# Ginkgo – a platform-portable math library responding to the needs of the US Exascale Computing Project

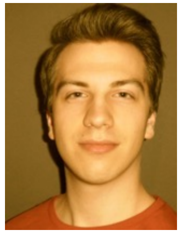
EXASCALE COMPUTING PROJECT

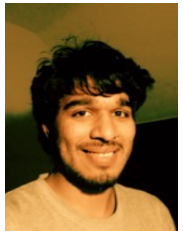
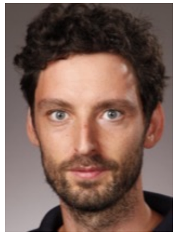
Approved for public release

Hartwig Anzt, University of Tennessee





National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

# The Design of an ECP Math Library

MAGMA SPARSE

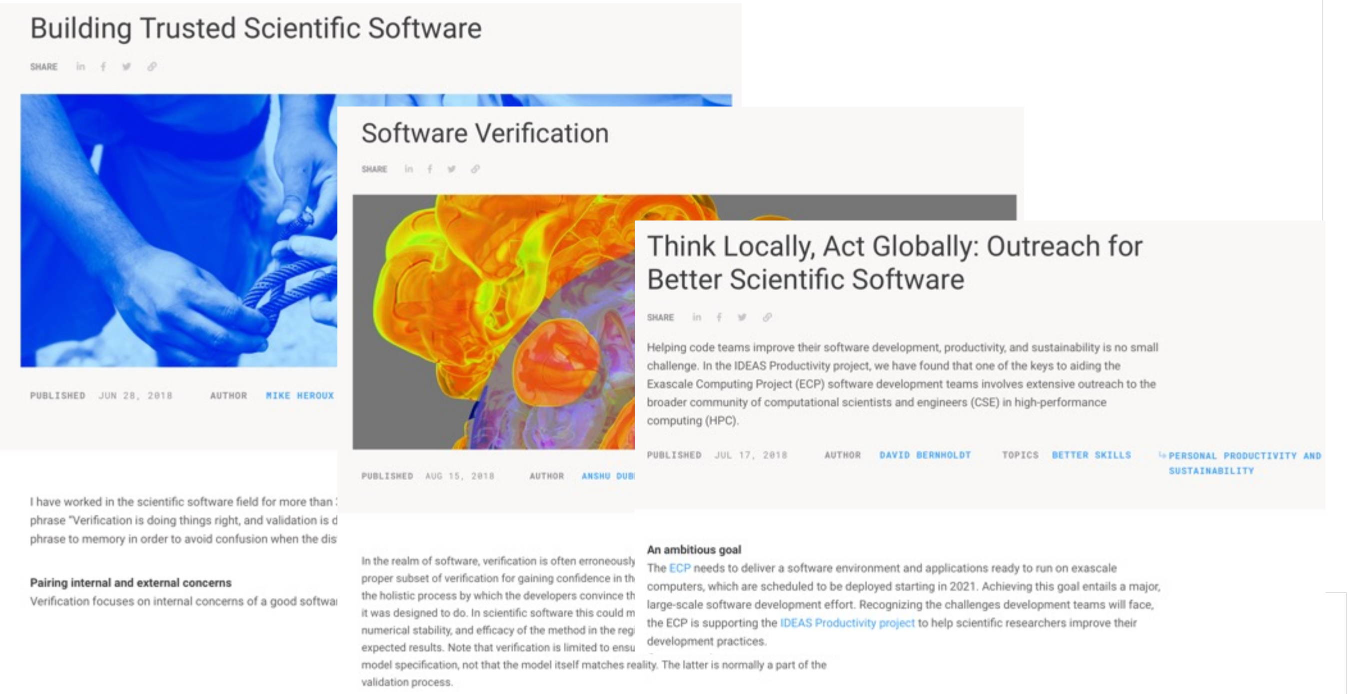
MAGMA-sparse as a “child” of MAGMA explores the development of sparse linear algebra for NVIDIA GPUs.

