Everything Everywhere All at Once - How Improving Software Sustainability, Productivity, and Performance Naturally Go Hand in Hand

A story from the trenches of the US Exascale Computing Project

Prof. Dr. Hartwig Anzt

Technical University of Munich

05/23/2024

The US Exascale Computing Project

Advancing Scientific Discovery

The ECP aims to ensure availability of the exascale computing ecosystem necessary for developing clean energy systems, improving the resilience of our infrastructure, designing new materials that can perform in extreme environments, adapting to changes in the water cycle, developing smaller and more powerful accelerators for use in medicine and industry, and much more. Several projects focus on data-intensive problems to enable effective use of the data streams from powerful scientific facilities, complex environmental genomes, and cancer research (patient genetics, tumor genomes, molecular simulations, and clinical data).

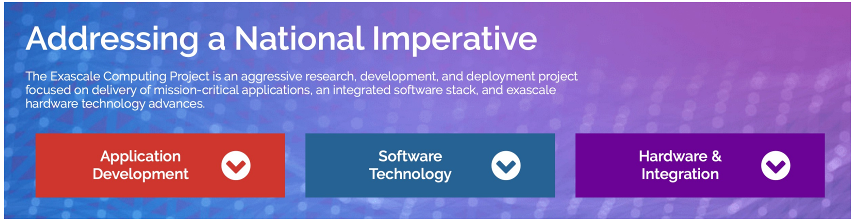
Strengthening National Security

The ECP teams are also developing new applications for supporting the NNSA Stockpile Stewardship Program, which is responsible for maintaining the readiness and reliability of our nuclear weapons systems—without underground testing. Assessing the performance of weapons systems subject to hostile environments and potential threat scenarios exceeds the capabilities of current HPC systems and codes. NNSA application projects are focused on providing the sophisticated modeling and analysis tools needed to sustain the U.S. nuclear deterrence.

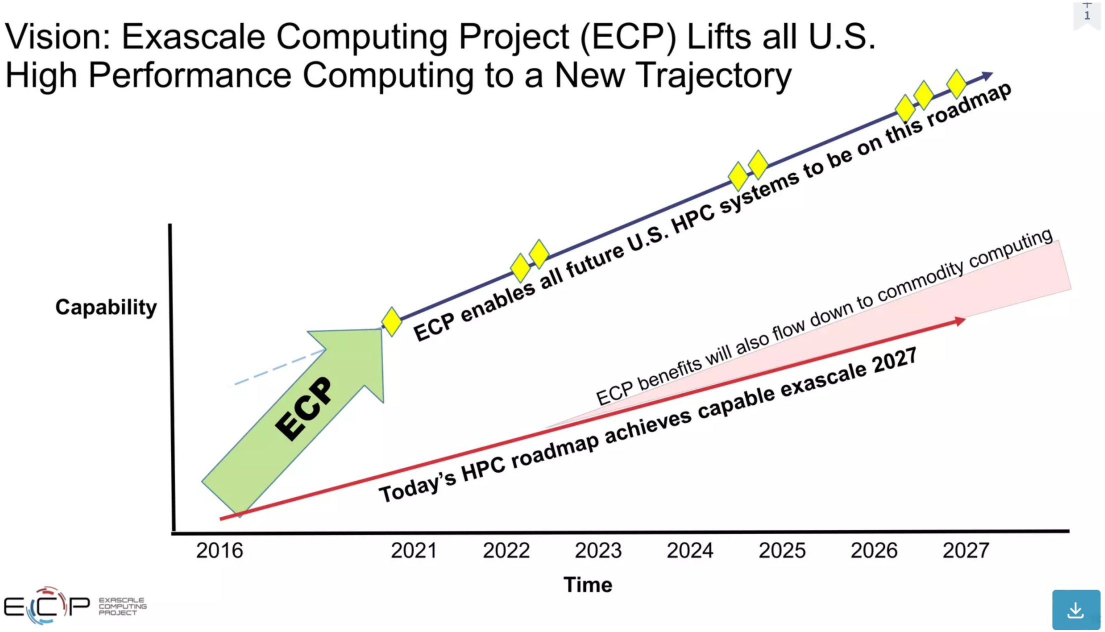
Improving Industrial Competitiveness

Exascale systems will be used to accelerate research that leads to innovative products and speeds commercialization, creating jobs and driving US competitiveness across industrial sectors, such as the emerging energy economy. To ensure alignment with US industry needs, the ECP is engaging senior technology decision makers from among the country's most prominent private sector companies.

The US Exascale Computing Project



© Paul Messina



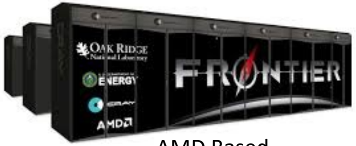
The US Exascale Computing Project

EXASCALE COMPUTING PROJECT

US$4B – what is it spent on?

- 3 computers
  - $600M each
  - $400M to vendors for Design, Path, Fast - Forward

  21 Applications



AMD Based
(Up & running)
20 MW

Intel Based
(Up & running)
40 MW

AMD APU Based
(panned)

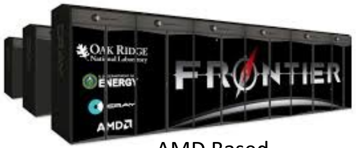
The US Exascale Computing Project

EXASCALE COMPUTING PROJECT

US$4B – what is it spent on?

