

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

Approved for public release



Hartwig Anzt, University of Tennessee



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)*

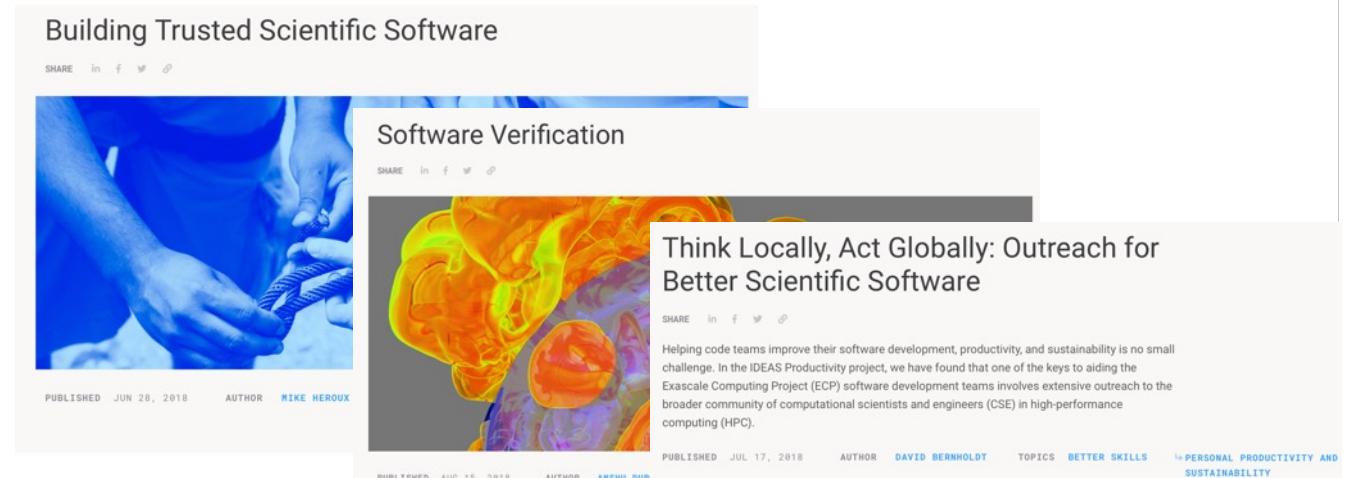
Designing an ECP library for sustaining simulation performance

MAGMA SPARSE

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

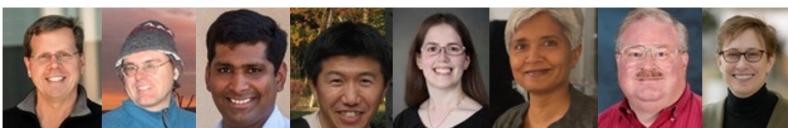
Design considerations for Ginkgo

- 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



The image shows three blog post cards from the IDEAS productivity website:

- Building Trusted Scientific Software** (Published Jun 28, 2018, Author: MIKE HEROUX) - An image of hands tying a knot in a blue rope. Text: "I have worked in the scientific software field for more than one decade. A phrase I often hear is ‘Verification is doing things right, and validation is doing the right thing.’ This phrase is often repeated without much thought given to what it means. In this post, I will try to explain what I mean by this phrase and why it is important to remember it when developing scientific software."
- Software Verification** (Published Aug 15, 2018, Author: ANSHU DUB) - An image of a colorful 3D visualization of a complex system. Text: "In the realm of software, verification is often erroneously equated with validation. Verification is a proper subset of validation for gaining confidence in the correctness of the software. It is the process by which the developers convince themselves that the software does what it was designed to do. In scientific software this could mean numerical stability, and efficacy of the method in the regard of the expected results. Note that verification is limited to ensuring that the model specification matches the reality. The latter is normally a part of the validation process."
- Think Locally, Act Globally: Outreach for Better Scientific Software** (Published Jul 17, 2018, Author: DAVID BERNHOLDT) - An image of a colorful 3D visualization of a complex system. Text: "Helping code teams improve their software development, productivity, and sustainability is no small challenge. In the IDEAS Productivity project, we have found that one of the keys to aiding the Exascale Computing Project (ECP) software development teams involves extensive outreach to the broader community of computational scientists and engineers (CSE) in high-performance computing (HPC)."



Designing an ECP library for sustaining simulation performance

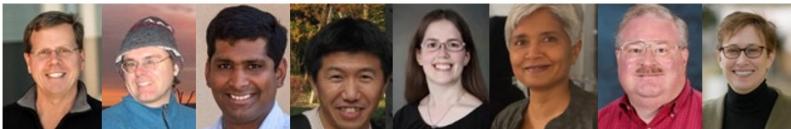
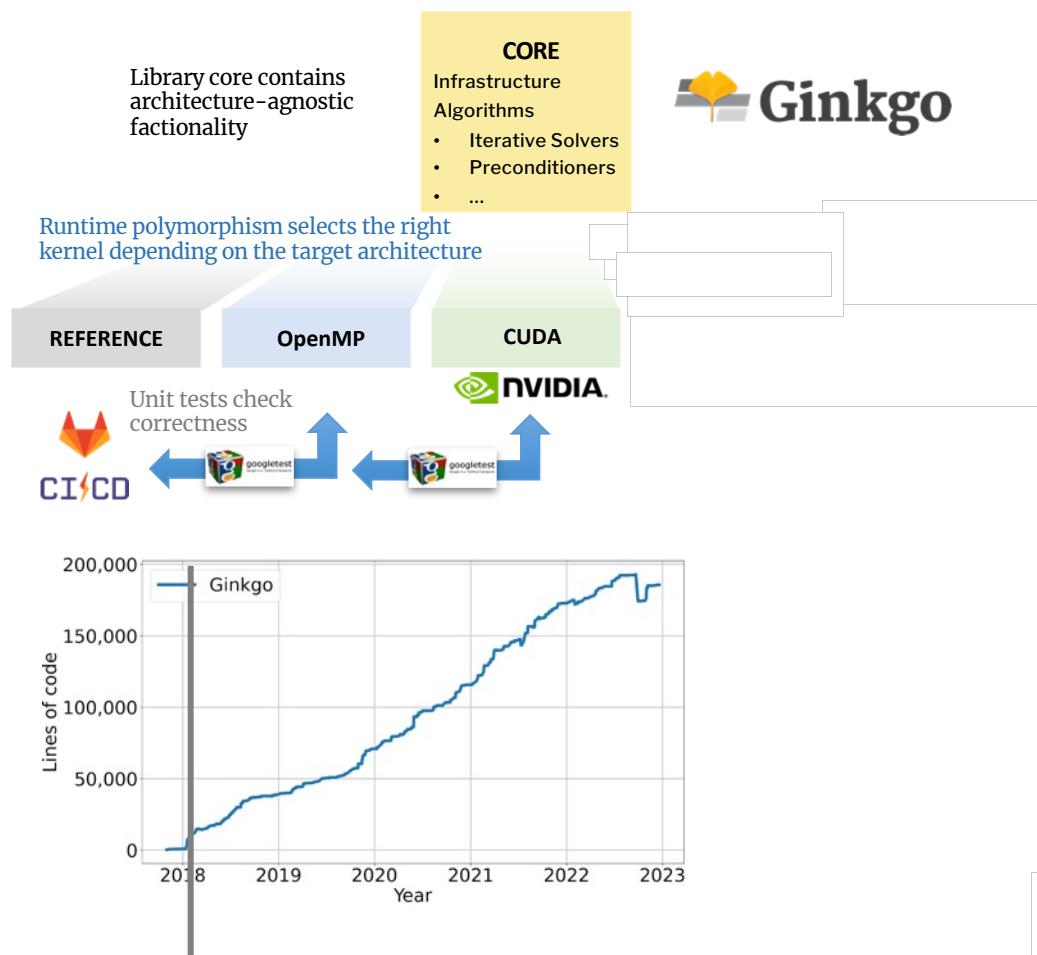
MAGMA SPARSE

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

$$x := A \cdot b$$

Preconditioner (for matrix A)

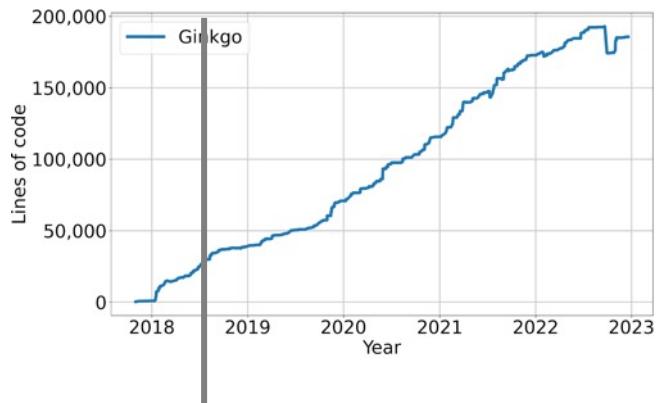
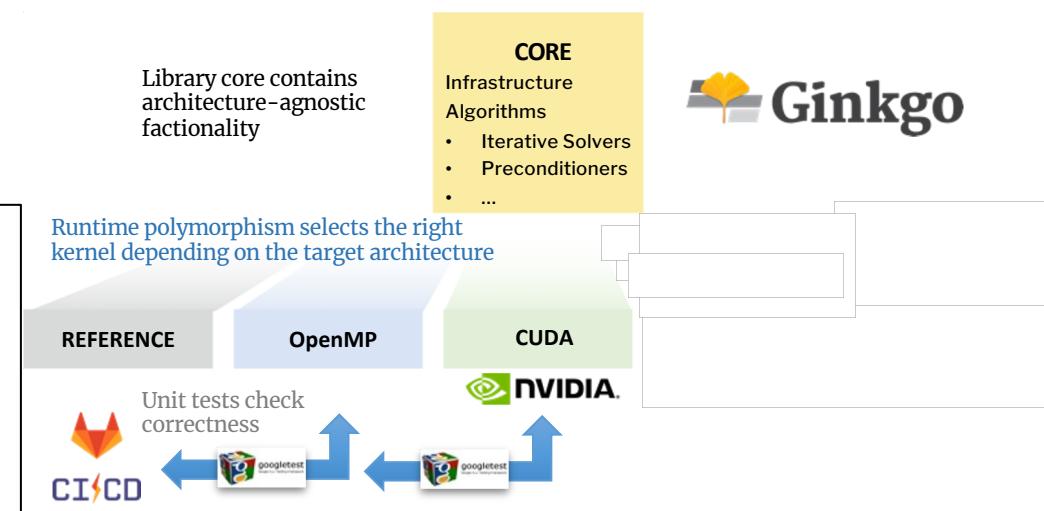
$$\begin{aligned} x &:= M^{-1} \cdot b \\ M^{-1} &\approx A^{-1} \end{aligned}$$

Solver (for system $Ax = b$)

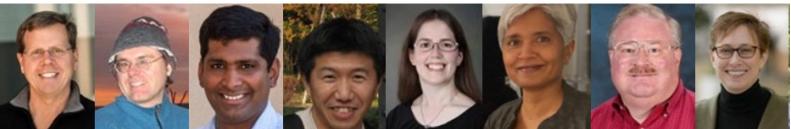
$$\begin{aligned} x &:= S \cdot b \\ S &\approx A^{-1} \\ S &= \Sigma(A) \end{aligned}$$