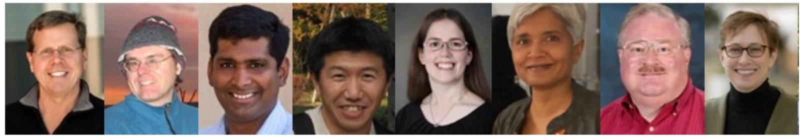
Design considerations for Ginkgo

- 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



ECP EXASCALE COMPUTING PROJECT

# The Design of an ECP Math Library

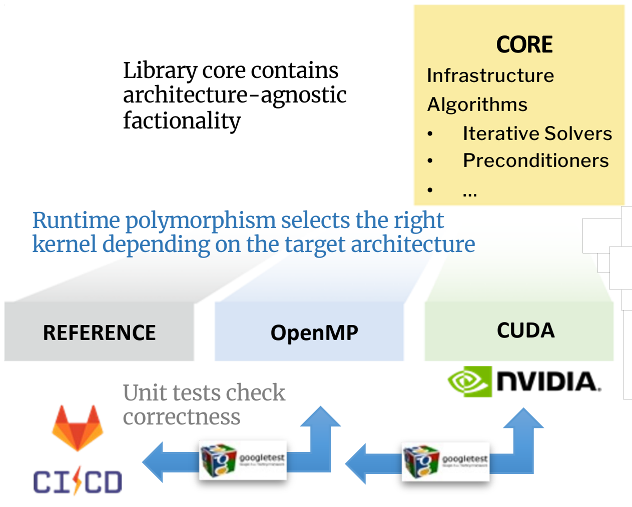
MAGMA SPARSE

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

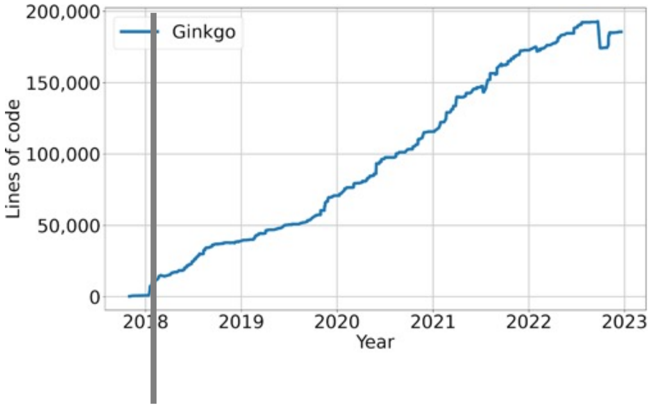
Design considerations for Ginkgo

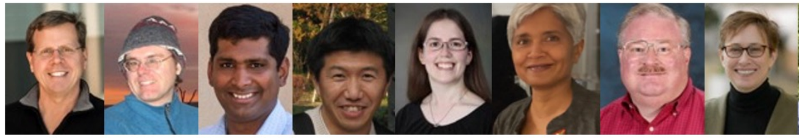
- Platform Portability
- Performance
- Rapid integration of new algorithms
- xSDK / E4S Community Policies
- BSSw expertise / experience
- Modern C++
- CI/CD and unit testing
- Open source & permissive licensing

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




Ginkgo



EXASCALE COMPUTING PROJECT

# The Design of an ECP Math Library

Library core contains architecture-agnostic factionality

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

Ginkgo

| 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 | ✓ | ✓ |

Runtime polymorphism selects the right kernel depending on the target architecture

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

NVIDIA

Unit tests check correctness

CI/CD

googletest googletest

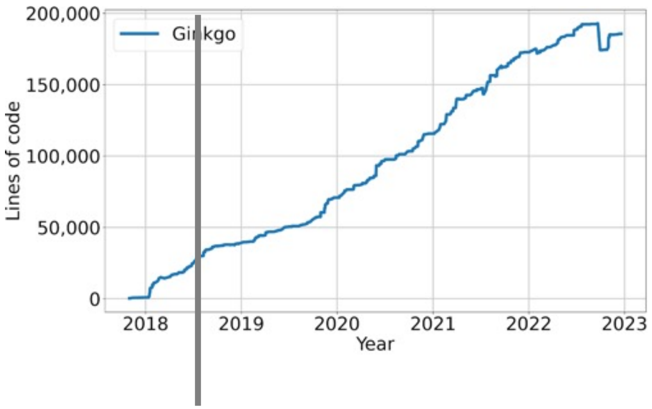
**Linear Operator Interface**

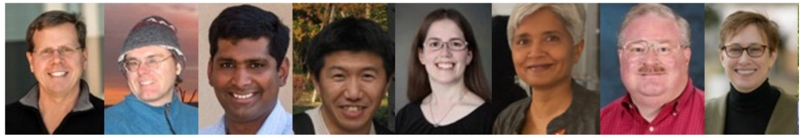
*We express everything as Linear Operator.*
- *Internally, we leverage C++ class inheritance.*
- *Applications can apply any functionality as a linear operator.*

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

All of them can be expressed as

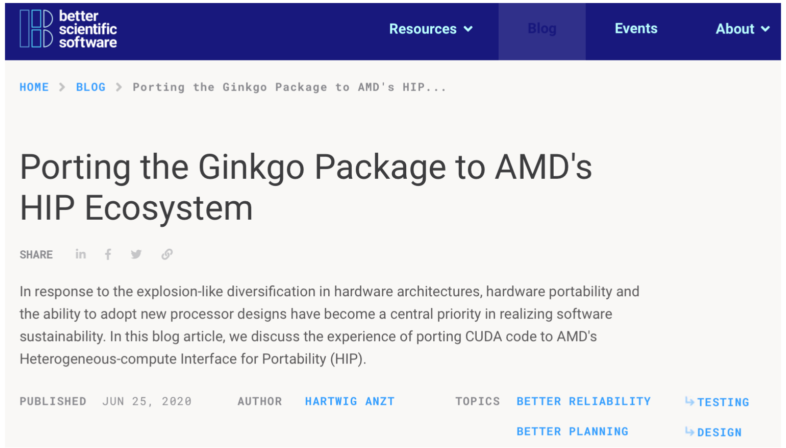
Application of a linear operator* (LinOp) $L : \mathbb{F}^m \to \mathbb{F}^m$



EXASCALE COMPUTING PROJECT

# The Design of an ECP Math Library


better scientific software — Resources, Blog, Events, About

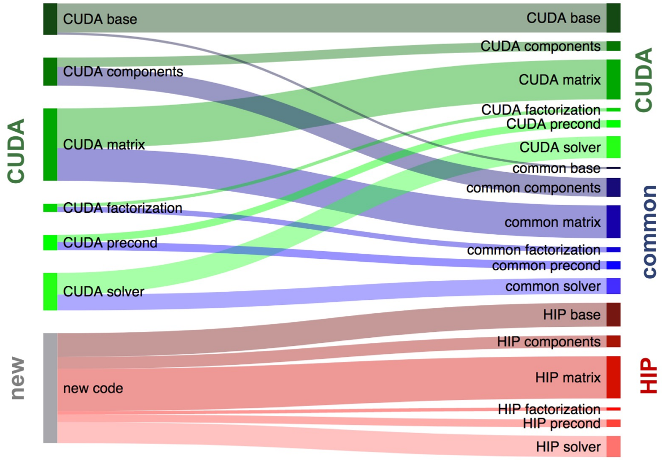
HOME > BLOG > Porting the Ginkgo Package to AMD's HIP...