- 3 computers
  - $600M each
  - $400M to vendors for Design, Path, Fast - Forward

  21 Applications



AMD Based
(Up & running)
20 MW

Intel Based
(Up & running)
40 MW

AMD APU Based
(panned)

Domain*	Base Challenge Problem
Wind Energy	2x2 5 MW turbine array in 3x3x1 km^3 domain
Nuclear Energy	Small Modular Reactor with complete in-vessel coolant loop
Fossil Energy	Burn fossil fuels cleanly with CLRs
Combustion	Reactivity controlled compression ignition
Accelerator Design	TeV-class 10^{2-3} times cheaper & smaller
Magnetic Fusion	Coupled gyrokinetics for ITER in H-mode
Nuclear Physics: QCD	Use correct light quark masses for first principles light nuclei properties
Chemistry: GAMESS	Heterogeneous catalysis: MSN reactions
Chemistry: NWChemEx	Catalytic conversion of biomass
Extreme Materials	Microstructure evolution in nuclear matls
Additive Manufacturing	Born-qualified 3D printed metal alloys

Domain*	Challenge Problem
Quantum Materials	Predict & control matls @ quantum level
Astrophysics	Supernovae explosions, neutron star mergers
Cosmology	Extract "dark sector" physics from upcoming cosmological surveys
Earthquakes	Regional hazard and risk assessment
Geoscience	Well-scale fracture propagation in wellbore cement due to attack of CO_2-saturated fluid
Earth System	Assess regional impacts of climate change on the water cycle @ 5 SYPD
Power Grid	Large-scale planning under uncertainty; underfrequency response
Cancer Research	Scalable machine learning for predictive preclinical models and targeted therapy
Metagenomics	Discover and characterize microbial communities through genomic and proteomic analysis
FEL Light Source	Protein and molecular structure determination using streaming light source data

Sustainable software development

PMR Core (17)	Compilers and Support (7)	Tools and Technology (11)	xSDK (16)	Visualization Analysis and Reduction (9)	Data mgmt, I/O Services, Checkpoint restart (12)	Ecosystem/E4S at-large (12)
QUO	openarc	TAU	hypre	ParaView	SCR	mpiFileUtils
Papyrus	Kitsune	HPCToolkit	FleSCI	Catalyst	FAODEL	TriBITS
SICM	LLVM	Dyninst Binary Tools	MFEM	VTK-m	ROMIO	MarFS
Legion	CHiLL autotuning comp	Gotcha	Kokkoskernels	SZ	Mercury (Mochi suite)	GUFI
Kokkos (support)	LLVM openMP comp	Caliper	Trilinos	zfp	HDF5	Intel GEOPM
RAJA	OpenMP V & V	PAPI	SUNDIALS	VisIt	Parallel netCDF	BEE
CHAI	Flang/LLVM Fortran comp	Program Database Toolkit	PETSc/TAO	ASCENT	ADIOS	FSEFI
PaRSEC*		Search (random forests)	libEnsemble	Cinema	Darshan	Kitten Lightweight Kernel
DARMA		Siboka	STRUMPACK	ROVER	UnifyCR	COOLR
GASNet-EX		C2C	SuperLU		VeloC	NRM
Qthreads		Sonar	ForTrilinos		IOSS	ArgoContainers
BOLT			SLATE		HXHIM	Spack
UPC++			MAGMA			
MPICH			DTK			
Open MPI			Tasmanian			
Umpire			Ginkgo			
AML						

Legend:
- PMR
- Tools
- Math Libraries
- Data and Vis
- Ecosystems and delivery

A few words about myself

- Born and raised in Karlsruhe
- PhD in Numerical Mathematics from KIT
- Focus on computational linear algebra and high performance computing (HPC)
- Linear solvers, preconditioners, …
- During my PostDoc at the University of Tennessee, I developed MAGMA sparse

MAGMA SPARSE

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

MATLAB

Limitations:

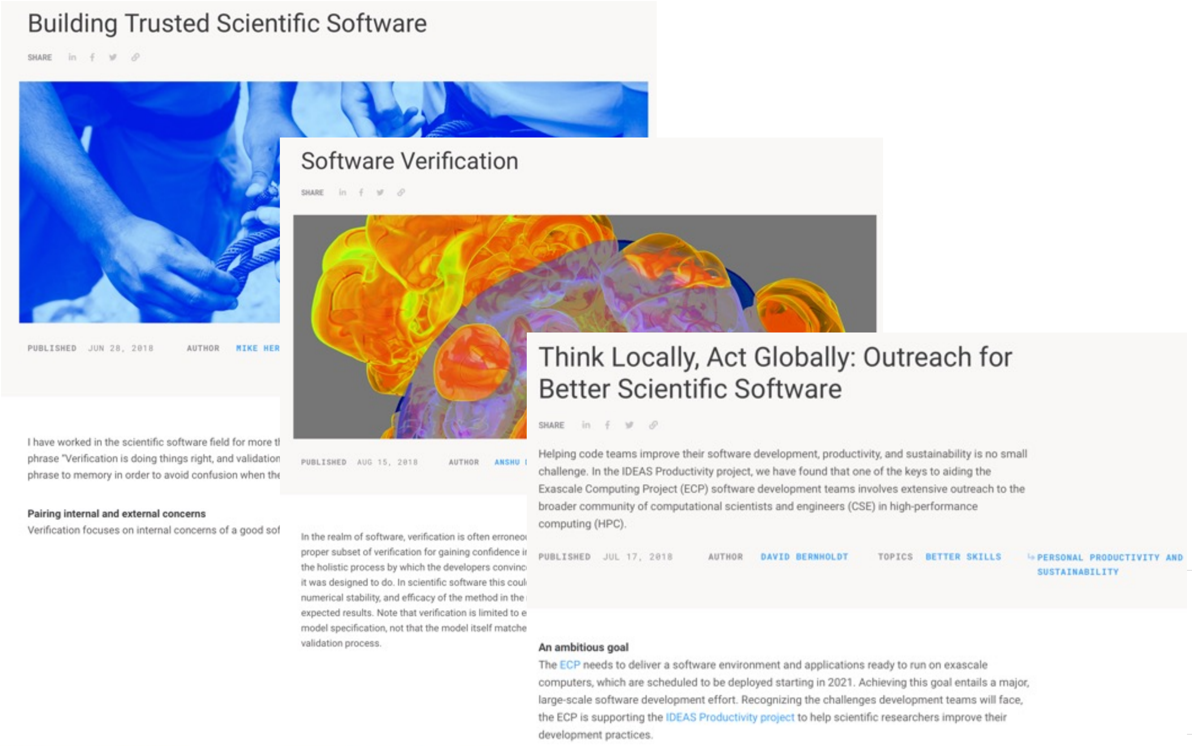
- *C code with hand-written build system*
- *Sparse unit testing*
- *Focus on NVIDIA GPUs*
- *Design-specific limitations (flexibility/extensibility)*

Designing an ECP math library

IDEAS productivity

better scientific software

TRILINOS


Building Trusted Scientific Software

SHARE

PUBLISHED JUN 28, 2018 AUTHOR MIKE HER

I have worked in the scientific software field for more t
phrase "Verification is doing things right, and validation
phrase to memory in order to avoid confusion when the

Pairing internal and external concerns

Verification focuses on internal concerns of a good sof

Software Verification

SHARE

PUBLISHED AUG 15, 2018 AUTHOR ANSHU I

In the realm of software, verification is often erroneo
proper subset of verification for gaining confidence i
the holistic process by which the developers convinc
it was designed to do. In scientific software this coul
numerical stability, and efficacy of the method in the
expected results. Note that verification is limited to e
model specification, not that the model itself matche
validation process.