All of them can be expressed as

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

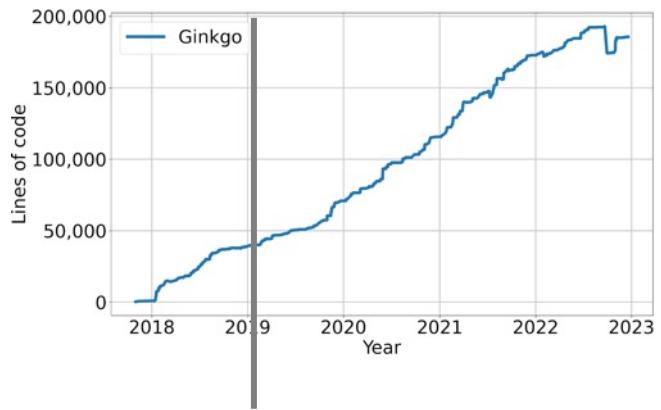
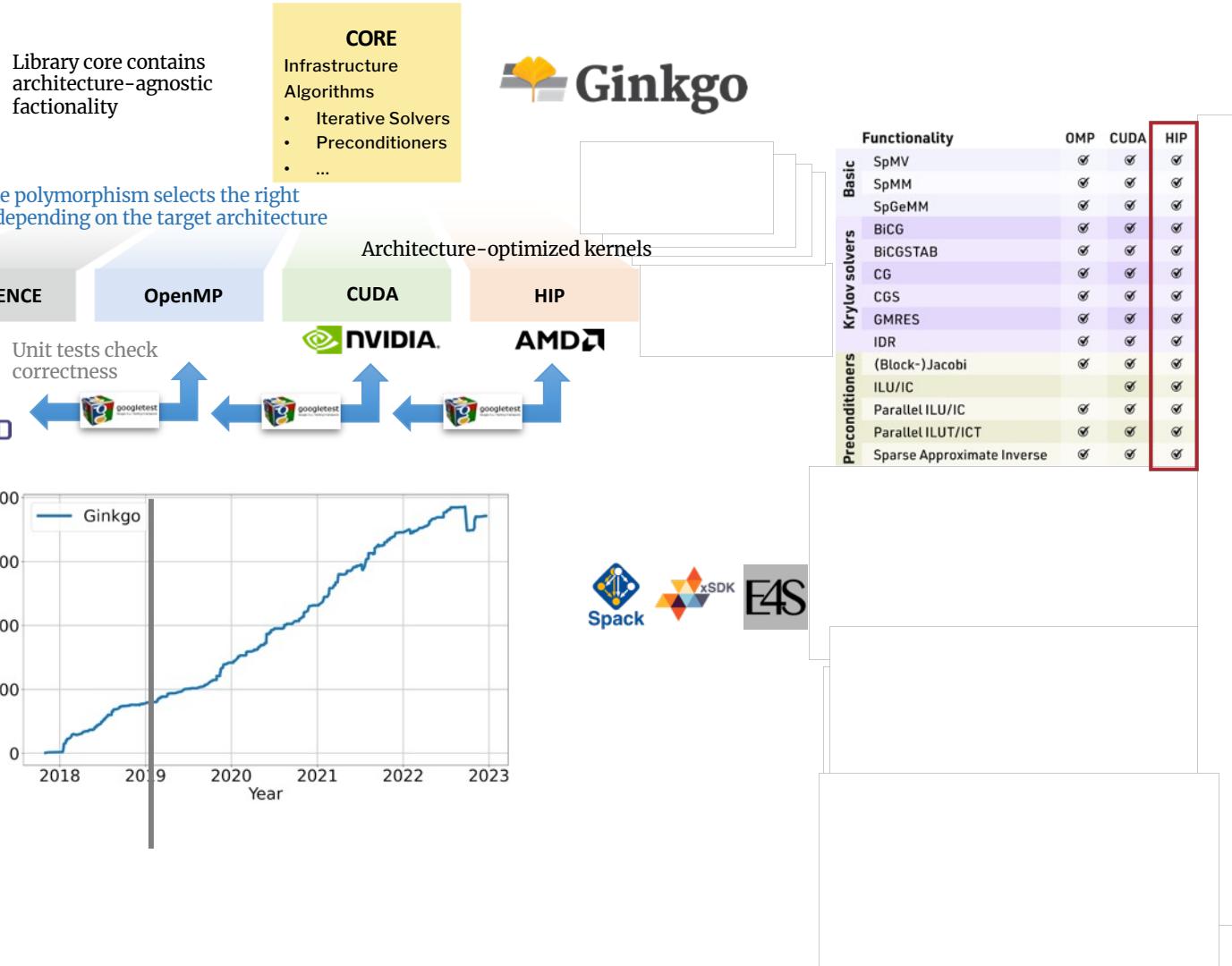
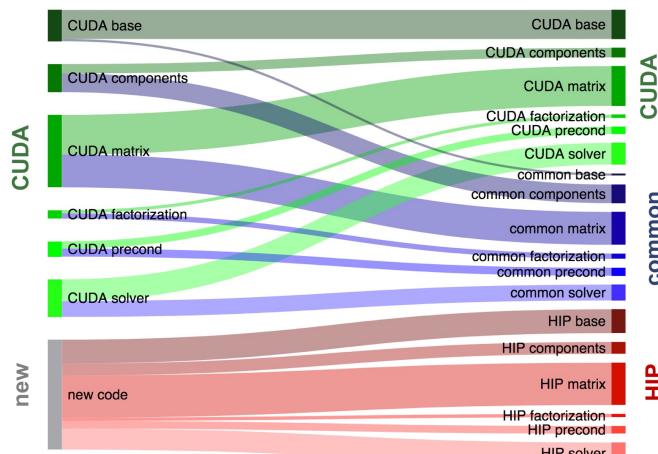


	Functionality	OMP	CUDA
Basic	SpMV	✓	✓
	SpMM	✓	✓
	SpGeMM	✓	✓
Krylov solvers	BiCG	✓	✓
	BICGSTAB	✓	✓
	CG	✓	✓
	CGS	✓	✓
	GMRES	✓	✓
	IDR	✓	✓
	(Block-)Jacobi	✓	✓
	ILU/IC		✓
Preconditioners	Parallel ILU/IC	✓	✓
	Parallel ILUT/ICT	✓	✓
	Sparse Approximate Inverse	✓	✓

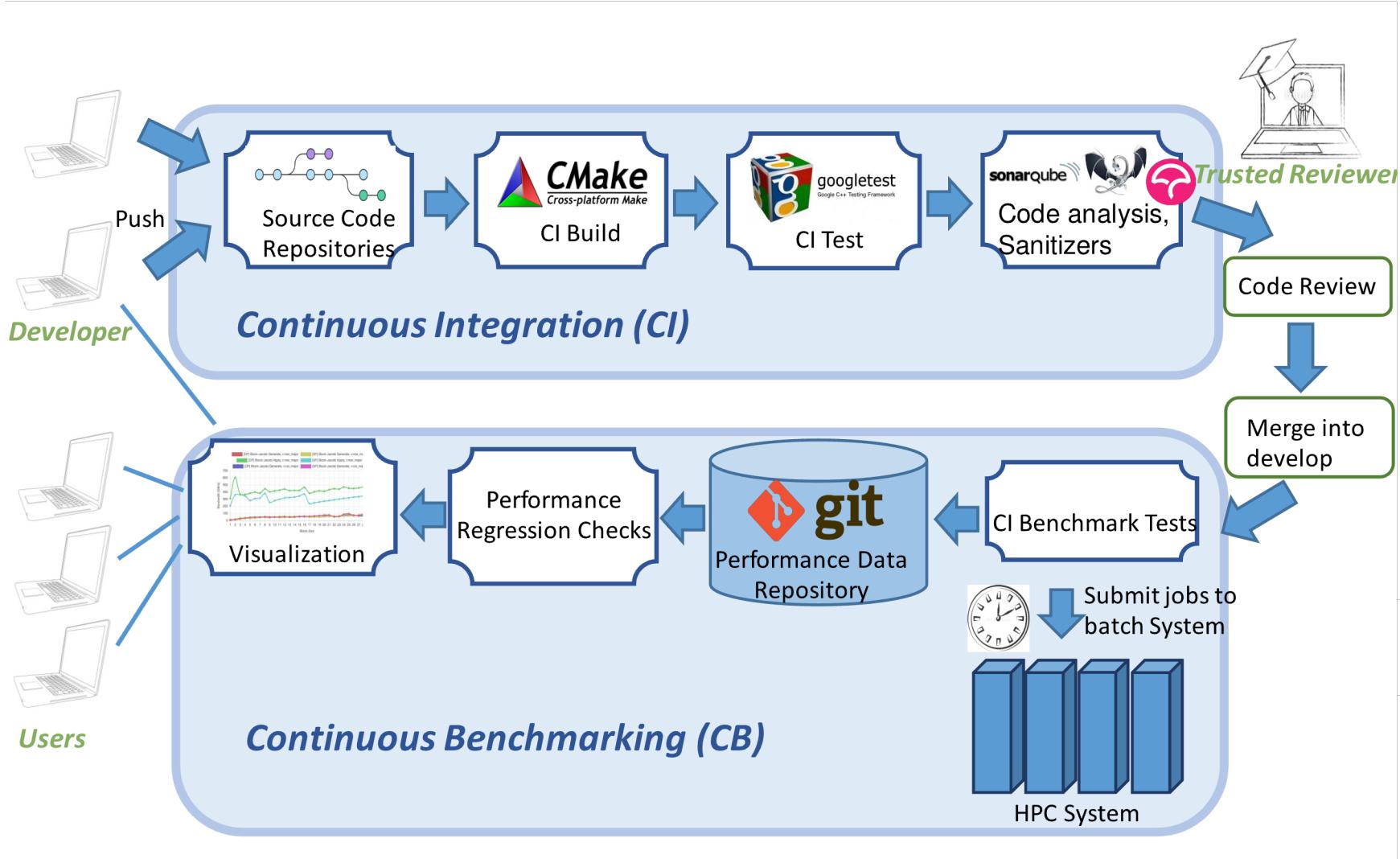


Extending to AMD GPUs

The screenshot shows a GitHub blog post from the Ginkgo repository. The title is "Porting the Ginkgo Package to AMD's HIP Ecosystem". The post discusses the experience of porting CUDA code to AMD's Heterogeneous-compute Interface for Portability (HIP). It includes a section on hardware diversification, portability, and sustainability. The post was published on June 25, 2020, by Hartwig Anzt. Topics covered include better reliability, better planning, testing, and design. The post also features a diagram illustrating the porting process from CUDA to HIP.



Sustainable software development & CI/CD



Sustainable software development & CI/CD

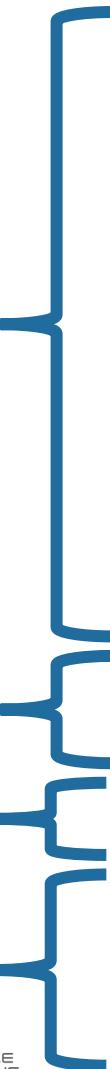
The image shows a screenshot of a CI/CD pipeline interface with several sections:

- Private NVIDIA system:** A vertical list of build jobs for NVIDIA systems.
- Private Intel systems:** A vertical list of build jobs for Intel systems.
- Private or NHR@KIT systems:** A vertical list of build jobs for various systems, including some with NHR@KIT.
- "Quick" overview CI jobs:** A section containing a mix of build, test, and code quality jobs for different systems.
- Private AMD system:** A vertical list of build, test, and code quality jobs for AMD systems.
- NHR@KIT GPU partition (with SLURM):** A vertical list of build, test, and code quality jobs for GPU partitions at NHR@KIT.
- NHR@KIT dedicated CI node:** A vertical list of build, test, and code quality jobs for a dedicated CI node at NHR@KIT.
- Extra jobs, ran before merging:** A section containing additional build and test jobs.

Jobs are color-coded by system type: blue for NVIDIA, green for Intel, purple for AMD, and grey for NHR@KIT. Some jobs have red or green boxes around them, likely indicating specific categories or highlights.

