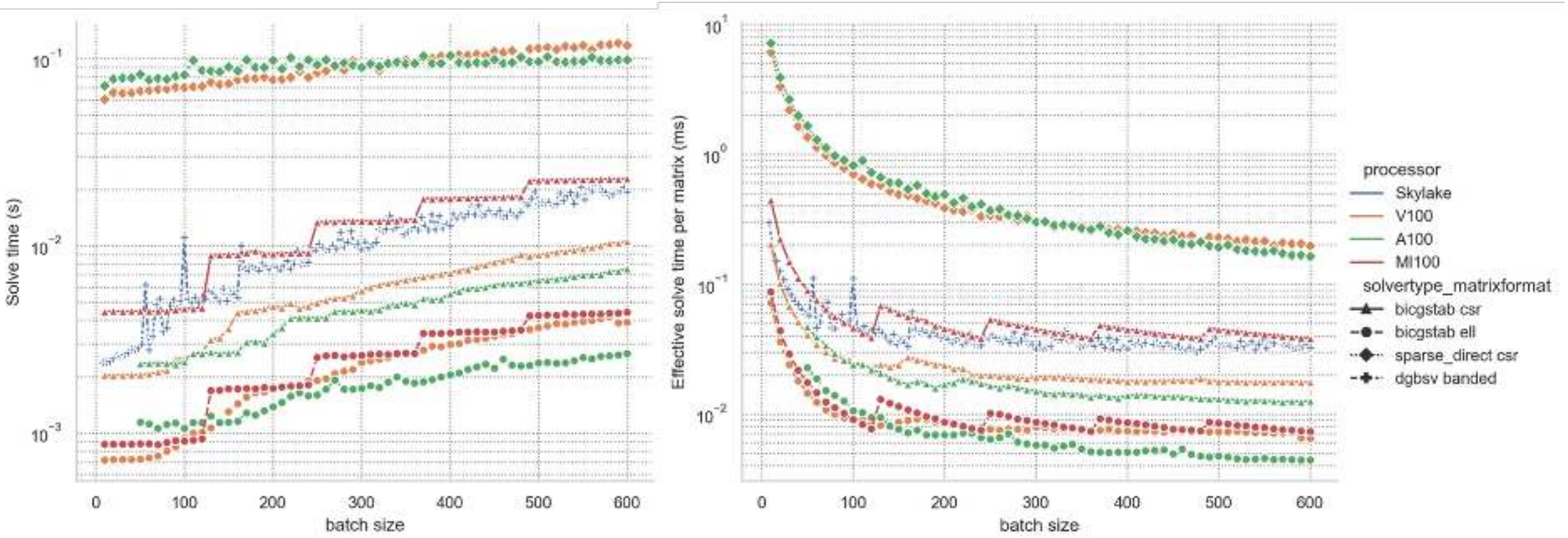
Batched focus effort – Fusion Plasma Simulations

NVIDIA A100 GPU



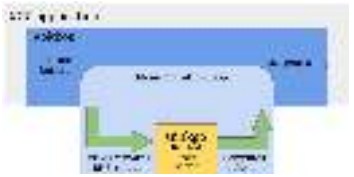
- Ions easy to solve
- Electrons hard to solve
- ELL format more suitable
- “step pattern” for runtime filling up multiprocessors

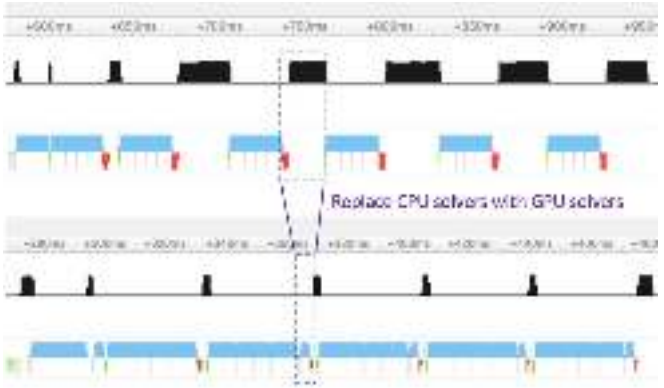
Batched focus effort – Fusion Plasma Simulations



Aditya Kashi, Pratik Nayak, Dhruva Kulkarni, Aaron Scheinberg, Paul Lin, and Hartwig Anzt. **Batched sparse iterative solvers on gpu for the collision operator for fusion plasma simulations**. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 157–167. IEEE, 2022.