Think Locally, Act Globally: Outreach for Better Scientific Software

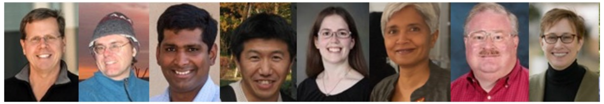
SHARE

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).

PUBLISHED JUL 17, 2018 AUTHOR DAVID BERNHOLDT TOPICS BETTER SKILLS PERSONAL PRODUCTIVITY AND SUSTAINABILITY

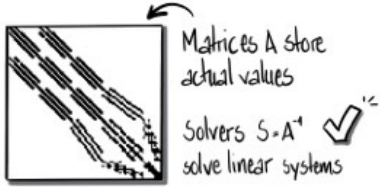
An ambitious goal

The ECP needs to deliver a software environment and applications ready to run on exascale computers, which are scheduled to be deployed starting in 2021. Achieving this goal entails a major, large-scale software development effort. Recognizing the challenges development teams will face, the ECP is supporting the IDEAS Productivity project to help scientific researchers improve their development practices.

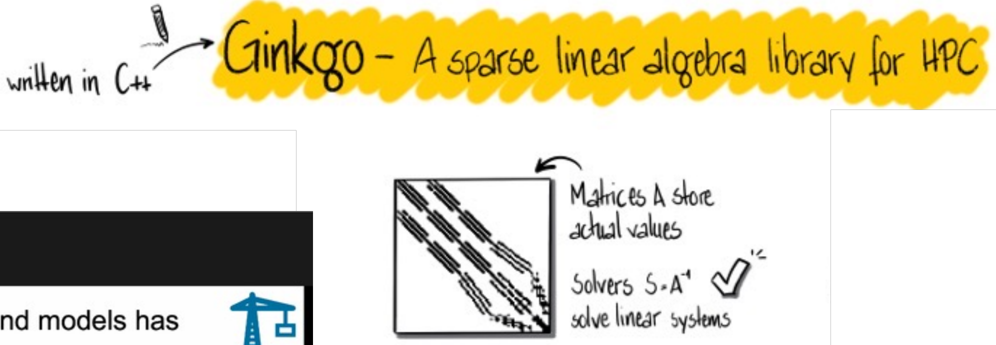


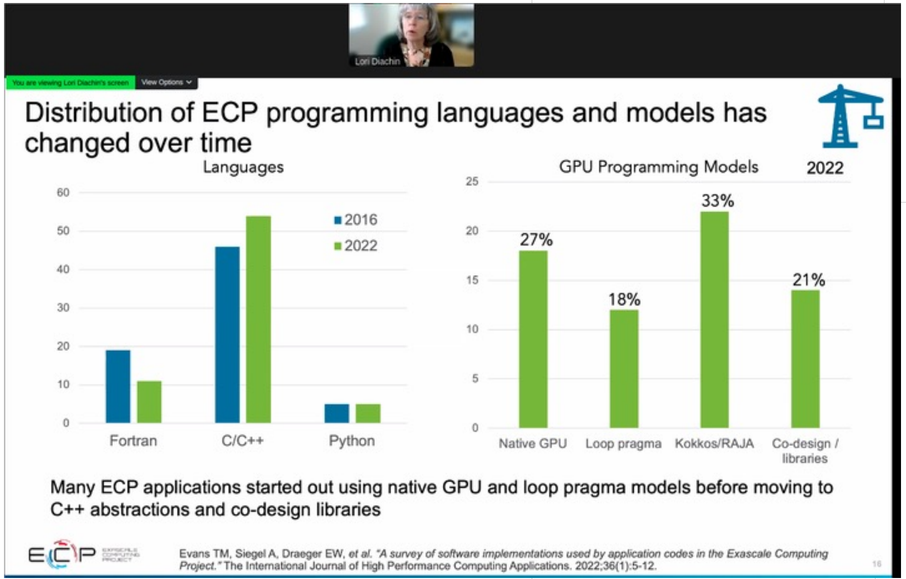
Designing software for performance, portability, & sustainability