Sustainable software development & CI/CD

NVIDIA



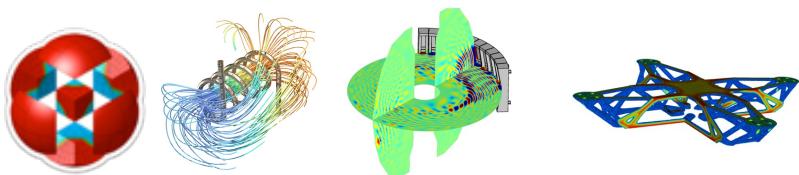
CPU only

Intel

AMD

Build	Test	Deploy
build/cuda90/openmpi/gnu5/llvm39	test/cuda90/openmpi/gnu5/llvm39	deploy/cuda90/openmpi/gnu5/llvm39
build/cuda91/mvapich2/gnu6/llvm40	test/cuda91/mvapich2/gnu6/llvm40	deploy/cuda91/mvapich2/gnu6/llvm40
build/cuda92/mvapich2/gnu7/llvm50/intel2017	test/cuda92/mvapich2/gnu7/llvm50/intel2017	deploy/cuda92/mvapich2/gnu7/llvm50/intel2017
build/cuda100/mvapich2/gnu7/llvm60/intel2018	test/cuda100/mvapich2/gnu7/llvm60/intel2018	deploy/cuda100/mvapich2/gnu7/llvm60/intel2...
build/cuda101/openmpi/gnu8/llvm7/intel2019	test/cuda101/openmpi/gnu8/llvm7/intel2019	deploy/cuda101/openmpi/gnu8/llvm7/intel2019
build/cuda101/openmpi/gnu8/llvm11/intel2019	test/cuda101/openmpi/gnu8/llvm11/intel2019	deploy/cuda101/openmpi/gnu8/llvm11/intel2019
build/cuda102/nompi/gnu8/llvm8/intel2019	test/cuda102/nompi/gnu8/llvm8/intel2019	deploy/cuda102/nompi/gnu8/llvm8/intel2019
build/cuda110/mvapich2/gnu9/llvm9/intel2020	test/cuda110/mvapich2/gnu9/llvm9/intel2020	deploy/cuda110/mvapich2/gnu9/llvm9/intel2020
build/cuda114/openmpi/gnu11/llvm12	test/cuda114/openmpi/gnu11/llvm12	deploy/cuda114/openmpi/gnu11/llvm12
build/nocuda/mvapich2/gnu5/llvm39/intel2018	test/nocuda/mvapich2/gnu5/llvm39/intel2018	deploy/nocuda/mvapich2/gnu5/llvm39/intel2018
build/nocuda/openmpi/gnu9/llvm8	test/nocuda/openmpi/gnu9/llvm8	deploy/nocuda/openmpi/gnu9/llvm8
build/oneapi/openmpi	test/oneapi/openmpi	deploy/oneapi/openmpi
build/rocm40/openmpi/gnu5/llvm50	test/rocm40/openmpi/gnu5/llvm50	deploy/rocm40/openmpi/gnu5/llvm50
build/rocm45/mvapich2/gnu8/llvm8	test/rocm45/mvapich2/gnu8/llvm8	deploy/rocm45/mvapich2/gnu8/llvm8
build/rocm502/openmpi/gnu11/llvm11	test/rocm502/openmpi/gnu11/llvm11	deploy/rocm502/openmpi/gnu11/llvm11

Input from the “first customer”



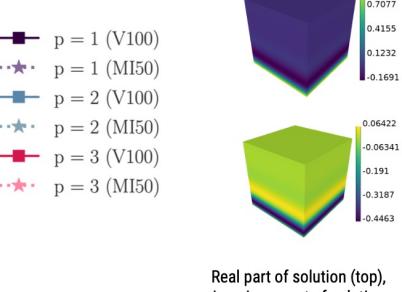
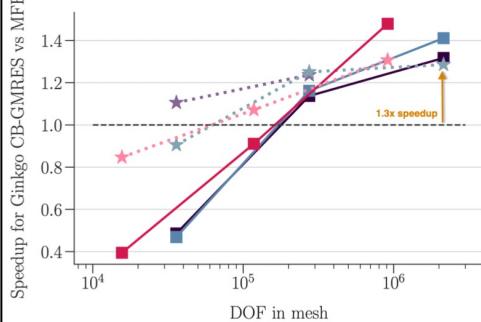
MFEM is a *free, lightweight, scalable C++ library* for finite element methods.

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:

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

with $a = 1$, $b = 1$, $\omega = 10$, $c = 20$



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 functionality

CORE

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



Runtime polymorphism selects the right kernel depending on the target architecture

REFERENCE OpenMP



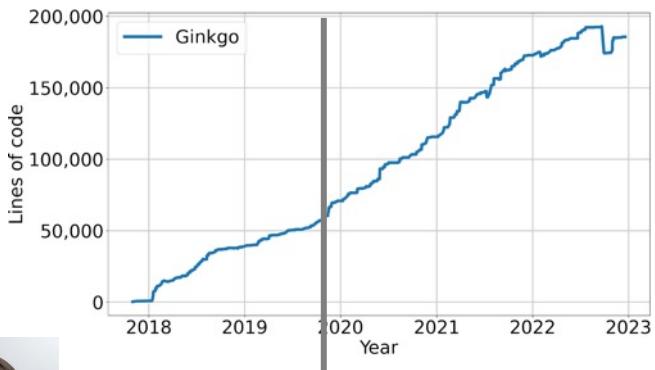
Unit tests check correctness



CUDA



Architecture-optimized kernels

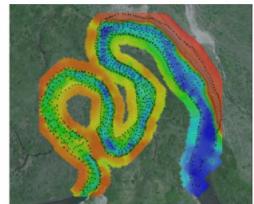


	OMP	CUDA	HIP
Basic			
SpMV	✓	✓	✓
SpMM	✓	✓	✓
SpGeMM	✓	✓	✓
Krylov solvers			
BiCG	✓	✓	✓
BICGSTAB	✓	✓	✓
CG	✓	✓	✓
CGS	✓	✓	✓
GMRES	✓	✓	✓
IDR	✓	✓	✓
(Block-)Jacobi	✓	✓	✓
ILU/IC		✓	✓
Parallel ILU/IC	✓	✓	✓
Parallel ILUT/ICT	✓	✓	✓
Sparse Approximate Inverse	✓	✓	✓
Preconditioners			
On-Device Matrix Assembly	✓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

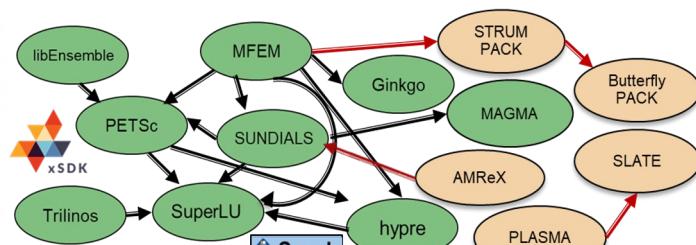


Part of the xSDK effort

xSDK: Extreme-scale Scientific Software Development Kit



Integrated surface-subsurface hydrology simulations of river meanders require the combined use of xSDK packages.



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/>



Library core contains architecture-agnostic functionality

CORE
Infrastructure Algorithms

- Iterative Solvers
- Preconditioners
- ...

Runtime polymorphism selects the right kernel depending on the target architecture

REFERENCE OpenMP

Architecture-optimized kernels

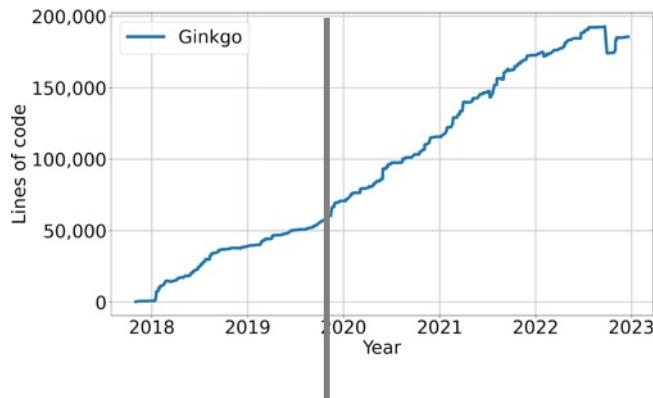
CUDA

HIP

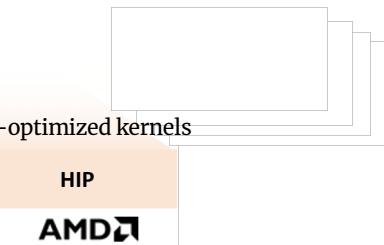


CI/CD

Unit tests check correctness

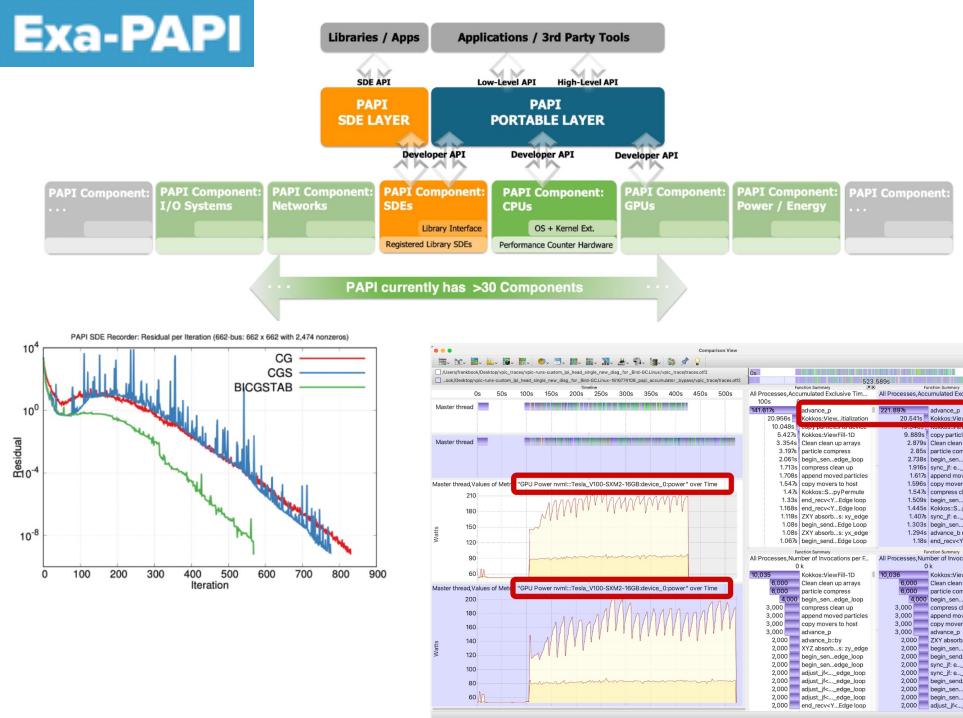


Ginkgo



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

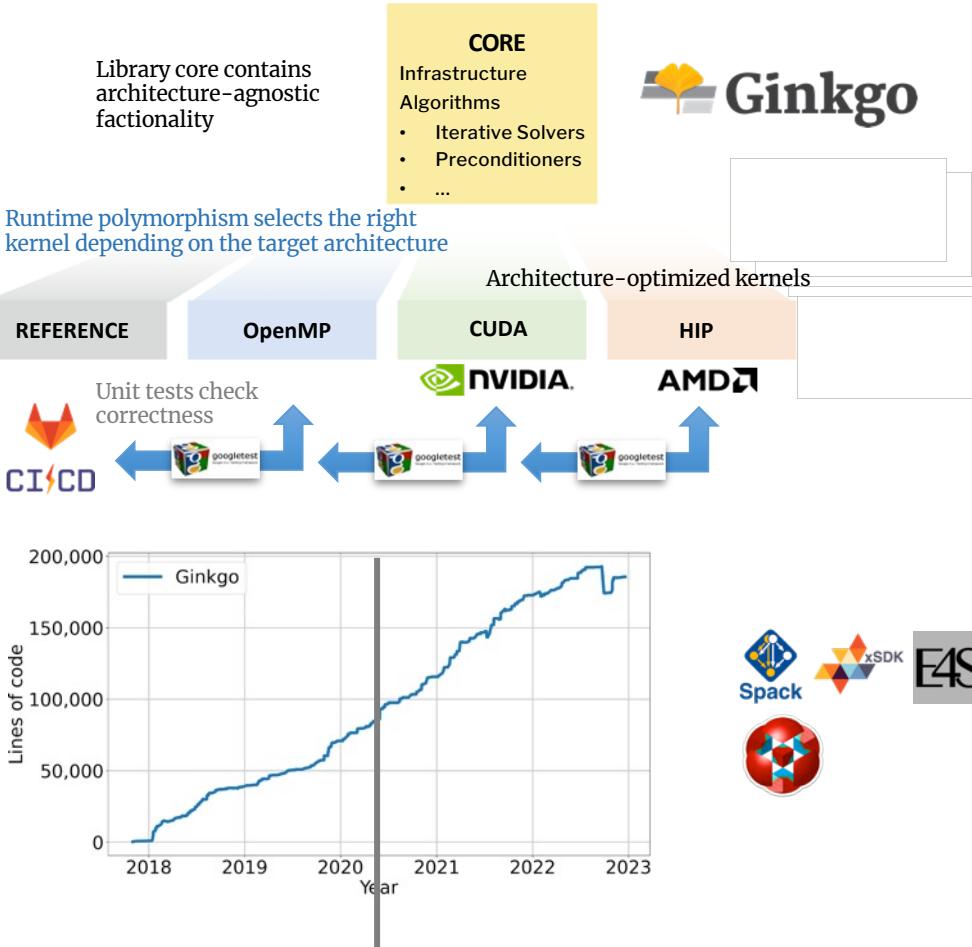
Adding profiling functionality



Library core contains architecture-agnostic functionality

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

Runtime polymorphism selects the right kernel depending on the target architecture