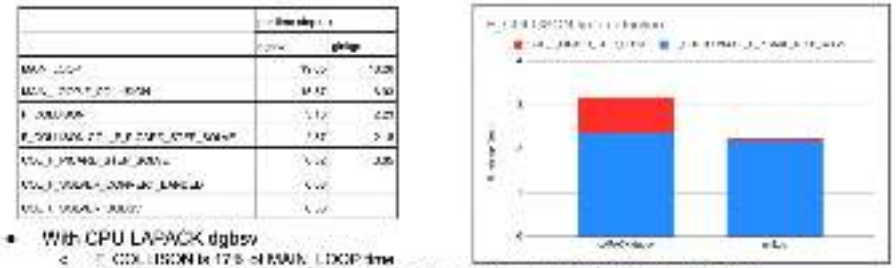
Batched focus effort – Fusion Plasma Simulations





XGC collision operator solve LAPACK vs. Ginkgo: XGC pe459_d3d_EM_heatload test case

- XGC pe459_d3d_EM_heatload (Aaron's test case; used for Summit, Perlmutter and Crusher scaling studies)
- Preliminary study on 32 nodes of Perlmutter (128 A100s)
  - 2 poloidal planes (216k nodes per plane); 22.4M pl/GPU, 89.6M pl/node (pl_num=700k)
  - Ran 20 time steps, collisions calculated every other time step



- With CPU LAPACK dgbsv
  - COL_F_COLLISION is 17% of MAIN_LOOP time
  - COL_F_PICARD_STEP_SOLVE is 24% of F_COLLISION time and 4.0% of MAIN_LOOP time
  - COL_F_SOLVER_DGBSV is 60% of COL_F_PICARD_STEP_SOLVE
  - COL_F_SOLVER_CONVERT_BANDED is 10% of COL_F_PICARD_STEP_SOLVE
- Replacing CPU LAPACK cgbsv by GPU Ginkgo
  - COL_F_PICARD_STEP_SOLVE reduced from 0.82s to 0.194s per step, reduction of 94%
  - F_COLLISION reduced from 3.18s to 2.21s per step, reduction of 31%
  - MAIN_LOOP time reduced by 4.1%

Velocity grid: 33x33; mesh size: 1287 nodes

XGC collision operator solve performed on GPU

© Doug Kothe

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

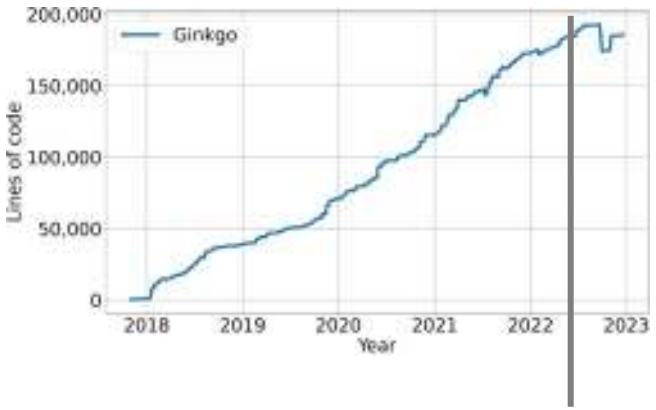
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