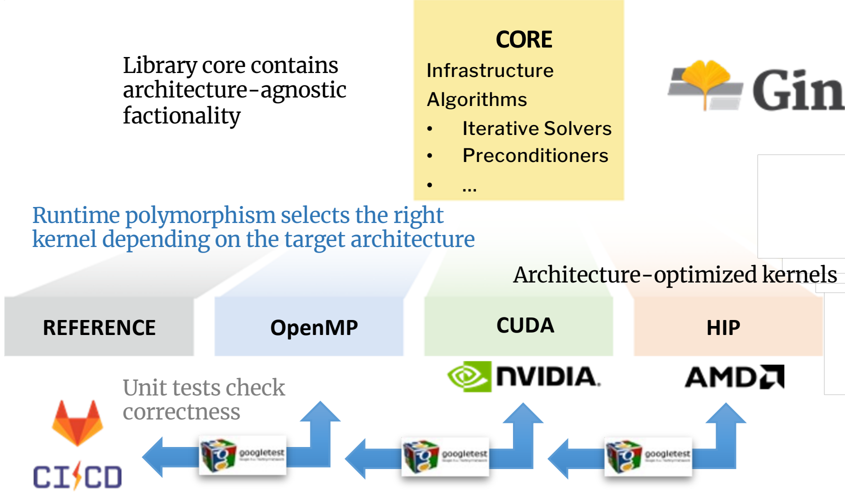
## Porting the Ginkgo Package to AMD's HIP Ecosystem

SHARE

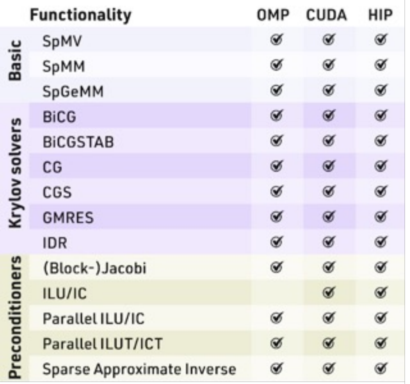
In response to the explosion-like diversification in hardware architectures, hardware portability and the ability to adopt new processor designs have become a central priority in realizing software sustainability. In this blog article, we discuss the experience of porting CUDA code to AMD's Heterogeneous-compute Interface for Portability (HIP).

PUBLISHED JUN 25, 2020 AUTHOR HARTWIG ANZT TOPICS BETTER RELIABILITY ↳TESTING BETTER PLANNING ↳DESIGN

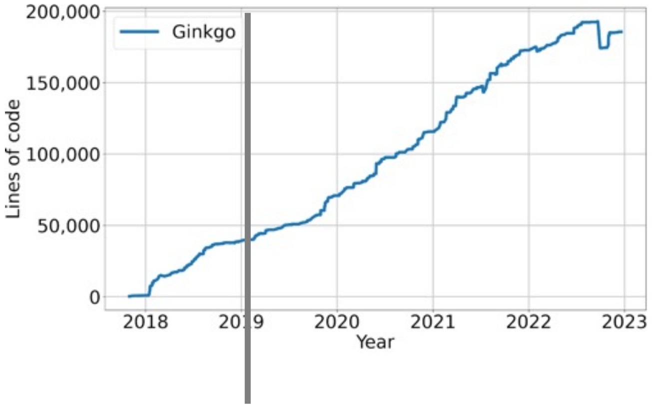



Ginkgo



| | 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

ECP EXASCALE COMPUTING PROJECT

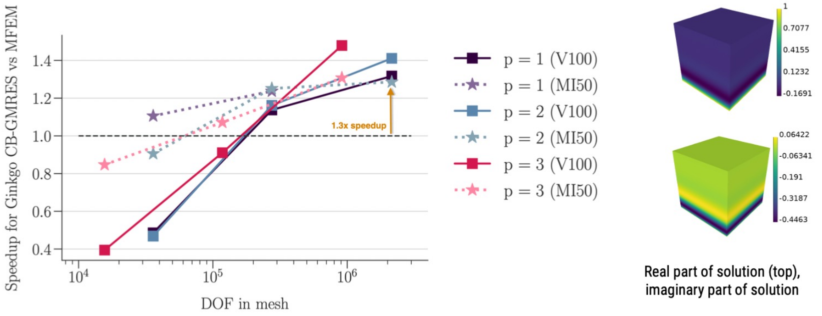
# The Design of an ECP Math Library

## 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:

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

with a = 1, b = 1, ω = 10, c = 20




Real part of solution (top), imaginary part of solution

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.

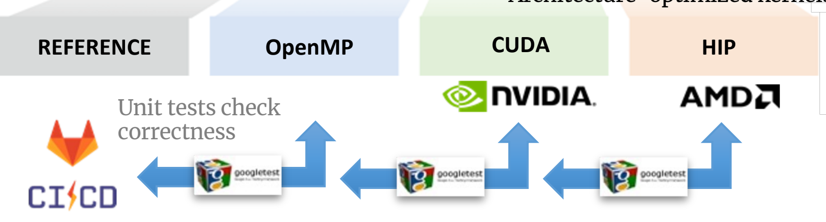
Library core contains architecture-agnostic factionality

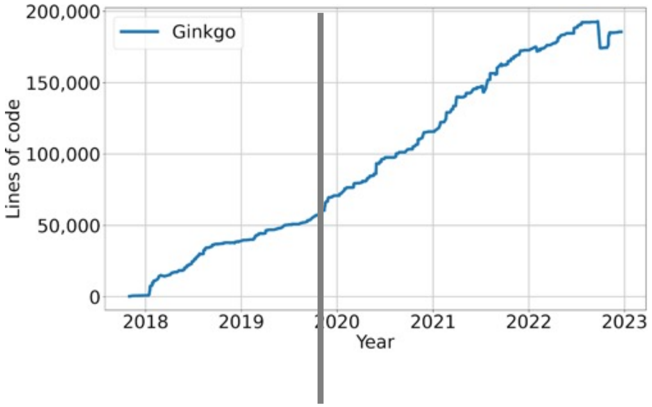
**CORE**
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