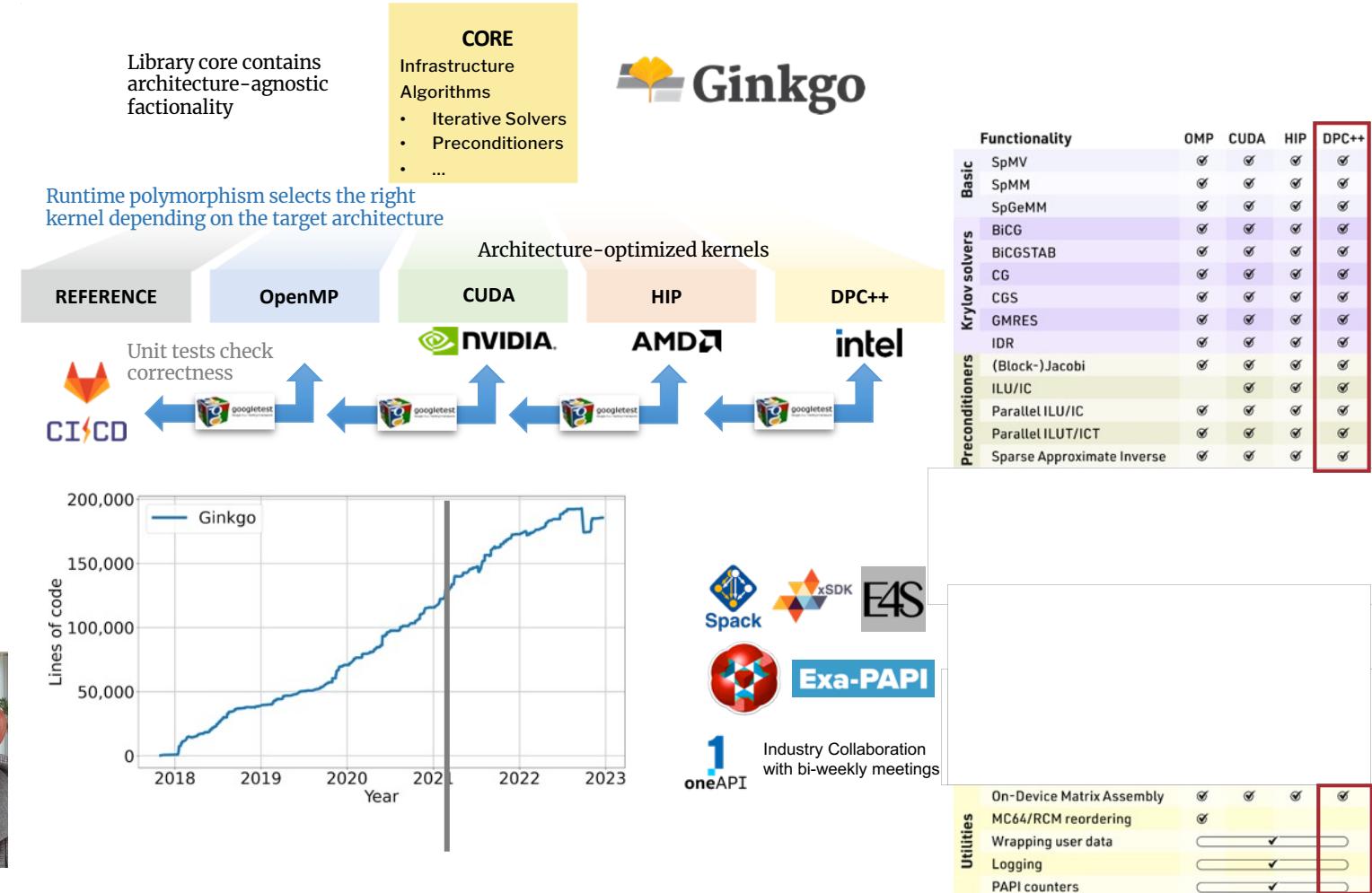


Functionality		OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
	BICG	✓	✓	✓
	BICGSTAB	✓	✓	✓
	CG	✓	✓	✓
Krylov solvers	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
Preconditioners	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
<hr/>				
Utilities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



Extending to Intel GPUs

The screenshot shows the HPC WIRE website with a news article titled "Preparing for the Arrival of Intel's Discrete High-Performance GPUs" by Hartwig Anzt. The article discusses the upcoming arrival of Intel's discrete GPUs. Below the article, there is a GitHub repository page for "yhmtsai / try_oneapi". The repository has 70 commits and was last updated on Aug 7, 2022. It contains several files like arg_struct, atomic, check_unit, classical_csr, clinfo, coop_cuda, and coop_draft. There is also a "format update" commit. The repository has 0 stars, 2 forks, and 2 watching.



Extending to Intel GPUs



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

cuBLAS backend (and potentially other domains) fails with latest LLVM builds #223

Summary

As first observed in #219 many tests in cuBLAS backend is failing with latest LLVM builds.

Version

I have tried LLVM commit: 66361038b63caaae566fc9648f5da50b74222b83 and got the below tests failing (showing only a few of them)

```

1 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
3 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.RealDoublePrecision/Column_Major_TITAN_RTX (Failed)
7 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.ComplexDoublePrecision/Column_Major_TITAN_RTX (Failed)
17 - BLAS/RT/IamaXTestSuite/IamaXTests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
19 - BLAS/RT/IamaXTestSuite/IamaXTests.RealDoublePrecision/Column_Major_TITAN_RTX (Failed)
23 - BLAS/RT/IamaXTestSuite/IamaXTests.ComplexDoublePrecision/Column_Major_TITAN_RTX (Failed)
27 - BLAS/RT/AsmTestSuite/AsmTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
35 - BLAS/RT/AsmTestSuite/AsmTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
67 - BLAS/RT/ScalTestSuite/ScalTests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
81 - BLAS/RT/ScalTestSuite/ScalTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
85 - BLAS/RT/ScalTestSuite/ScalTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)

```

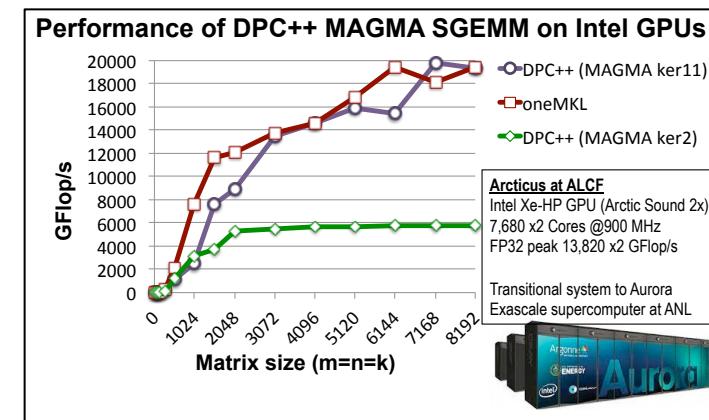
From [DPCPP AoT documentation](#), not clear:

- The options are also required at linking time? Unused in files without kernels?
- Any example of other projects integrating AoT in a CMake setup?

[Intel Compiler \(Fortran/C/C++/I0\) - Intel Discrete GPU Accelerator - Joint Laboratory for System Evaluation \(anl.gov\)](#)

hang_atomic_on_local

Ticket number: CMPLRLLVM-36572 (works in PVC, but still fails on ATS node)
related to driver not compiler self



... but also docker image contributions and bug fixes!

ginkgohub/oneapi:cuda11.6

DIGEST: sha256:0bc4c10d79a75b183ac1deafcd753365c6e1a94edc3046a9a0eb8ba2d7b9d94

OS/ARCH: linux/amd64
COMPRESSED SIZE: 6.63 GB
LAST PUSHED: 22 days ago by [yhmtsa](#)

Fix cuda/hip backend location #219

Merged mkrainuk merged 2 commits into [oneapi-src:develop](#) from [yhmtsa:fix_cuda_backend_location](#) 20 days ago

Description

From [intel/llvm#6407](#), it moves almost all headers from CL/sycl to sycl
I followed [#199](#) way
make the header can use sycl/* if they exist and allow the old intel llvm.
I also update the CL/sycl.hpp which are not changed before.

All Submissions

Do all unit tests pass locally? Attach a log. A: It is a compiling issue.
✓ Have you formatted the code using clang-format?

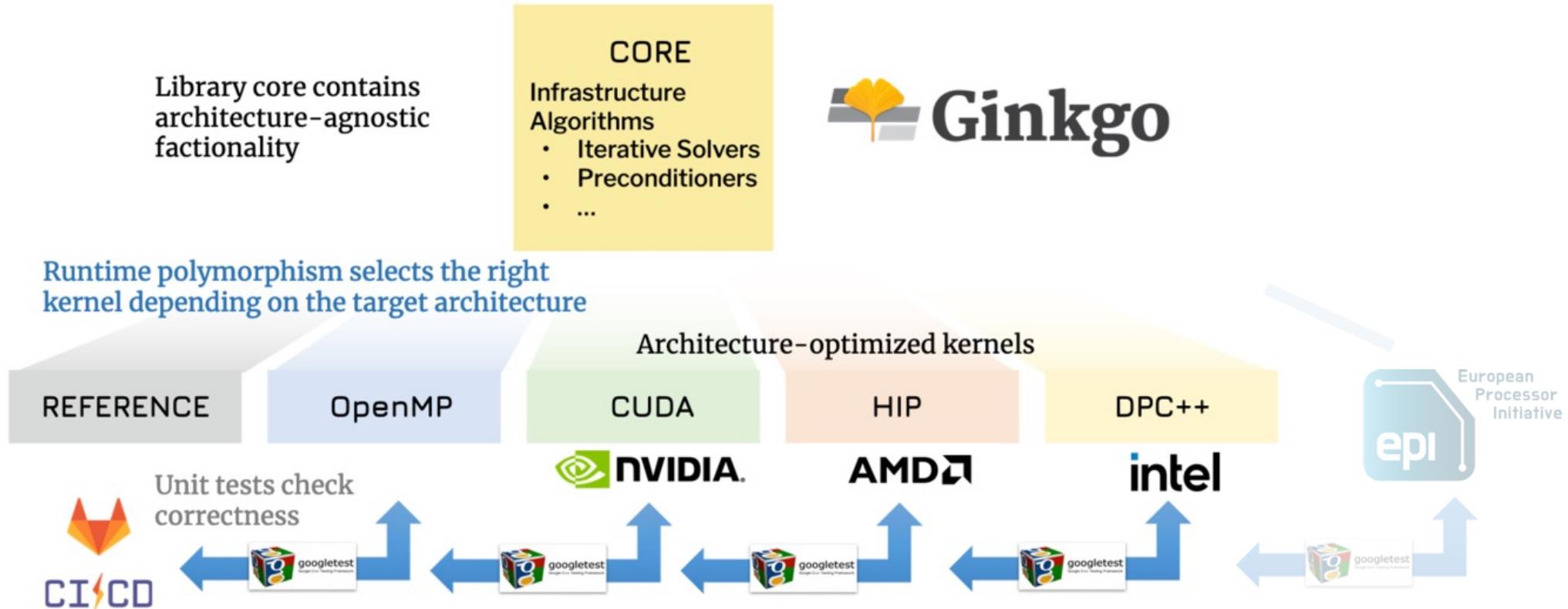
Bug fixes

Have you added relevant regression tests? A: It is a compiling issue.
✓ Have you included information on how to reproduce the issue (either in a GitHub issue or in this PR)?
Reproduce:
compile the latest intel llvm and this repo, it will not be able to compile due to missing headers.