Ginkgo – A sparse linear algebra library for HPC



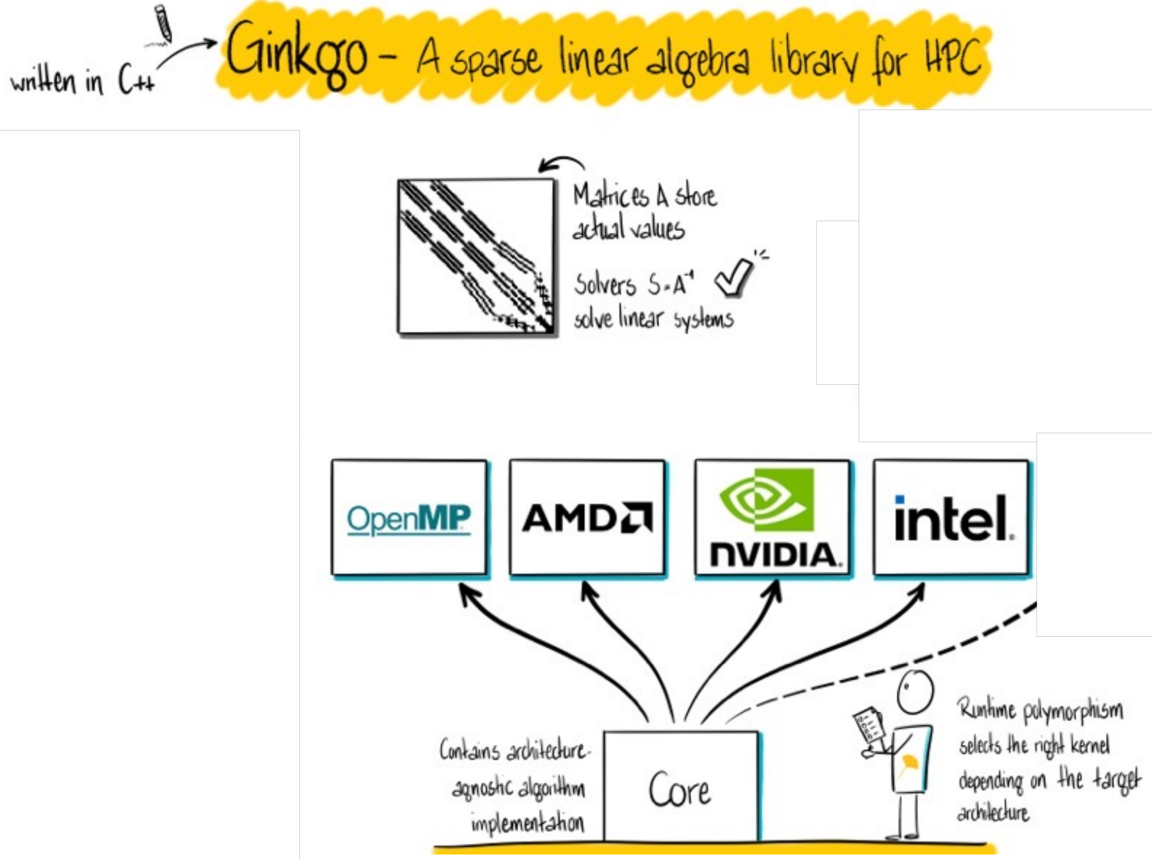
Designing software for performance, portability, & sustainability