Architecture-optimized kernels

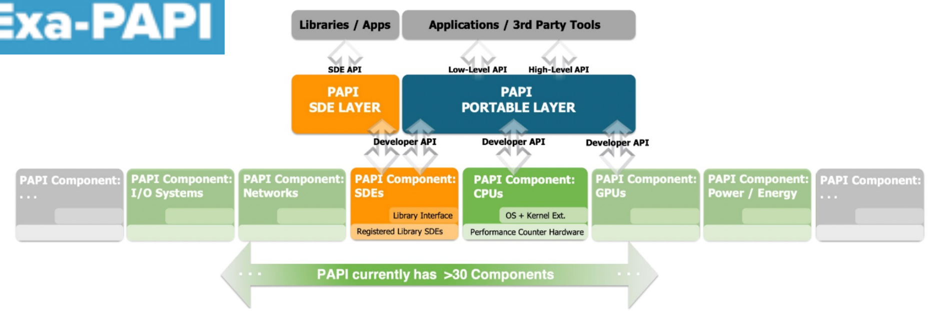


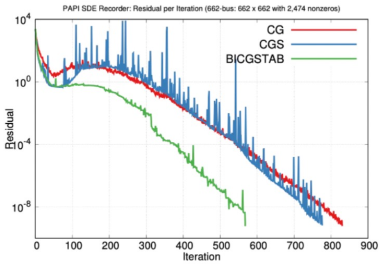



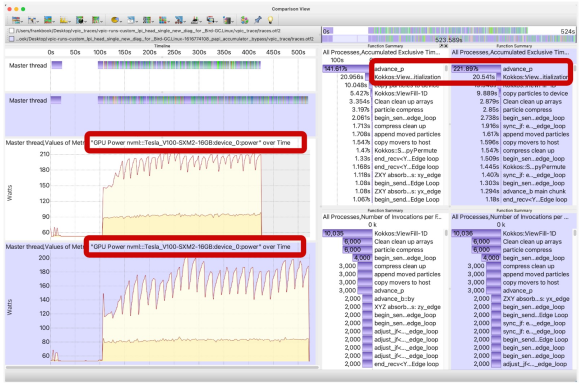

| | Functionality | OMP | CUDA | HIP |
|---|---|---|---|---|
| Basic | SpMV | ✓ | ✓ | ✓ |
| | SpMM | ✓ | ✓ | ✓ |
| | SpGeMM | ✓ | ✓ | ✓ |
| Krylov solvers | BiCG | ✓ | ✓ | ✓ |
| | BiCGSTAB | ✓ | ✓ | ✓ |
| | CG | ✓ | ✓ | ✓ |
| | CGS | ✓ | ✓ | ✓ |
| | GMRES | ✓ | ✓ | ✓ |
| | IDR | ✓ | ✓ | ✓ |
| Preconditioners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