yhmtsa added 2 commits 2 months ago
use the correct sycl path after [intel/llvm#6407](#)
fix the missing sycl/sycl.hpp

mkmruiel commented on Aug 1
@yhmtsa Thanks for the PR. Is the description from sycl/CL to CL correct? My understanding is all header files moved

Portability as central design principle

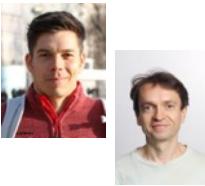


Focus efforts as lightweight tool in ECP to address challenges



Focus efforts

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



Library core contains architecture-agnostic functionality

CORE

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



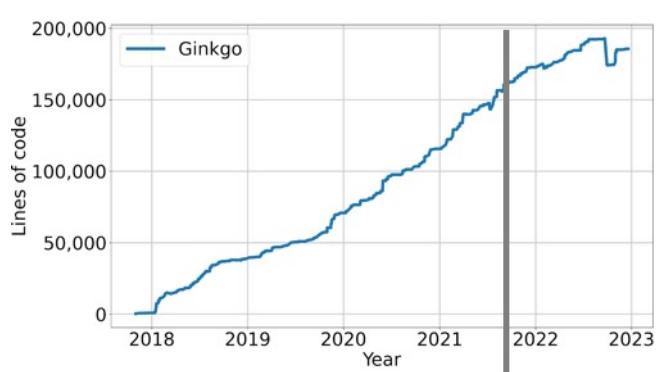
Runtime polymorphism selects the right kernel depending on the target architecture

REFERENCE

OpenMP



Unit tests check correctness



CUDA



HIP



DPC++

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



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 E. 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 19, 2021

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

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

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



Library core contains architecture-agnostic functionality

CORE

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



Runtime polymorphism selects the right kernel depending on the target architecture

REFERENCE

OpenMP



Unit tests check correctness



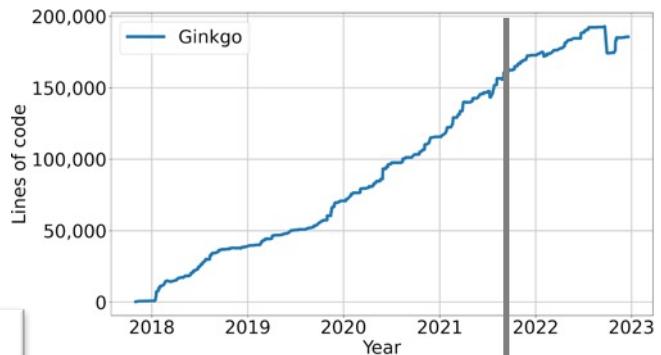
CUDA



HIP



DPC++



Utilities

On-Device Matrix Assembly	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
MC64/RCM reordering	<input checked="" type="checkbox"/>			
Wrapping user data		<input checked="" type="checkbox"/>		
Logging		<input checked="" type="checkbox"/>		
PAPI counters		<input checked="" type="checkbox"/>		

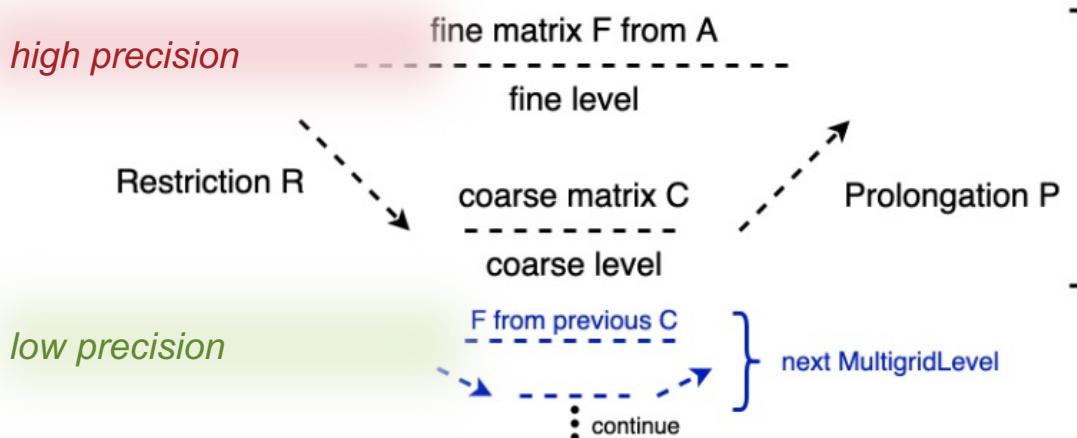
Mixed precision AMG on GPUs

- Preconditioning iterative solvers

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

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

- Mixed Precision Multigrid Preconditioner



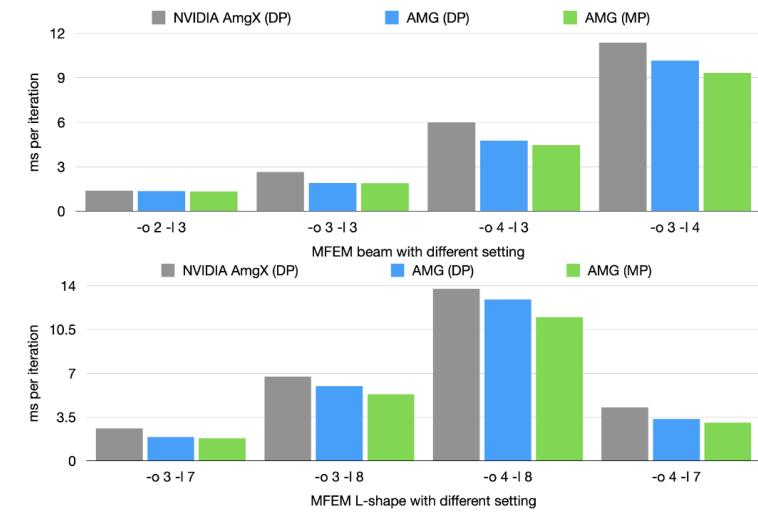
```
1 multigrid::build()
2   .with_max_levels(10u) // equal to NVIDIA/AMGX 11 max levels
3   .with_min_coarse_row(64u)
4   .with_pre_smoothen(sm, sm_f)
5   .with_mg_level(pgm, pgm_f)
6   .with_level_selector(
7     [](const size_type level, const LinOp*) -> size_type {
8       // Only the first level is generated by MultigridLevel(double).
9       // The subsequent levels are generated by MultigridLevel(float).
10      return level >= 1 ? 1 : 0;
11    })
12   .with_coarsest_solver(coarsest_solver_f)
```



MultigridLevel



Mike Tsai



Mixed precision AMG on GPUs



- Mixed precision
- batched



ICL UTK @ICL_UTK · Sep 13

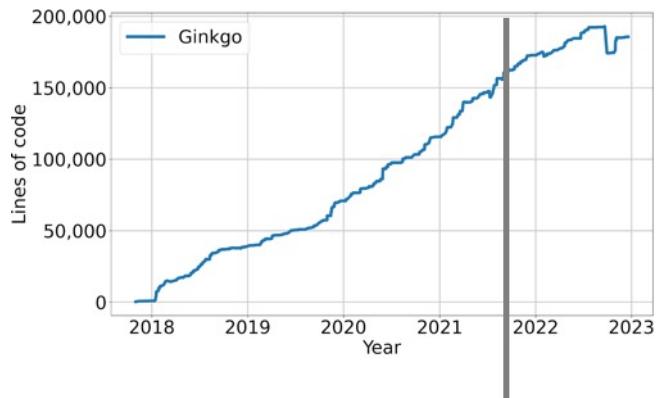
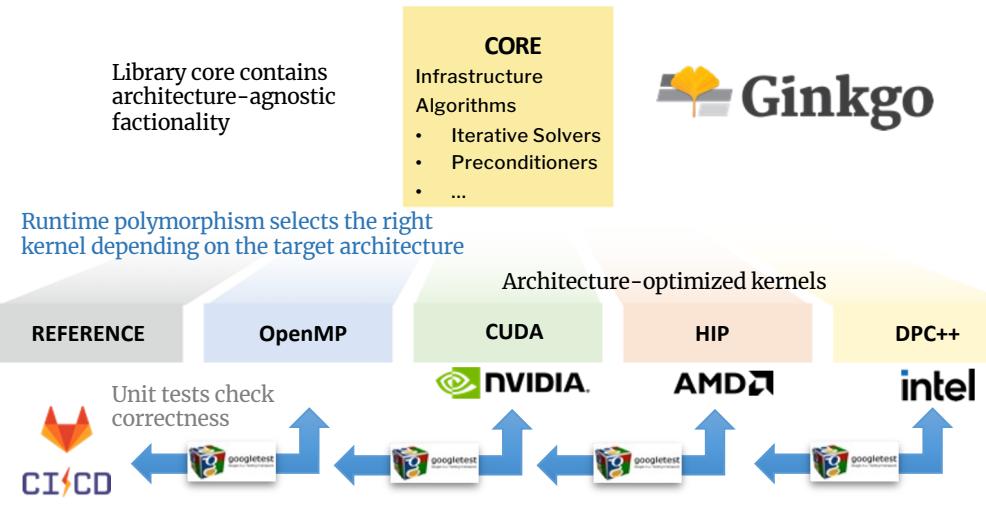
Congratulations to Yu-Hsiang Mike Tsai from [@KITKarlsruhe](#), in collaboration with ICL's Natalie Beams and [@HartwigAnzt](#)! Their paper "Mixed Precision Algebraic Multigrid on GPUs" took home a best paper award at PPAM2022. ppam.edu.pl



Library core contains architecture-agnostic functionality

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

Runtime polymorphism selects the right kernel depending on the target architecture