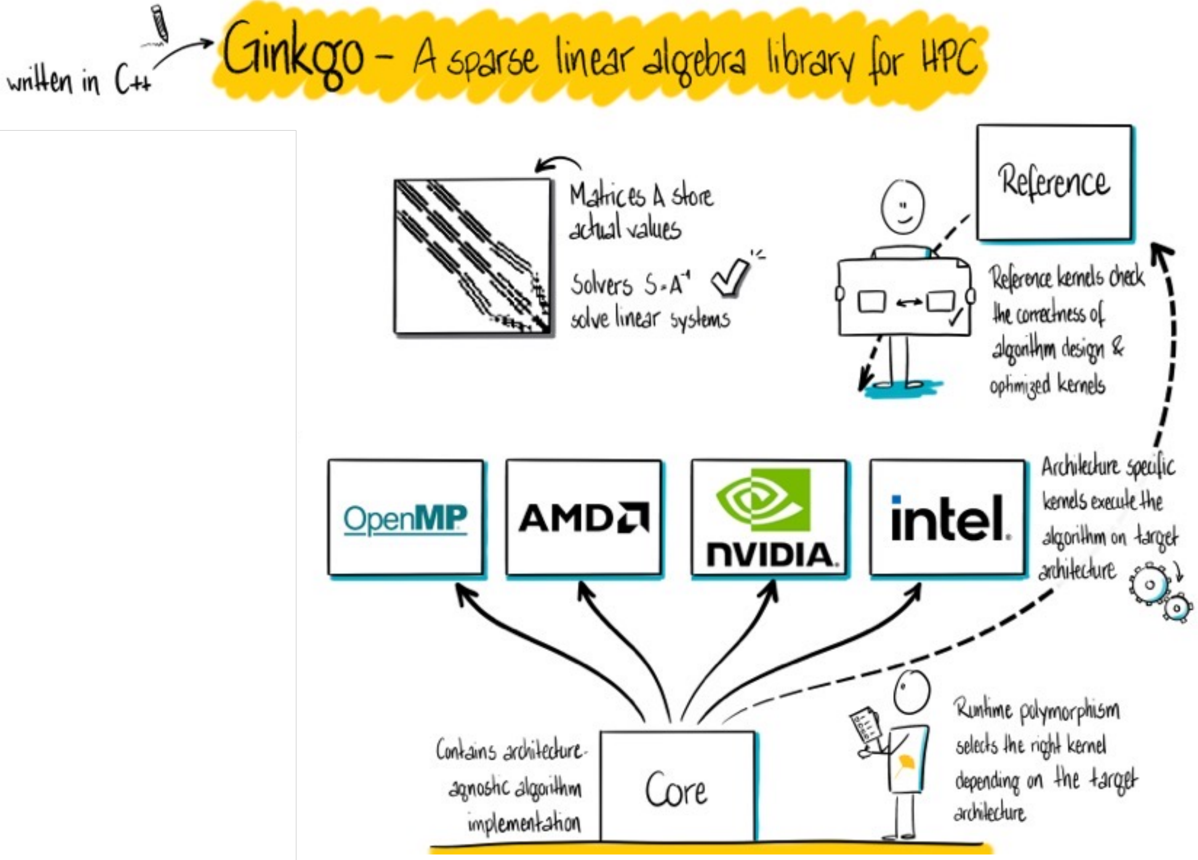


Lori Daichin, 05/22/2024

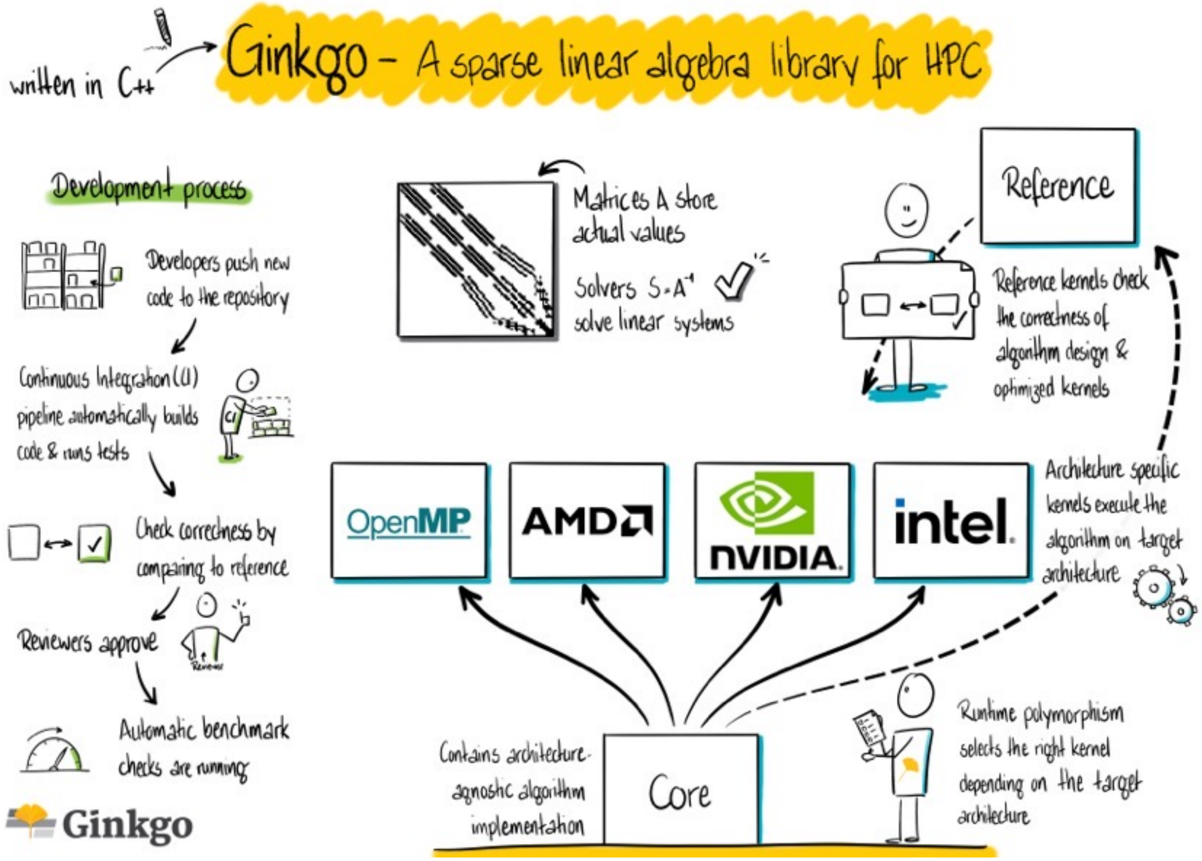
Designing software for performance, portability, & sustainability



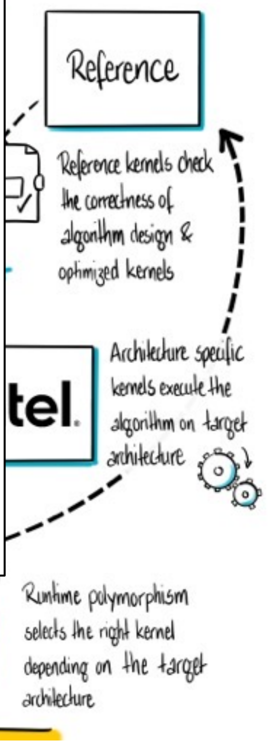
Designing software for performance, portability, & sustainability



Designing software for performance, portability, & sustainability



Designing software for performance, portability, & sustainability



Linear Operator Interface

We express everything as Linear Operator.

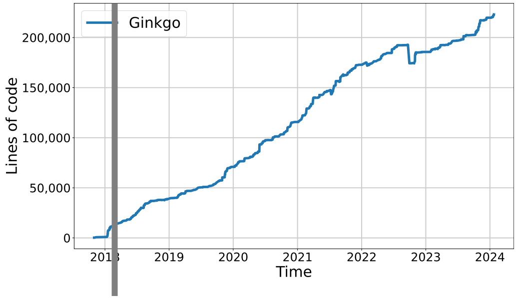
- *Internally, we leverage C++ class inheritance.*
- *Applications can apply any functionality as a linear operator.*