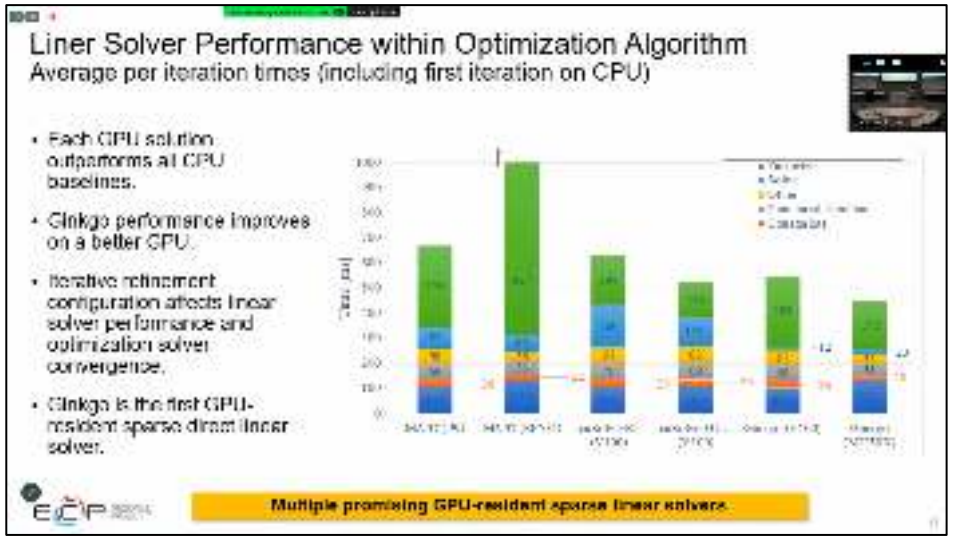
oneAPI

Industry Collaboration with bi-weekly meetings

XGC

	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
Batched	Batched BiCGSTAB	✓	✓	✓	
	Batched CG	✓	✓	✓	
	Batched GMRES	✓	✓	✓	
	Batched ILU	✓	✓	✓	
	Batched ISAI	✓	✓	✓	
	Batched Jacobi	✓	✓	✓	
AMG	AMG preconditioner	✓	✓	✓	✓
	AMG solver	✓	✓	✓	✓
	Parallel Graph Match	✓	✓	✓	✓
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

Sparse direct solvers for power grid simulations



EXASGD © Slaven Peles

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

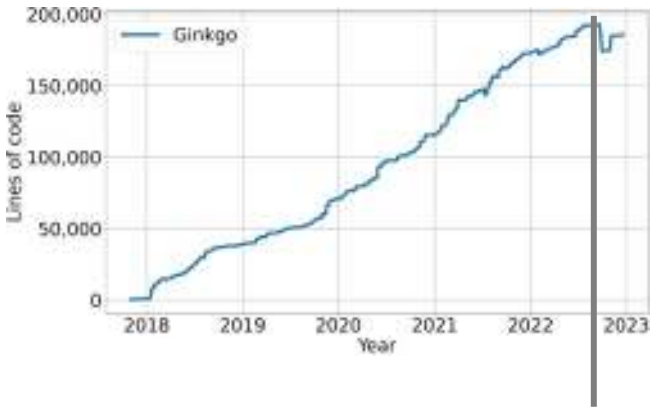
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