Industry Collaboration
with bi-weekly meetings

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

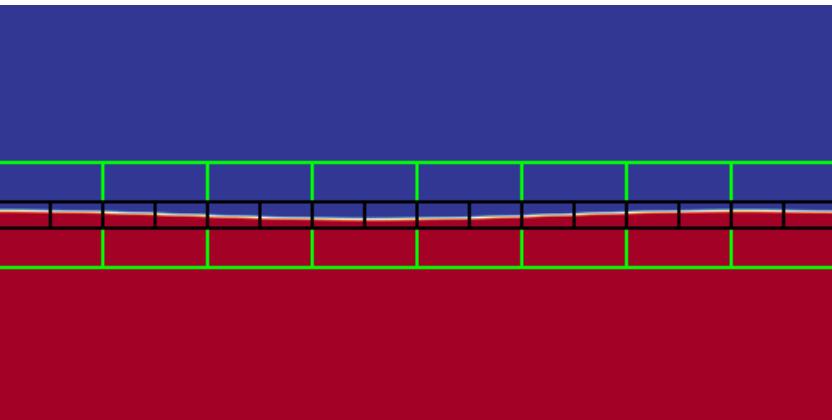


Batched focus effort – Combustion Simulations

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>

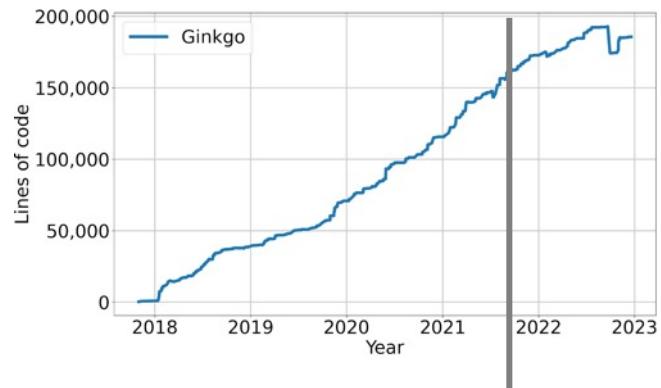
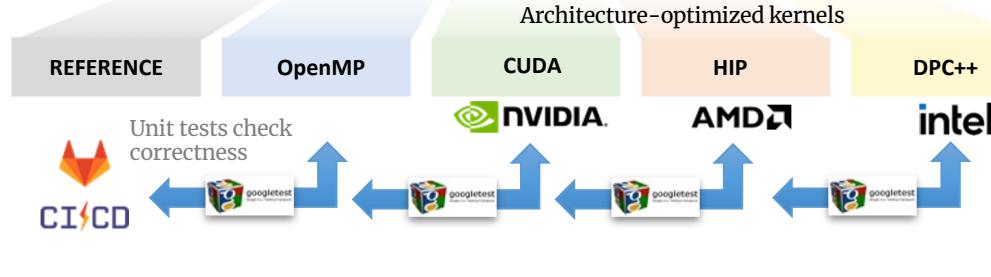


Problem	Size	Non-zeros (A)	Non-zeros (L+U)
dodecane_lu	54	2,332 (80%)	2,754 (94%)
drm19	22	438 (90%)	442 (91%)
gri12	33	978 (90%)	1,018 (93%)
gri30	54	2,560 (88%)	2,860 (98%)
isoctane	144	6,135 (30%)	20,307 (98%)
lidryer	10	91 (91%)	91 (91%)

Library core contains architecture-agnostic functionality

CORE
Infrastructure Algorithms
• Iterative Solvers
• Preconditioners
• ...

Runtime polymorphism selects the right kernel depending on the target architecture

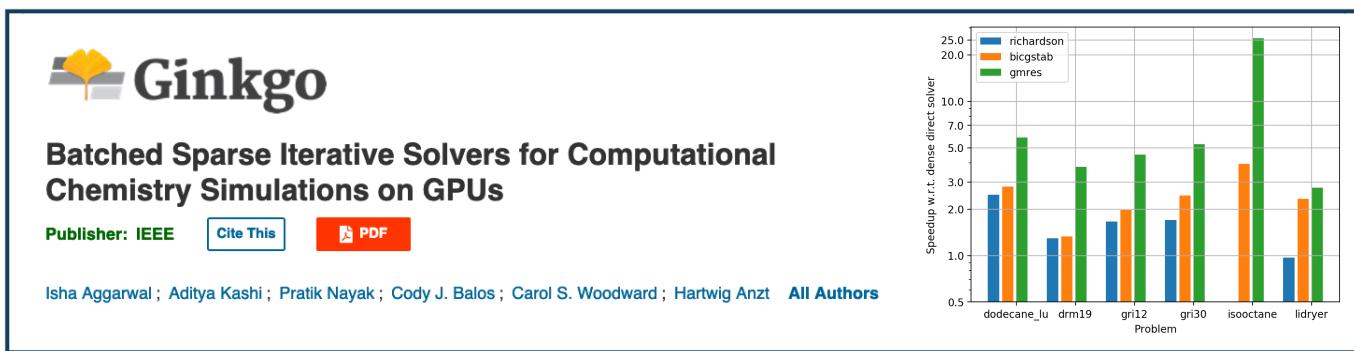
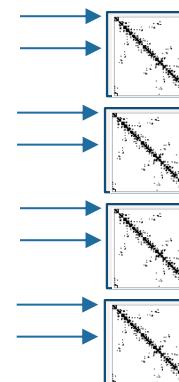
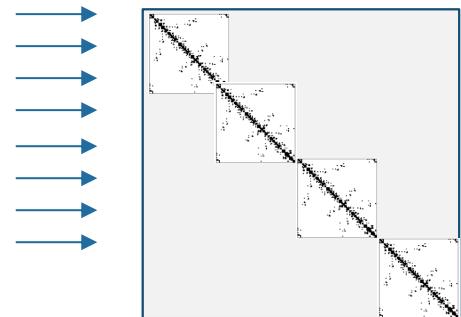
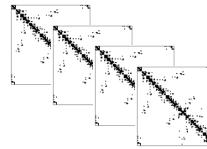


	OMP	CUDA	HIP	DPC++
Basic				
SpMV	✓	✓	✓	✓
SpMM	✓	✓	✓	✓
SpGeMM	✓	✓	✓	✓
Krylov solvers				
BiCG	✓	✓	✓	✓
BICGSTAB	✓	✓	✓	✓
CG	✓	✓	✓	✓
CGS	✓	✓	✓	✓
GMRES	✓	✓	✓	✓
IDR	✓	✓	✓	✓
(Block-)Jacobi	✓	✓	✓	✓
ILU/IC		✓	✓	✓
Parallel ILU/IC	✓	✓	✓	✓
Parallel ILUT/ICT	✓	✓	✓	✓
Sparse Approximate Inverse	✓	✓	✓	✓
Preconditioners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

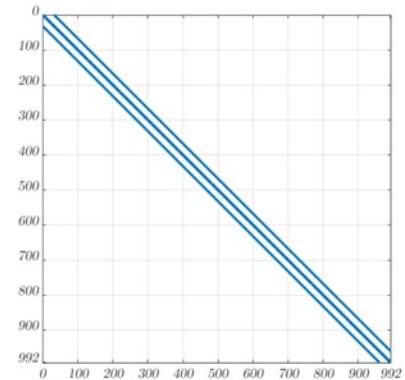
- Many sparse problems of medium size have to be solved concurrently.
 - ~ 50 – 2,000 unknowns, < 50% dense;
 - All sparse systems may share the same sparsity pattern;
 - An approximate solution may be acceptable (e.g., inside a non-linear solver);
- One solution is to arrange the individual systems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



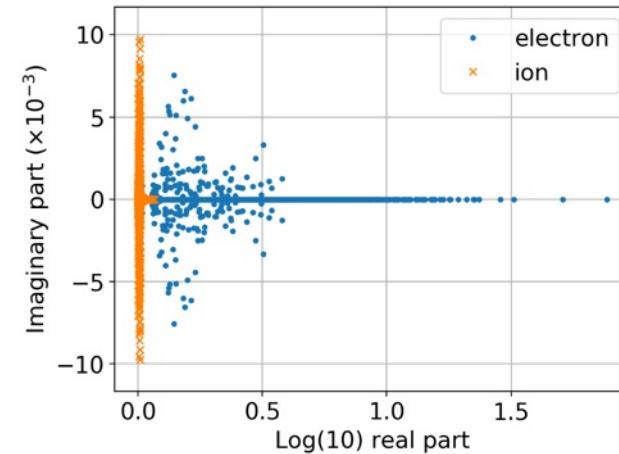
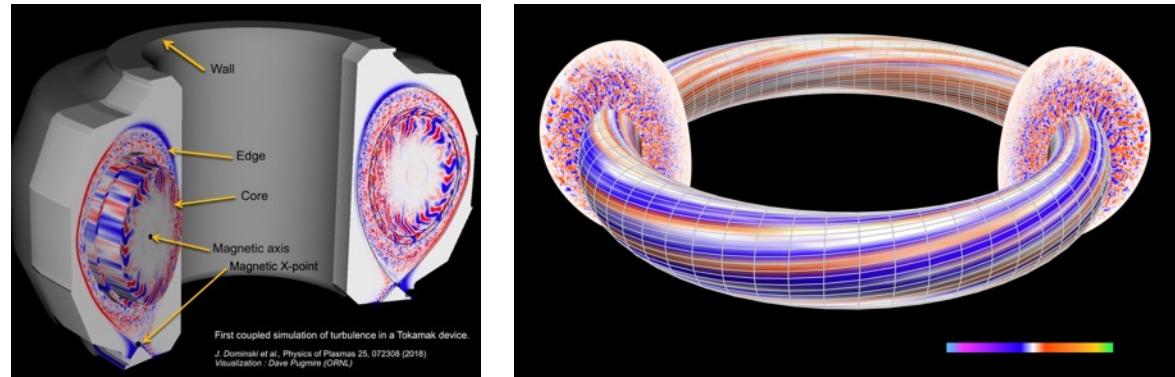
Batched focus effort – Fusion Plasma Simulations

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

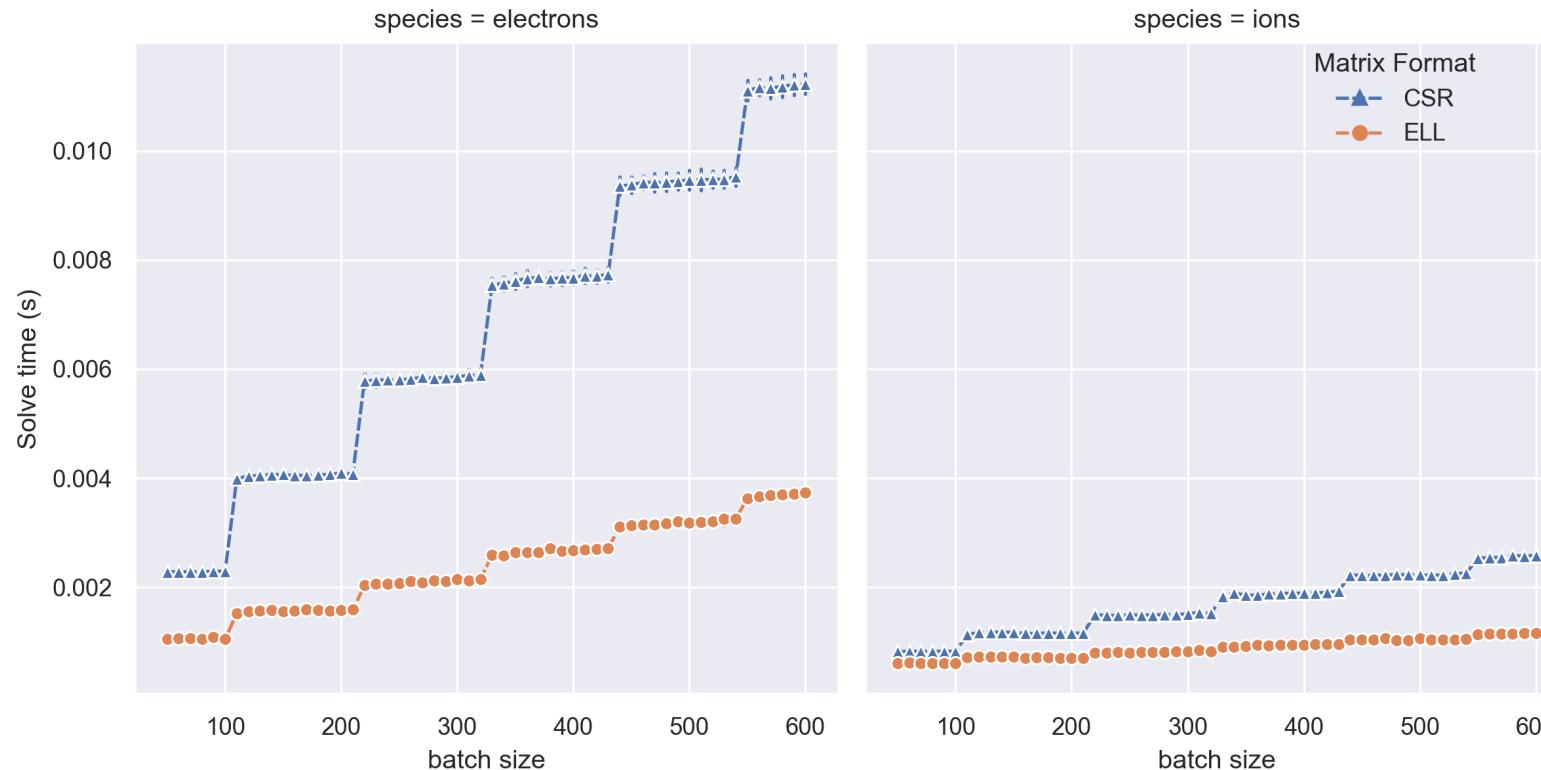


- Two species
- Ions easy to solve
- Electrons hard to solve
- Banded matrix structure
- Non-symmetric, need BiCGSTAB
- $n = \sim 1,000$
- $nz = \sim 9,000$