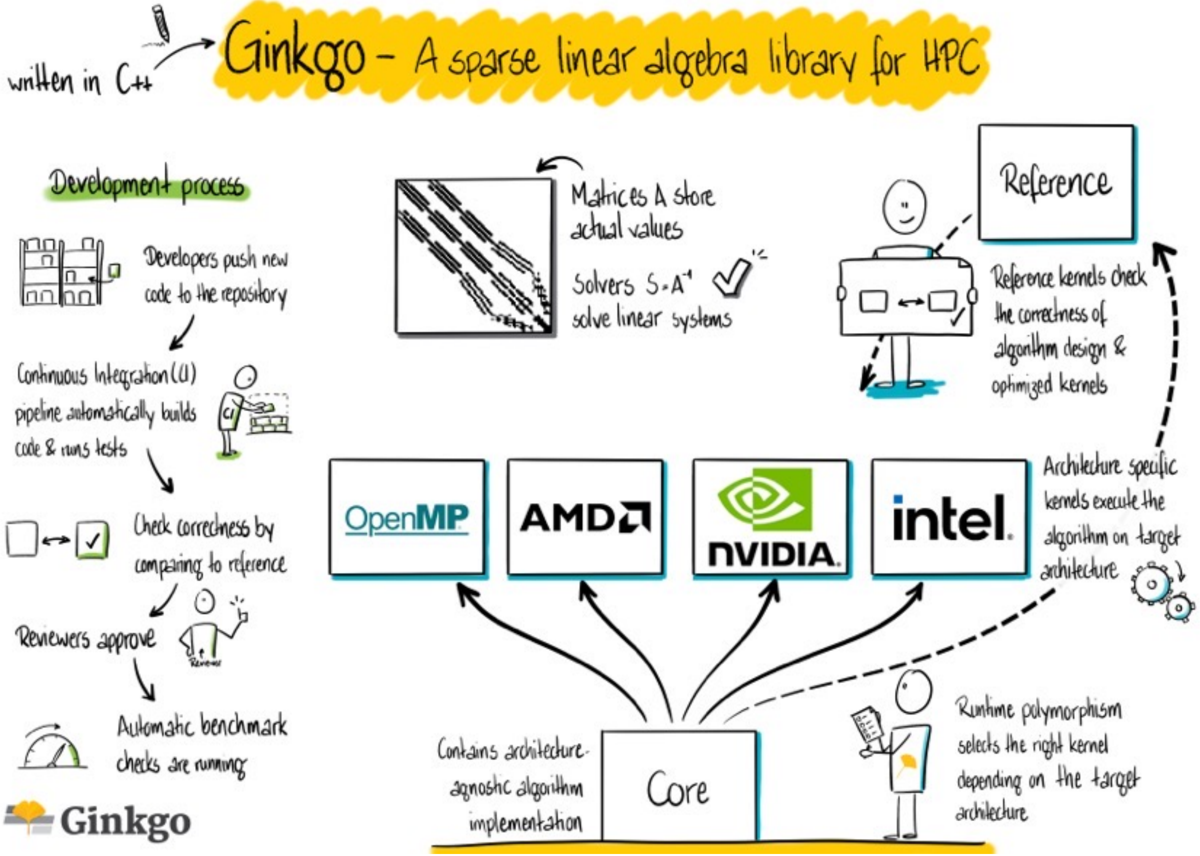
Matrix-Vector Product	Preconditioner (for matrix A)	Solver (for system $Ax = b$)
$x := A \cdot b$	$x := M^{-1} \cdot b$	$x := S \cdot b$
	$M^{-1} \approx A^{-1}$	$S \approx A^{-1}$
	$M^{-1} = \Pi(A)$	$S = \Sigma(A)$

All of them can be expressed as

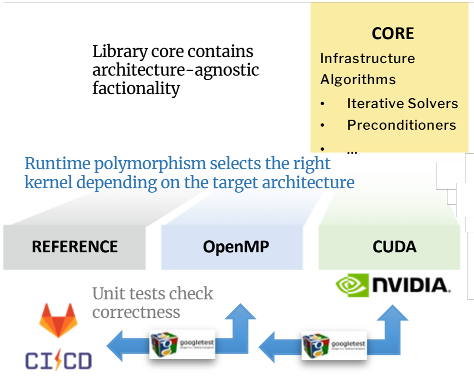
Application of a linear operator* (LinOp) $L : \mathbb{F}^m \rightarrow \mathbb{F}^m$

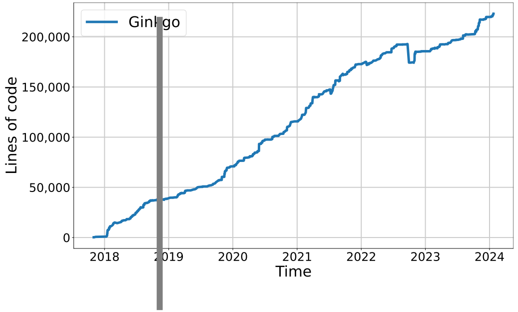
Designing software for performance, portability, & sustainability





Starting with the CUDA backend

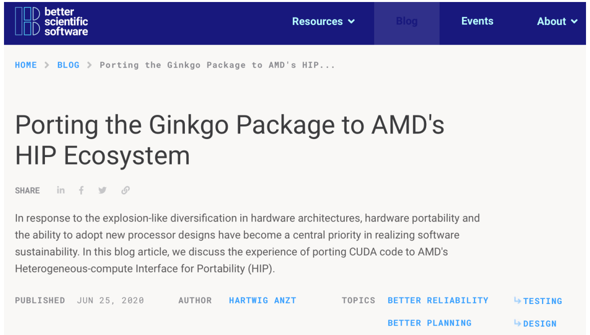




	Functionality	OMP	CUDA
Basic	SpMV	✓	✓
	SpMM	✓	✓
	SpGeMM	✓	✓
Krylov solvers	BiCG	✓	✓
	BiCGSTAB	✓	✓
	CG	✓	✓
	CGS	✓	✓
	GMRES	✓	✓
	IDR	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓
	ILU/IC		✓
	Parallel ILU/IC	✓	✓
	Parallel ILUT/ICT	✓	✓
	Sparse Approximate Inverse	✓	✓

Extending to AMD GPUs

~2 months


better scientific software

Resources Blog Events About

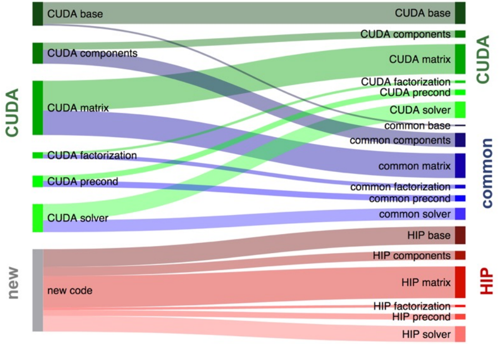
HOME > BLOG > Porting the Ginkgo Package to AMD's HIP...