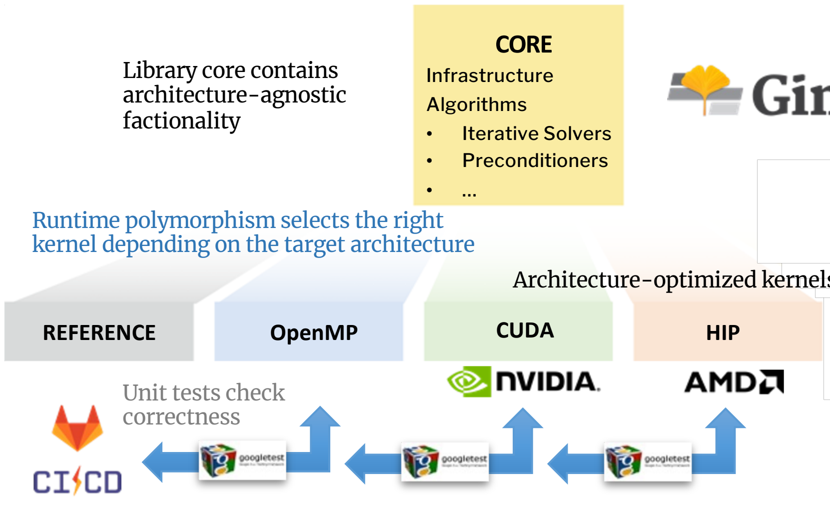
# The Design of an ECP Math Library

Exa-PAPI



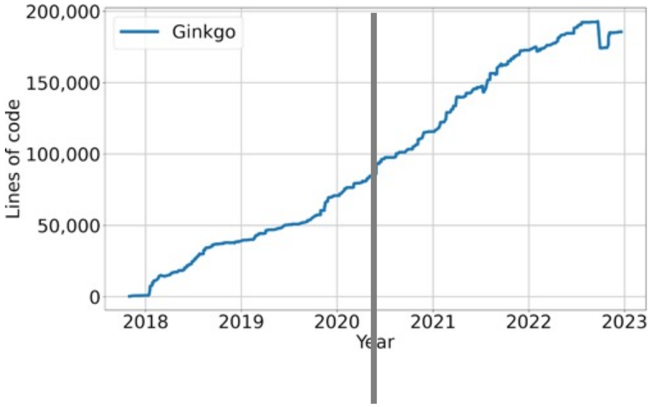





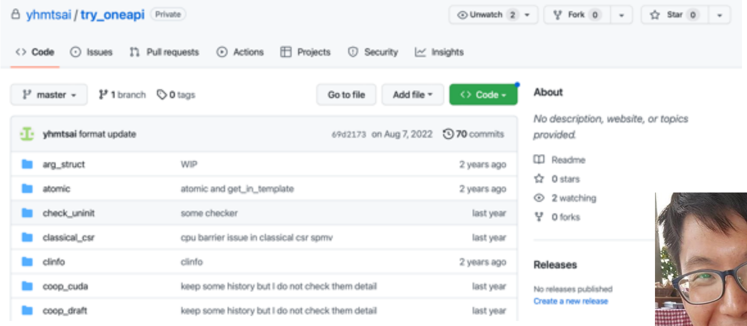

Ginkgo

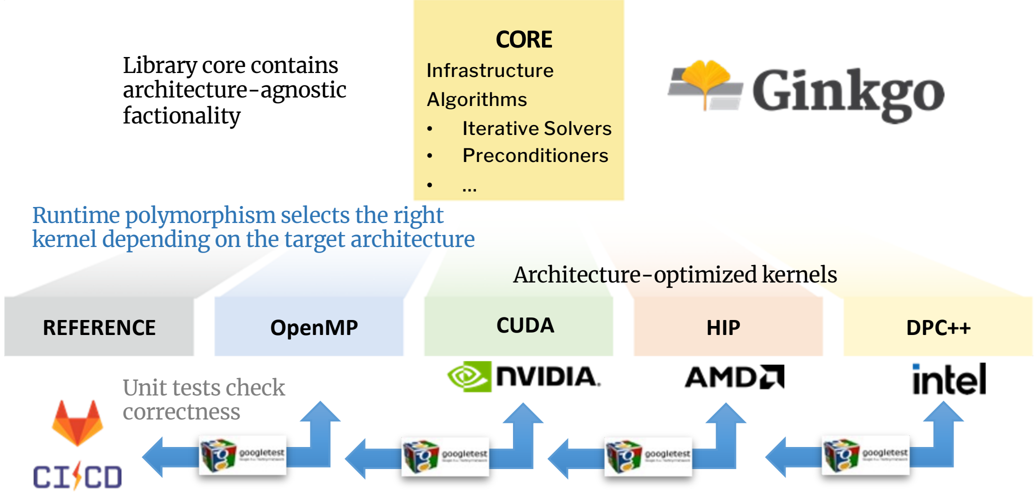
| Functionality | | OMP | CUDA | HIP |
|---|---|---|---|---|
| Basic | SpMV | ✓ | ✓ | ✓ |
| | SpMM | ✓ | ✓ | ✓ |
| | SpGeMM | ✓ | ✓ | ✓ |
| Krylov solvers | BiCG | ✓ | ✓ | ✓ |
| | BiCGSTAB | ✓ | ✓ | ✓ |
| | CG | ✓ | ✓ | ✓ |
| | CGS | ✓ | ✓ | ✓ |
| | GMRES | ✓ | ✓ | ✓ |
| | IDR | ✓ | ✓ | ✓ |
| Preconditioners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




Spack xSDK E4S

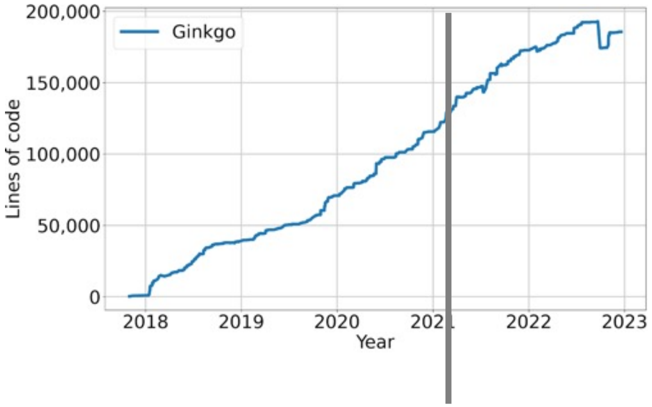
# The Design of an ECP Math Library


Ginkgo

<table>
<tr><th></th><th>Functionality</th><th>OMP</th><th>CUDA</th><th>HIP</th><th>DPC++</th></tr>
<tr><td>Basic</td><td>SpMV</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>SpMM</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>SpGeMM</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Krylov solvers</td><td>BiCG</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>BiCGSTAB</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>CG</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>CGS</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>GMRES</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>IDR</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Preconditioners</td><td>(Block-)Jacobi</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>ILU/IC</td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Parallel ILU/IC</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Parallel ILUT/ICT</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Sparse Approximate Inverse</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Utilities</td><td>On-Device Matrix Assembly</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>MC64/RCM reordering</td><td>✓</td><td></td><td></td><td></td></tr>
<tr><td></td><td>Wrapping user data</td><td colspan="4">✓</td></tr>
<tr><td></td><td>Logging</td><td colspan="4">✓</td></tr>
<tr><td></td><td>PAPI counters</td><td colspan="4">✓</td></tr>
</table>




Spack xSDK E4S Exa-PAPI oneAPI

Industry Collaboration with bi-weekly meetings

EXASCALE COMPUTING PROJECT

# The Design of an ECP Math Library

xSDK

Focus efforts
- Mixed precision
- batched

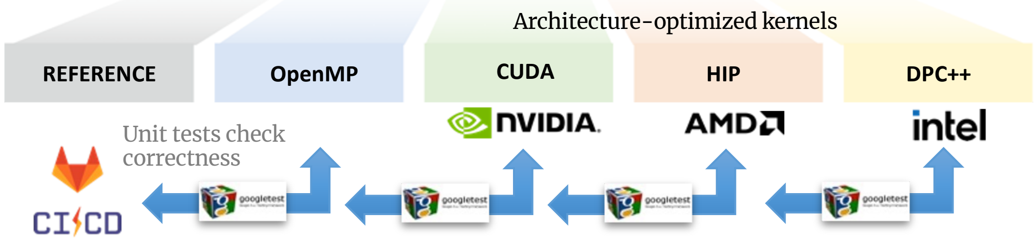
Library core contains architecture-agnostic factionality

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

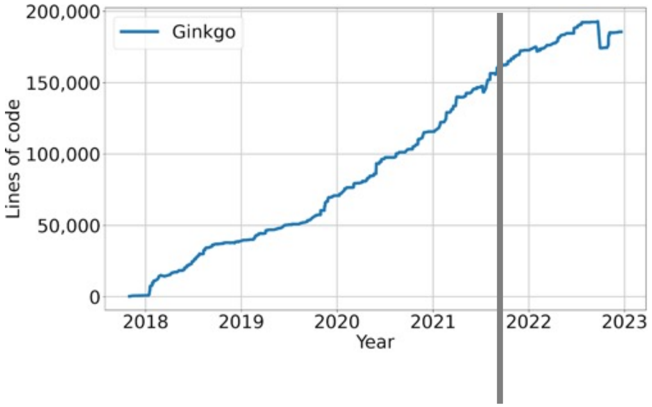
Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

Architecture-optimized kernels




| | 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

EXASCALE COMPUTING PROJECT

# The Design of an ECP Math Library

xSDK

Focus efforts
- 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 factionality

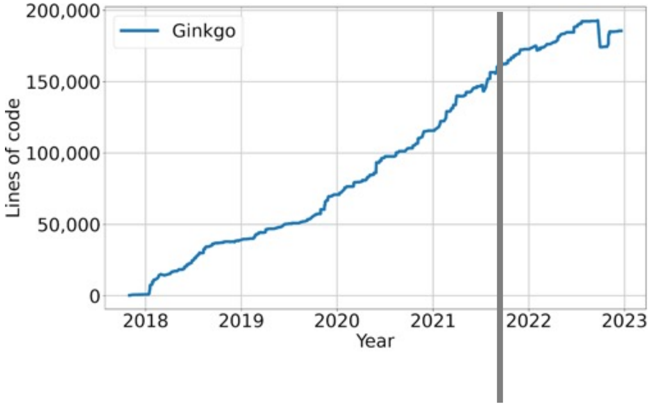
**CORE**
Infrastructure
Algorithms
- Iterative Solvers
- Preconditioners
- ...