Industry Collaboration with bi-weekly meetings

XGC

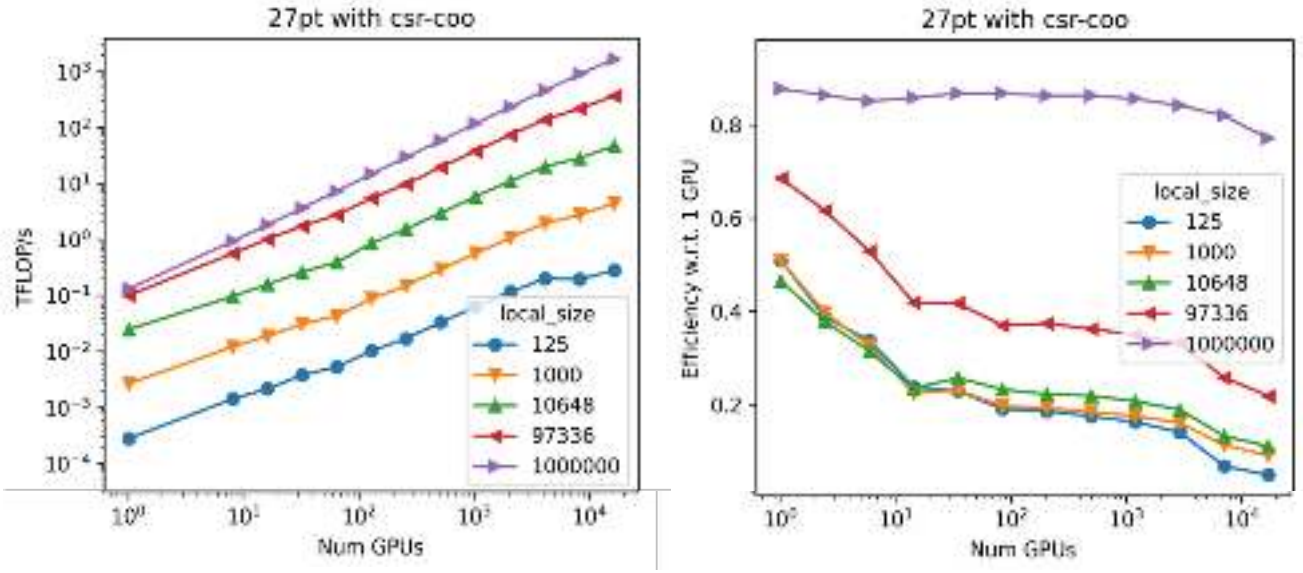
EXASGD

	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
Batched	Batched BICGSTAB	✓	✓	✓	
	Batched CG	✓	✓	✓	
	Batched GMRES	✓	✓	✓	
	Batched ILU	✓	✓	✓	
	Batched ISAI	✓	✓	✓	
	Batched Jacobi	✓	✓	✓	
AMG	AMG preconditioner	✓	✓	✓	✓
	AMG solver	✓	✓	✓	✓
	Parallel Graph Match	✓	✓	✓	✓
Sparse direct	Symbolic Cholesky	✓	✓	✓	✓
	Numeric Cholesky	UNDER DEVELOPMENT			
	Symbolic LU	✓	✓	✓	✓
	Numeric LU	✓	✓	✓	
	Sparse TRSV	✓	✓	✓	
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

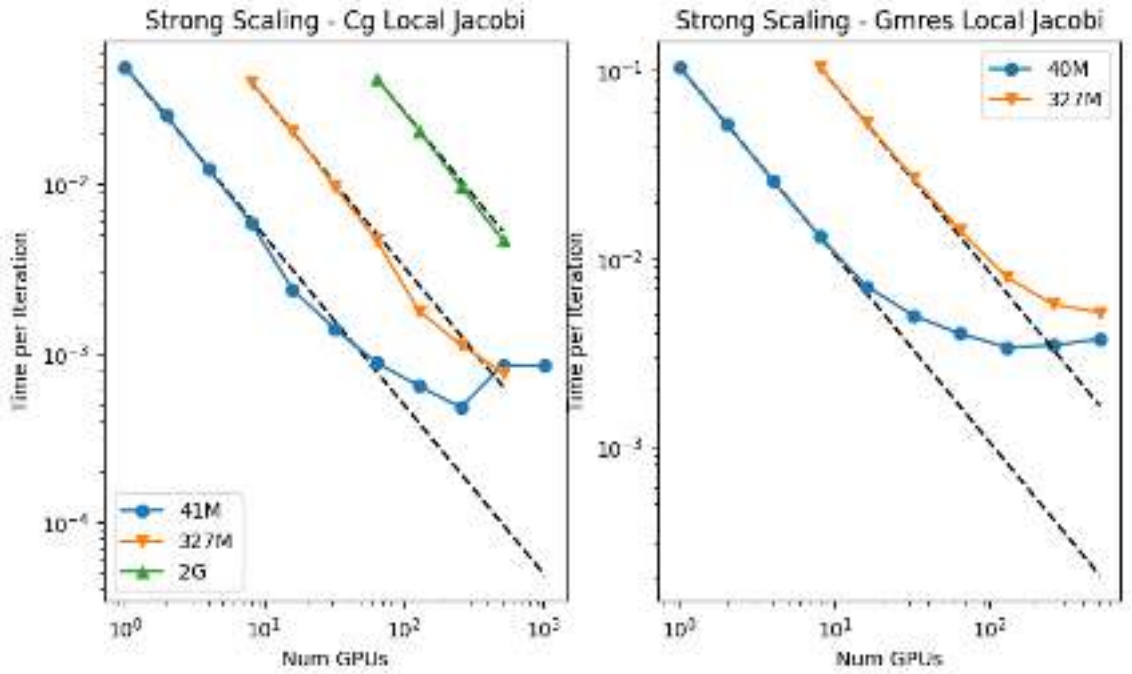
Distributed runs on Frontier (Cray + AMD MI250 GPUs)