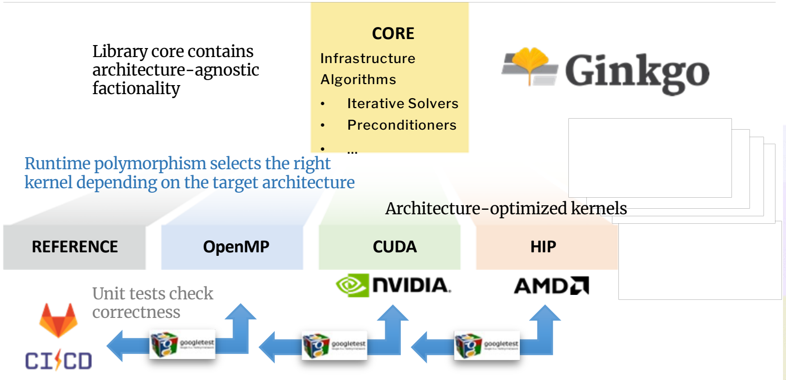
Porting the Ginkgo Package to AMD's HIP Ecosystem

SHARE

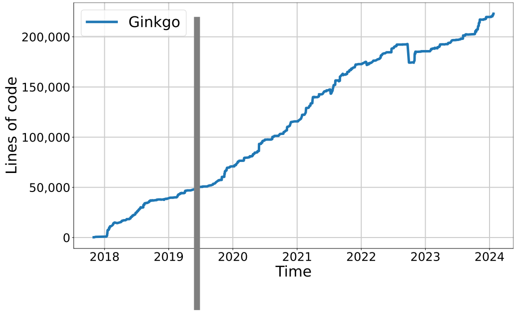
In response to the explosion-like diversification in hardware architectures, hardware portability and the ability to adopt new processor designs have become a central priority in realizing software sustainability. In this blog article, we discuss the experience of porting CUDA code to AMD's Heterogeneous-compute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

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

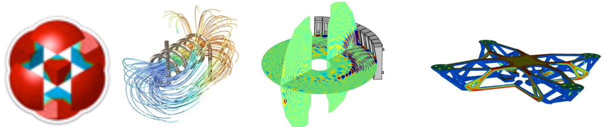




	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓



Input from the “first customer”



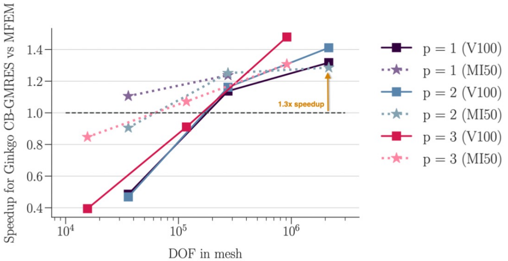
MFEM is a *free*, *lightweight*, *scalable* C++ library for finite element methods.

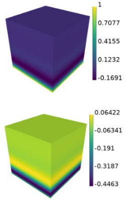
Speeding up MFEM’s “example 22” on GPUs

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:

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

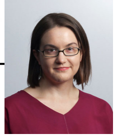
with a = 1, b = 1, ω = 10, c = 20

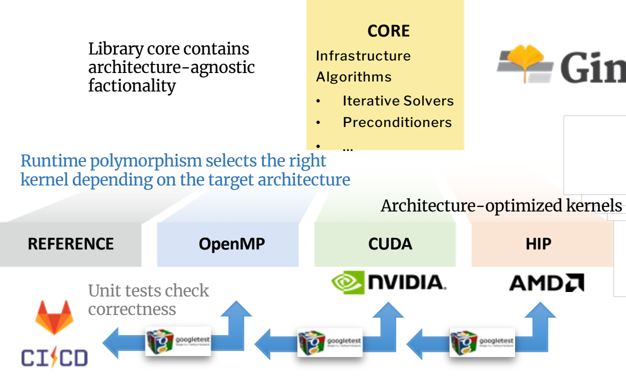


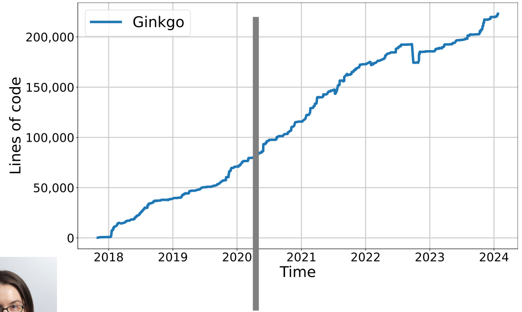


Real part of solution (top), imaginary part of solution

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.



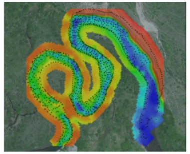




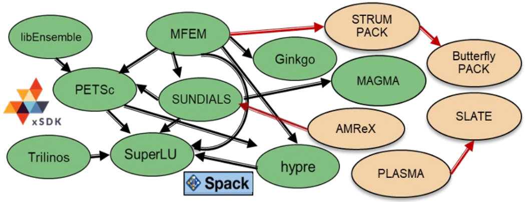
	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
Utilities	On-Device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

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.