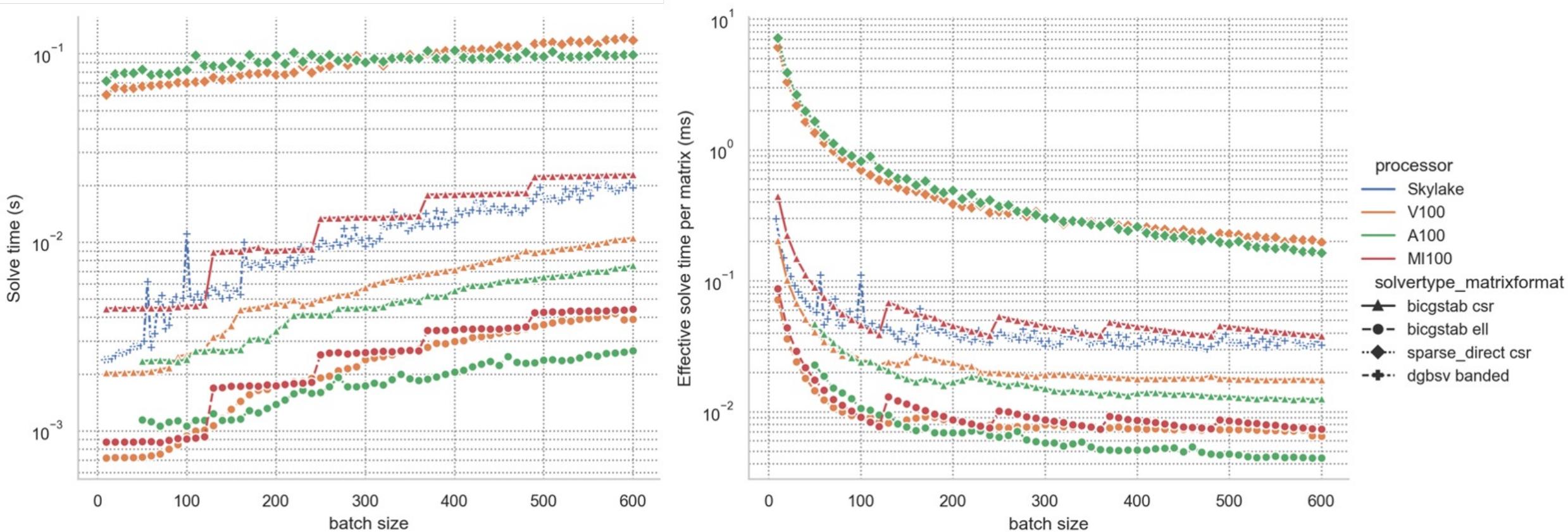
Batched focus effort – Fusion Plasma Simulations

NVIDIA A100 GPU



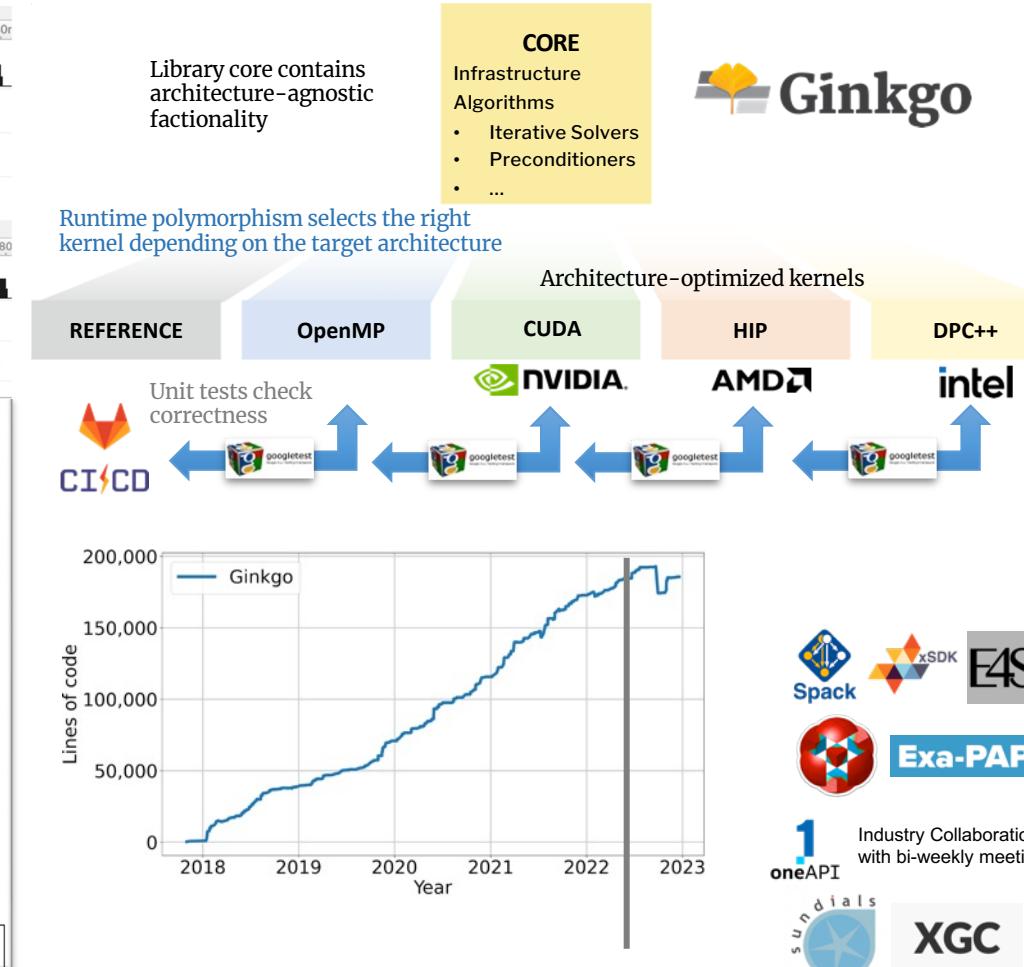
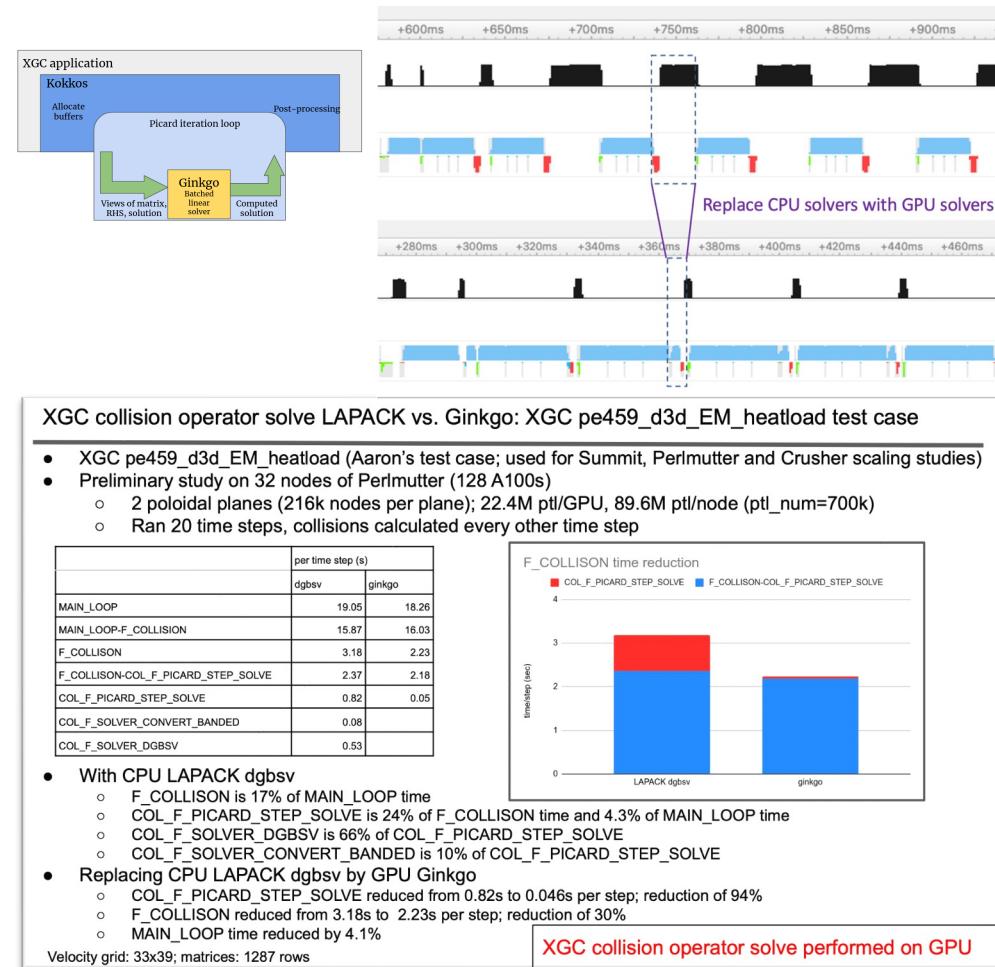
- Ions easy to solve
- Electrons hard to solve
- ELL format more suitable
- “step pattern” for runtime filling up multiprocessors

Batched focus effort – Fusion Plasma Simulations

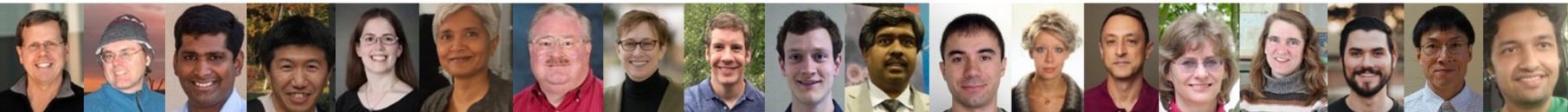


Aditya Kashi, Pratik Nayak, Dhruva Kulkarni, Aaron Scheinberg, Paul Lin, and Hartwig Anzt. **Batched sparse iterative solvers on gpu for the collision operator for fusion plasma simulations**. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 157–167. IEEE, 2022.