Ginkgo

Runtime polymorphism selects the right kernel depending on the target architecture

Architecture-optimized kernels


| | 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 | ✓ (all) | | | |
| | Logging | ✓ (all) | | | |
| | PAPI counters | ✓ (all) | | | |




Spack
xSDK
E4S
Exa-PAPI
oneAPI

Industry Collaboration with bi-weekly meetings

xSDK

Focus efforts
- Mixed precision
- batched

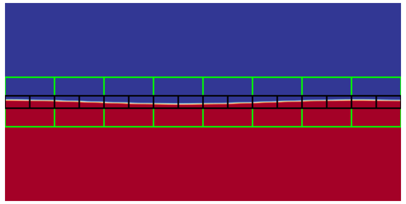
EXASCALE COMPUTING PROJECT

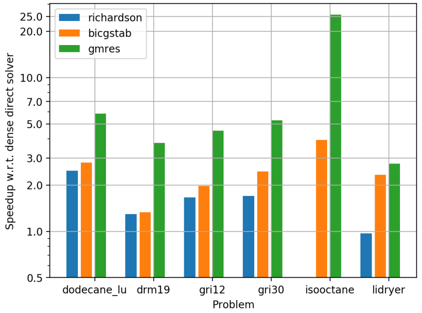
# The Design of an ECP Math Library

Focus efforts
- Mixed precision
- batched

**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%) |
| isooctane | 144 | 6,135 (30%) | 20,307 (98%) |
| lidryer | 10 | 91 (91%) | 91 (91%) |

**Batched Sparse Iterative Solvers for Computational Chemistry Simulations on GPUs**

Publisher: IEEE Cite This PDF

Isha Aggarwal ; Aditya Kashi ; Pratik Nayak ; Cody J. Balos ; Carol S. Woodward ; Hartwig Anzt All Authors

Library core contains architecture-agnostic factionality

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

Ginkgo

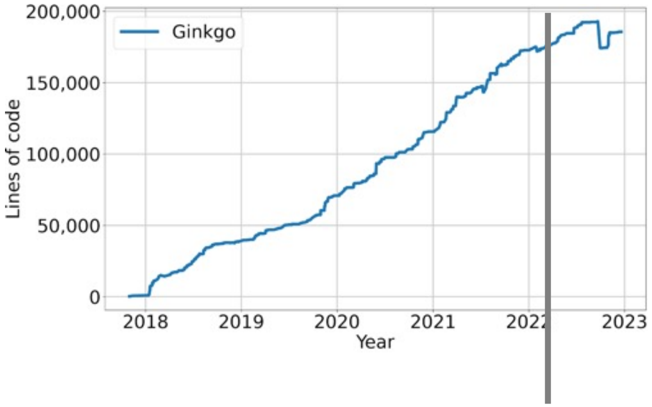
Runtime polymorphism selects the right kernel depending on the target architecture

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

sundials

<table>
<tr><th></th><th>Functionality</th><th>OMP</th><th>CUDA</th><th>HIP</th><th>DPC++</th></tr>
<tr><td>Basic</td><td>SpMV</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>SpMM</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>SpGeMM</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Krylov solvers</td><td>BiCG</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>BiCGSTAB</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>CG</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>CGS</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>GMRES</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>IDR</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Preconditioners</td><td>(Block-)Jacobi</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>ILU/IC</td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Parallel ILU/IC</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Parallel ILUT/ICT</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Sparse Approximate Inverse</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Batched</td><td>Batched BiCGSTAB</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td></td><td>Batched CG</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td></td><td>Batched GMRES</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td></td><td>Batched ILU</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td></td><td>Batched ISAI</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td></td><td>Batched Jacobi</td><td>✓</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td>AMG</td><td>AMG preconditioner</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>AMG solver</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>Parallel Graph Match</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Utilities</td><td>On-Device Matrix Assembly</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td></td><td>MC64/RCM reordering</td><td>✓</td><td></td><td></td><td></td></tr>
<tr><td></td><td>Wrapping user data</td><td colspan="4">✓</td></tr>
<tr><td></td><td>Logging</td><td colspan="4">✓</td></tr>
<tr><td></td><td>PAPI counters</td><td colspan="4">✓</td></tr>
</table>

# The Design of an ECP Math Library

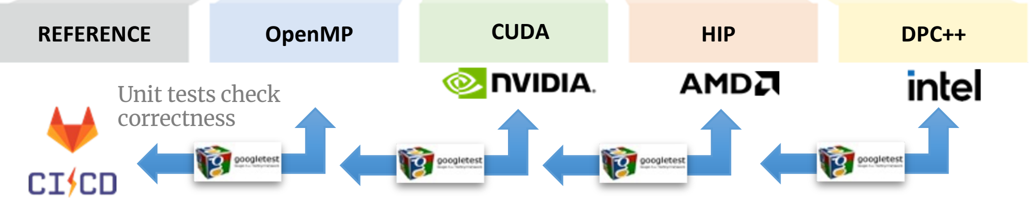
Focus efforts
- Mixed precision
- batched


Library core contains architecture-agnostic factionality

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

Runtime polymorphism selects the right kernel depending on the target architecture