Weak scaling: problem size increases with parallel resources

Weak scaling up to 16k GCDs (8k GPUs)

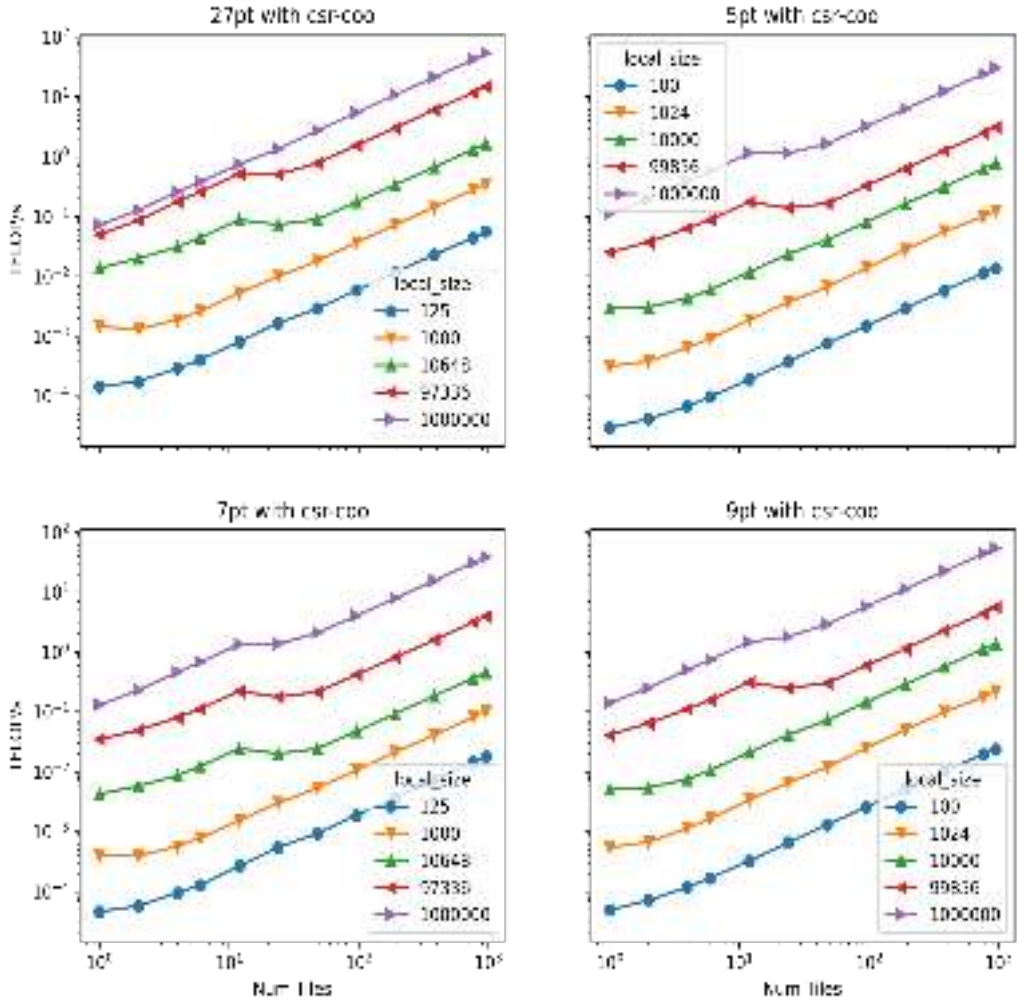


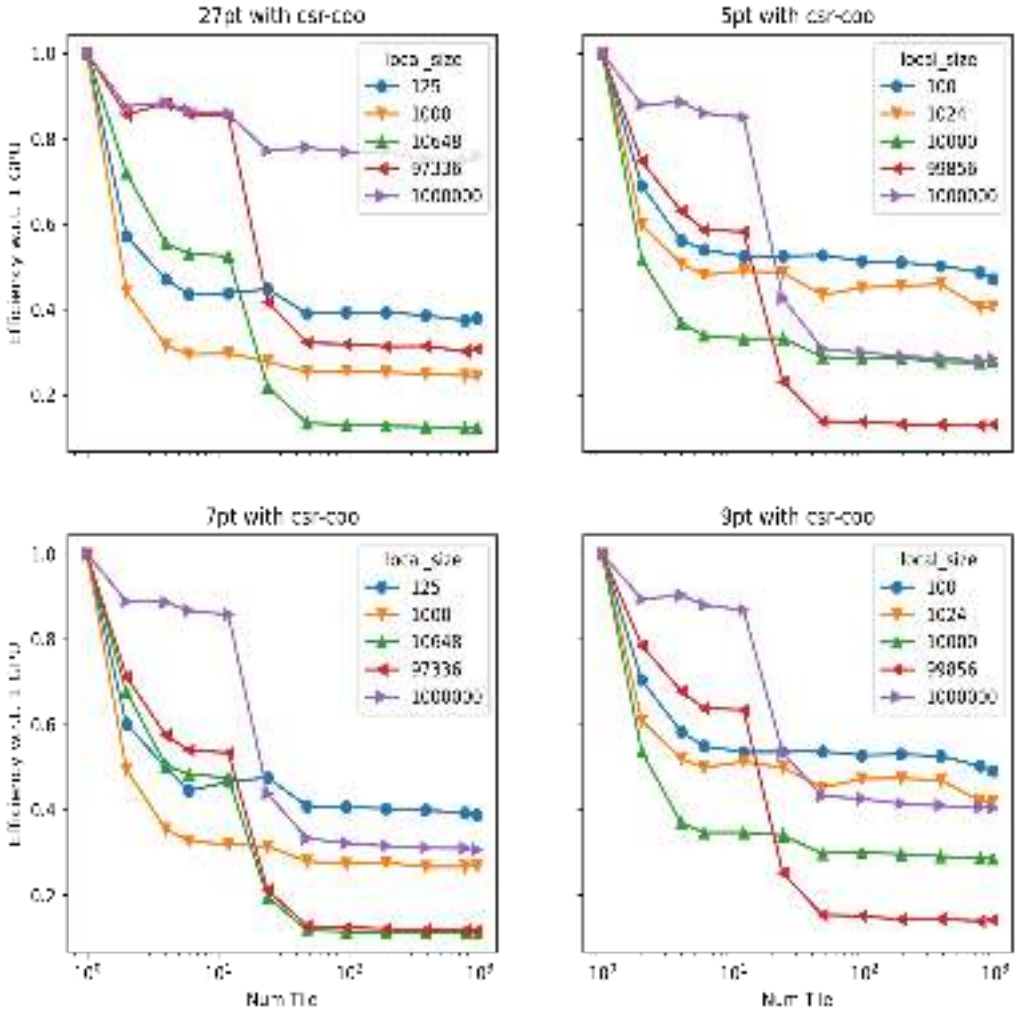
Strong scaling: problem size constant



Distributed runs on Sunspot (Intel PVCA GPUs)

Weak scaling: problem size increases with parallel resources

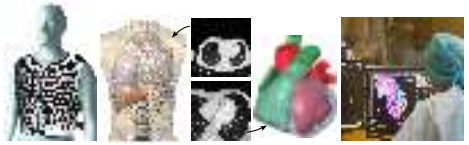




“Now” – Near completion of ECP

- Sustainable software design ready for the addition of new backends.