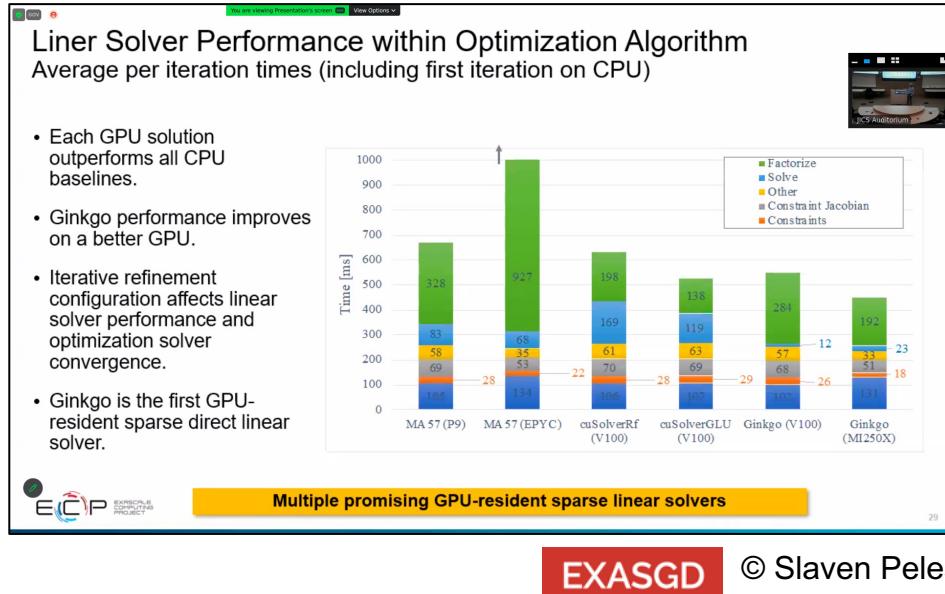
Batched focus effort – Fusion Plasma Simulations



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

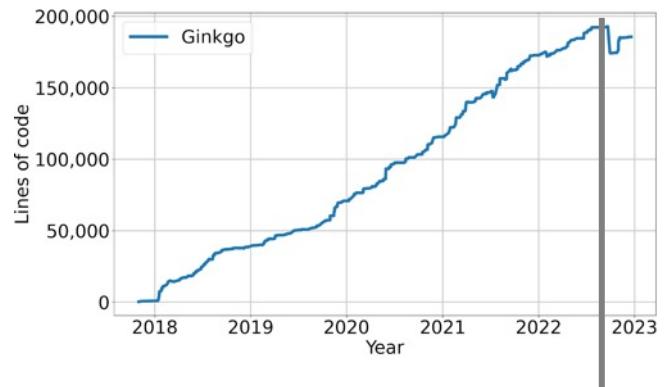


Library core contains architecture-agnostic functionality

CORE
Infrastructure Algorithms

- Iterative Solvers
- Preconditioners
- ...

Runtime polymorphism selects the right kernel depending on the target architecture



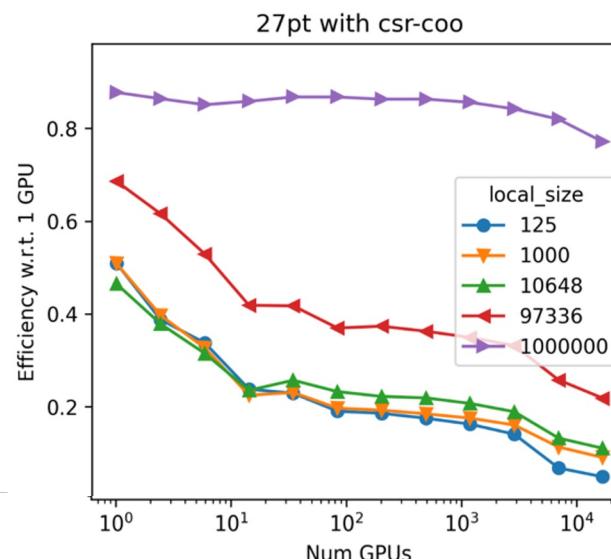
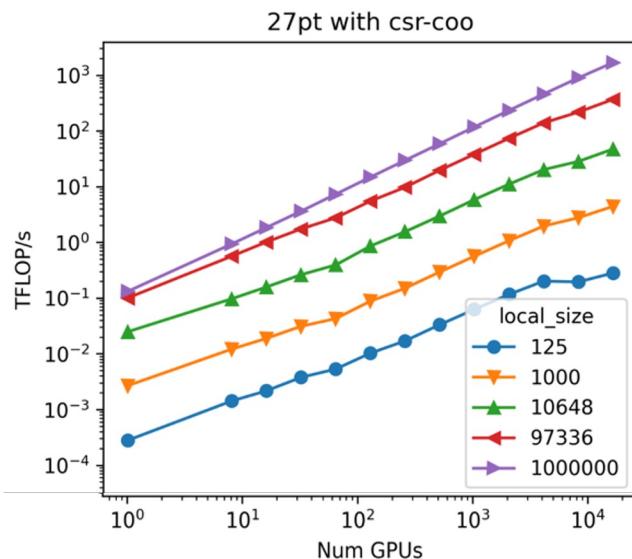
	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
Preconditioners	IDR	✓	✓	✓
	(Block-)Jacobi	✓	✓	✓
	ILU/IC	✓	✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
Batched	Batched BiCGSTAB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

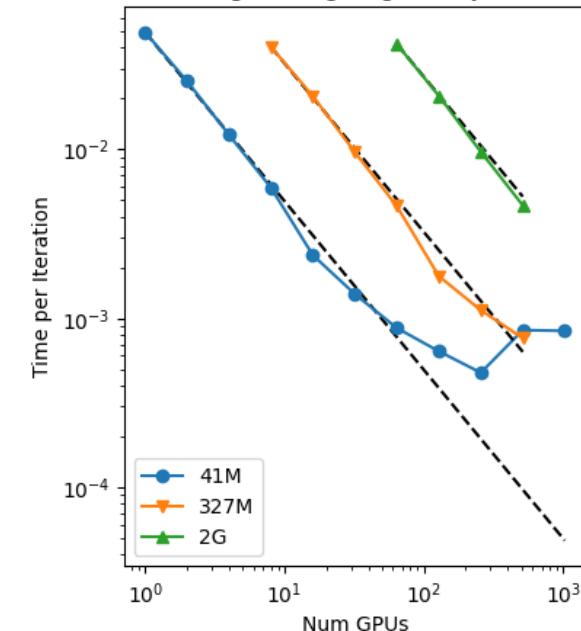
Weak scaling: problem size increases with parallel resources

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

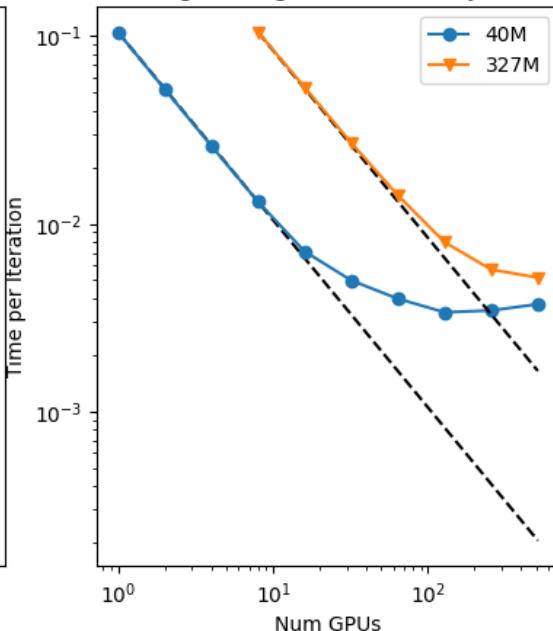


Strong scaling: problem size constant

Strong Scaling - Cg Local Jacobi



Strong Scaling - Gmres Local Jacobi



“Now” – Near completion of ECP

- Sustainable software design ready for the addition of new backends.
- EuroHPC Project MICROCARD uses Ginkgo



- BMBF PDExa project uses Ginkgo

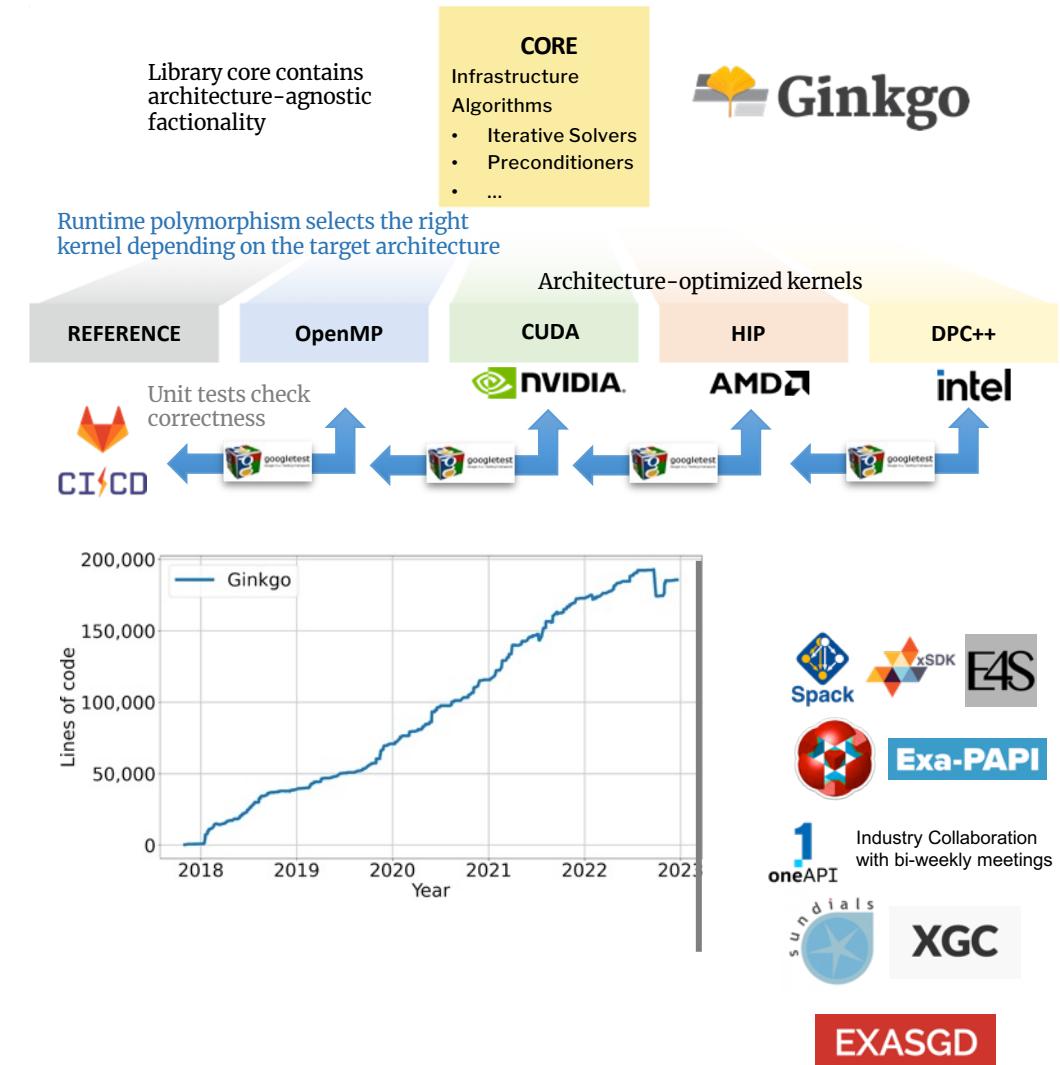


- BMBF ExaSIM project uses Ginkgo



The Open Source CFD Toolbox

<https://exasim-project.com>



	OMP	CUDA	HIP	DPC++
Basic				
SpMV	✓	✓	✓	✓
SpMM	✓	✓	✓	✓
SpGeMM	✓	✓	✓	✓
Krylov solvers				
BiCG	✓	✓	✓	✓
BICGSTAB	✓	✓	✓	✓
CG	✓	✓	✓	✓
CGS	✓	✓	✓	✓
GMRES	✓	✓	✓	✓
IDR	✓	✓	✓	✓
(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	✓	✓	✓	✓
Symbolic Cholesky	✓	✓	✓	✓
Numeric Cholesky				UNDER DEVELOPMENT
Symbolic LU	✓	✓	✓	
Numeric LU	✓	✓	✓	
Sparse TRSV	✓	✓	✓	
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.