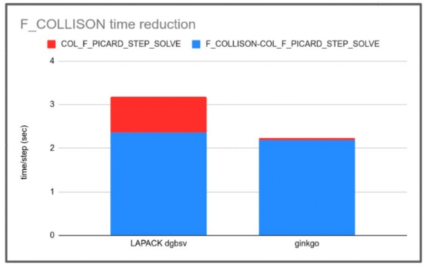
Architecture-optimized kernels




## 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 ptl/GPU, 89.6M ptl/node (ptl_num=700k)
  - Ran 20 time steps, collisions calculated every other time step

| | per time step (s) | |
|---|---|---|
| | dgbsv | ginkgo |
| MAIN_LOOP | 19.05 | 18.26 |
| MAIN_LOOP-F_COLLISION | 15.87 | 16.03 |
| F_COLLISON | 3.18 | 2.23 |
| F_COLLISON-COL_F_PICARD_STEP_SOLVE | 2.37 | 2.18 |
| COL_F_PICARD_STEP_SOLVE | 0.82 | 0.05 |
| COL_F_SOLVER_CONVERT_BANDED | 0.08 | |
| COL_F_SOLVER_DGBSV | 0.53 | |

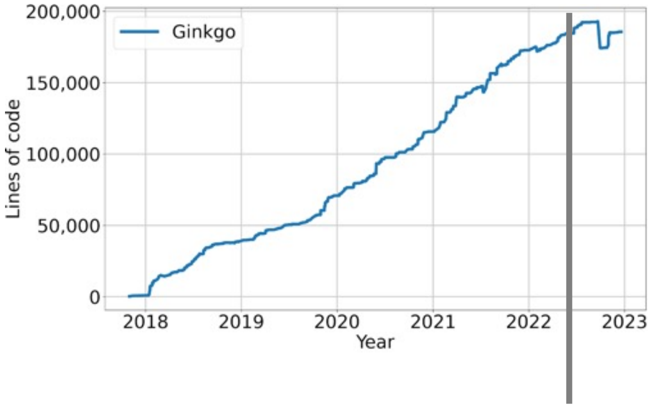



- With CPU LAPACK dgbsv
  - F_COLLISON is 17% of MAIN_LOOP time
  - COL_F_PICARD_STEP_SOLVE is 24% of F_COLLISON time and 4.3% of MAIN_LOOP time
  - COL_F_SOLVER_DGBSV is 66% of COL_F_PICARD_STEP_SOLVE
  - COL_F_SOLVER_CONVERT_BANDED is 10% of COL_F_PICARD_STEP_SOLVE
- Replacing CPU LAPACK dgbsv by GPU Ginkgo
  - COL_F_PICARD_STEP_SOLVE reduced from 0.82s to 0.046s per step; reduction of 94%
  - F_COLLISON reduced from 3.18s to 2.23s per step; reduction of 30%
  - MAIN_LOOP time reduced by 4.1%

Velocity grid: 33x39; matrices: 1287 rows

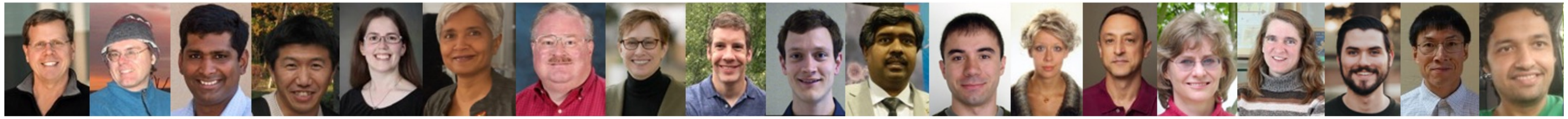
XGC collision operator solve performed on GPU