- EuroHPC Project MICROCARD uses Ginkgo



https://www.microcard.eu

- BMBF PDExa project uses Ginkgo

- BMBF ExaSIM project uses Ginkgo

https://exasim-project.com

Library core contains architecture-agnostic factionality

CORE
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

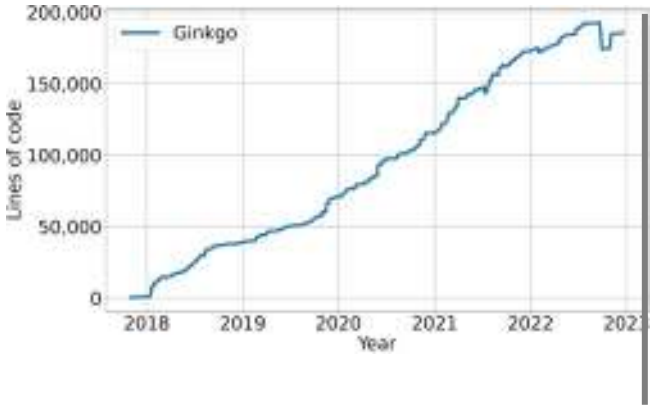
Architecture-optimized kernels

REFERENCE | OpenMP | CUDA | HIP | DPC++

NVIDIA AMD intel

Unit tests check correctness

CI/CD



Spack xSDK E4S

Exa-PAPI

oneAPI

Industry Collaboration with bi-weekly meetings

XGC

EXASGD

	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓	✓
Batched	Batched BiCGSTAB	✓	✓	✓	
	Batched CG	✓	✓	✓	
	Batched GMRES	✓	✓	✓	
	Batched ILU	✓	✓	✓	
	Batched ISAI	✓	✓	✓	
	Batched Jacobi	✓	✓	✓	
AMG	AMG preconditioner	✓	✓	✓	✓
	AMG solver	✓	✓	✓	✓
	Parallel Graph Match	✓	✓	✓	✓
Sparse direct	Symbolic Cholesky	✓	✓	✓	✓
	Numeric Cholesky	UNDER DEVELOPMENT			
	Symbolic LU	✓	✓	✓	✓
	Numeric LU	✓	✓	✓	
	Sparse TRSV	✓	✓	✓	
Utilities	On-Device Matrix Assembly	✓	✓	✓	✓
	MC64/RCM reordering	✓			
	Wrapping user data		✓		
	Logging		✓		
	PAPI counters		✓		

Lessons learnt from the Ginkgo development process

- **ECP earmarking roughly half the budget to Software & App development is a game changer.**
  - **Central component for the success of ECP.**
  - This concept needs to – and does become - the blueprint for other nations and projects.

- **Workforce recruitment and workforce retention are the key to success in software development.**
  - Money does not write software. RSEs do. **We need to create attractive career plans.**
  - We need to make research software development attractive to students. **Academic recognition.**

- **Anticipating the future in hardware development accelerates the porting process.**
  - **Blueprints** and **early access systems** both useful.
  - **Interaction with industry** is mutually beneficial.

- **Management, tools, and strategic initiatives, interaction and collegial behavior are important.**
  - Jira/Notion/[…] milestones and deliverables give projects and collaborative interactions a structure and timeline.
  - **Strategic focus groups, conferences,** and **meetings** bring experts together and **create collaboration**.
  - **Listen to the application needs. Value input and acknowledge collaborators.**