© Doug Kothe

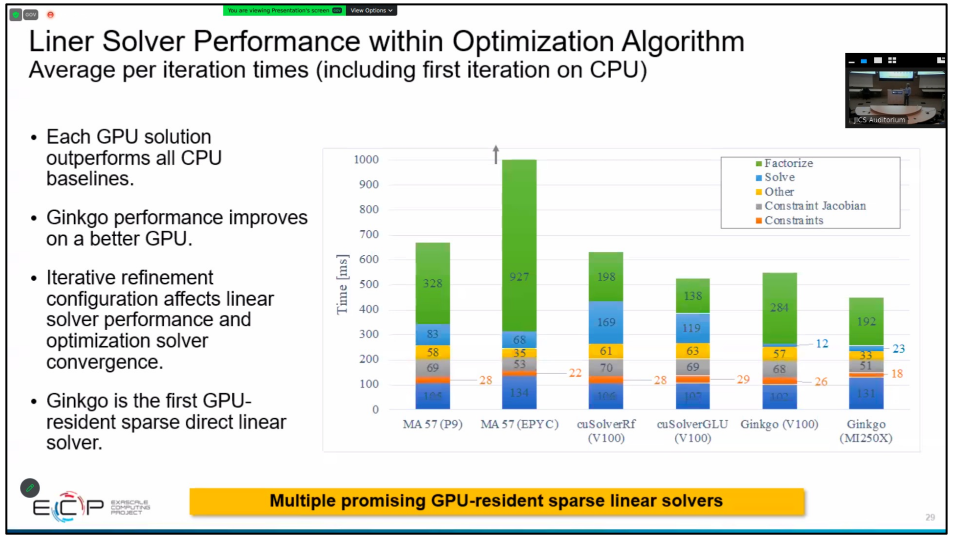



Industry Collaboration with bi-weekly meetings

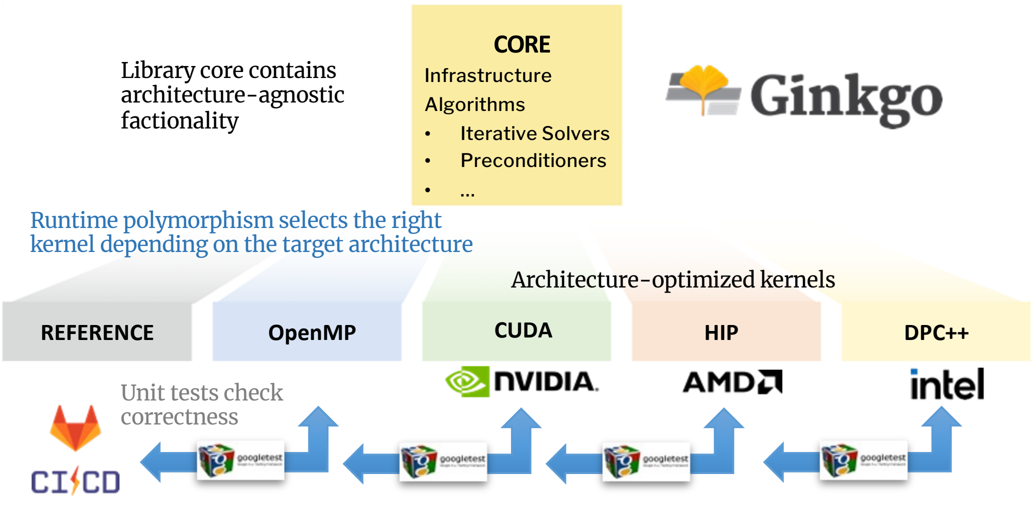
| | Functionality | OMP | CUDA | HIP | DPC++ |
|---|---|---|---|---|---|
| Basic | SpMV | ✓ | ✓ | ✓ | ✓ |
| | SpMM | ✓ | ✓ | ✓ | ✓ |
| | SpGeMM | ✓ | ✓ | ✓ | ✓ |
| Krylov solvers | BiCG | ✓ | ✓ | ✓ | ✓ |
| | BiCGSTAB | ✓ | ✓ | ✓ | ✓ |
| | CG | ✓ | ✓ | ✓ | ✓ |
| | CGS | ✓ | ✓ | ✓ | ✓ |
| | GMRES | ✓ |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

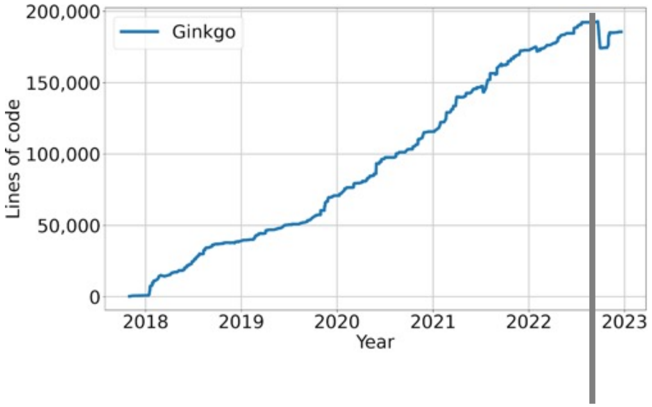


# The Design of an ECP Math Library




© Slaven Peles





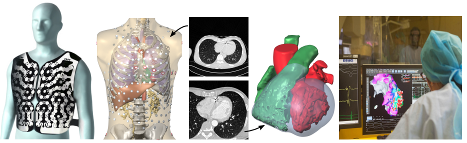
Industry Collaboration with bi-weekly meetings

| | 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 | ✓ | ✓ | ✓ | ✓ |
| 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 | ✓ | | | |

EXASCALE COMPUTING PROJECT

# “Now” – Near completion of ECP

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

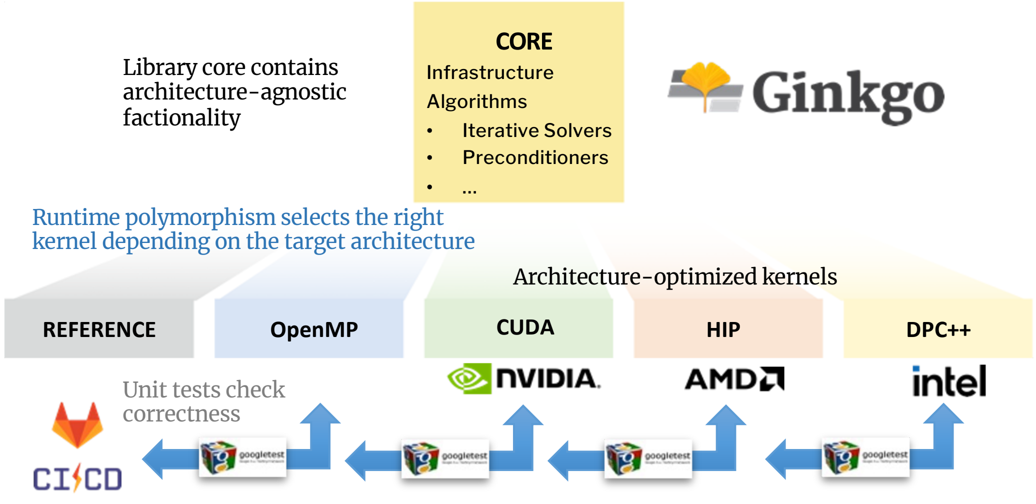


https://www.microcard.eu

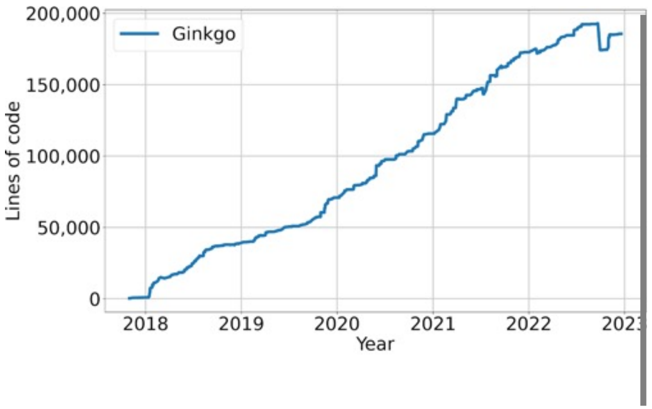
- BMBF PDExa project uses Ginkgo

- BMBF ExaSIM project uses Ginkgo

https://exasim-project.com




| | 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 | ✓ (all) | | | |
| | Logging | ✓ (all) | | | |
| | PAPI counters | ✓ (all) | | | |




Industry Collaboration with bi-weekly meetings

# 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.**