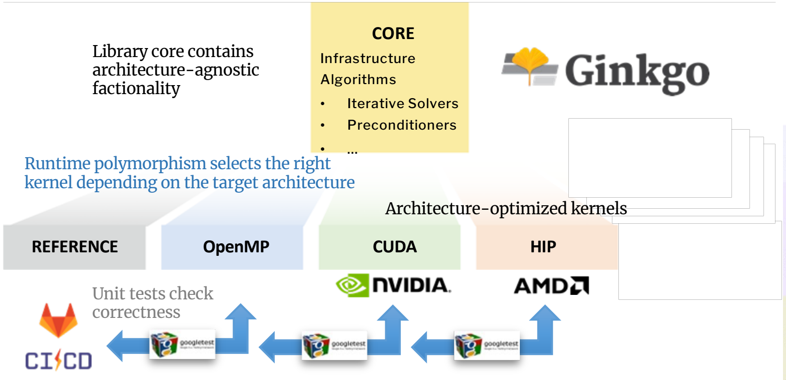


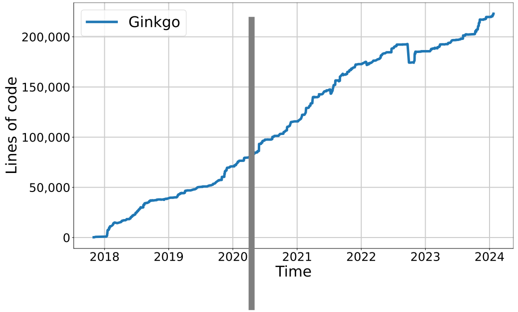
xsdk-examples v.0.3.0

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/

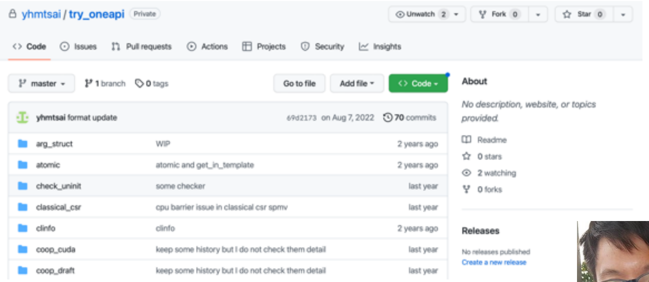


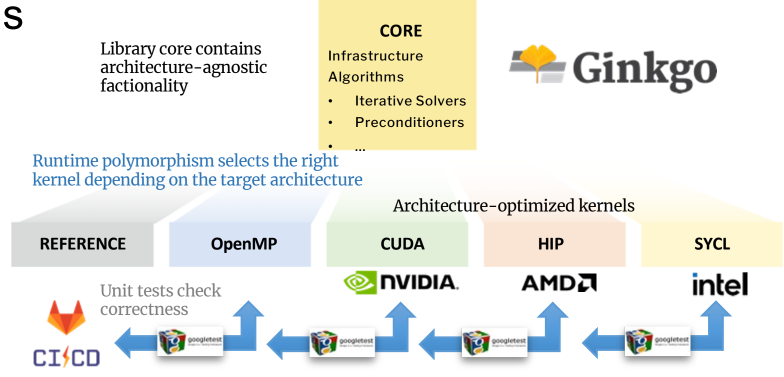


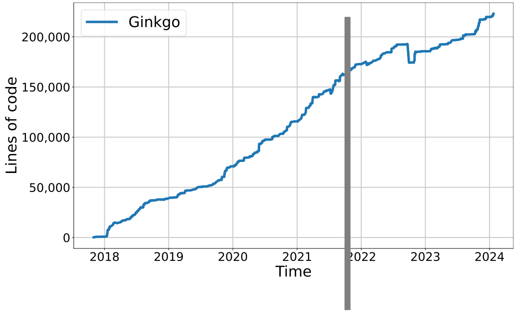
	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
Utilities	On-Device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

Extending to Intel GPUs

~12 months



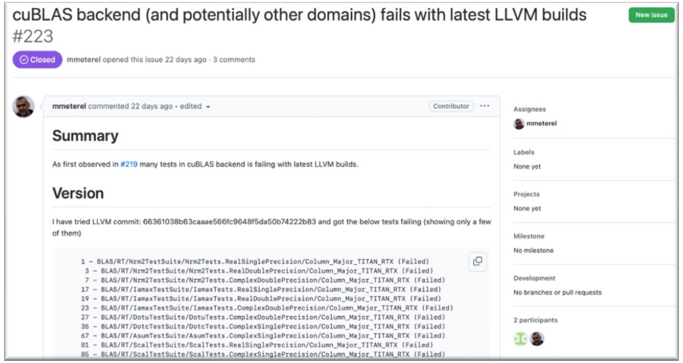




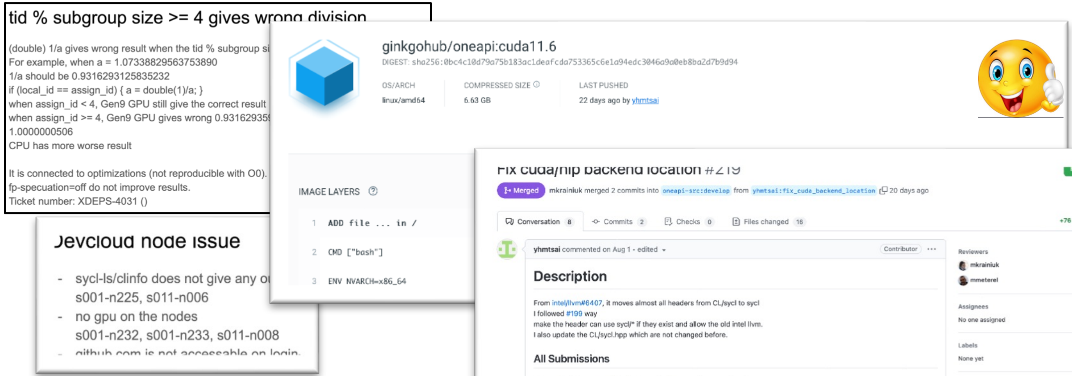
	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate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

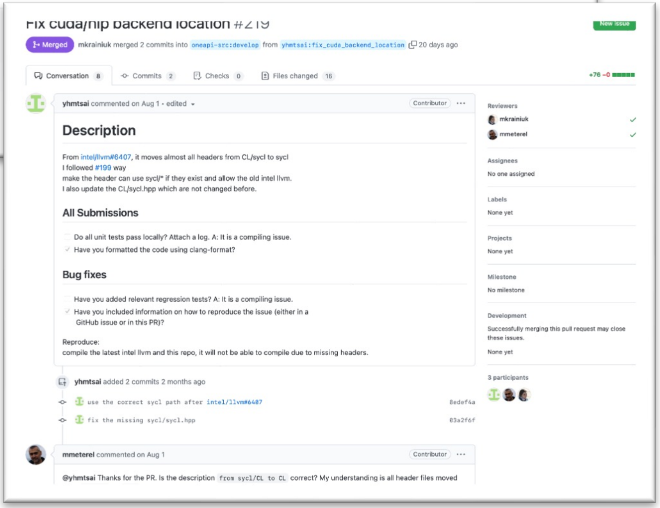
Extending to Intel GPUs

- 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!

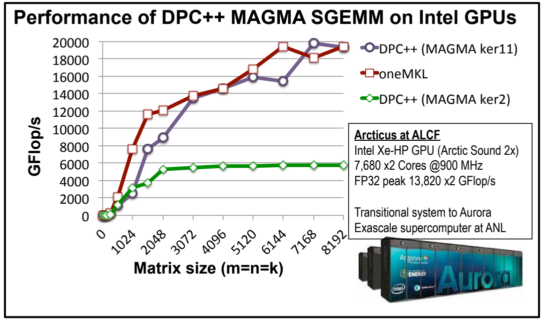




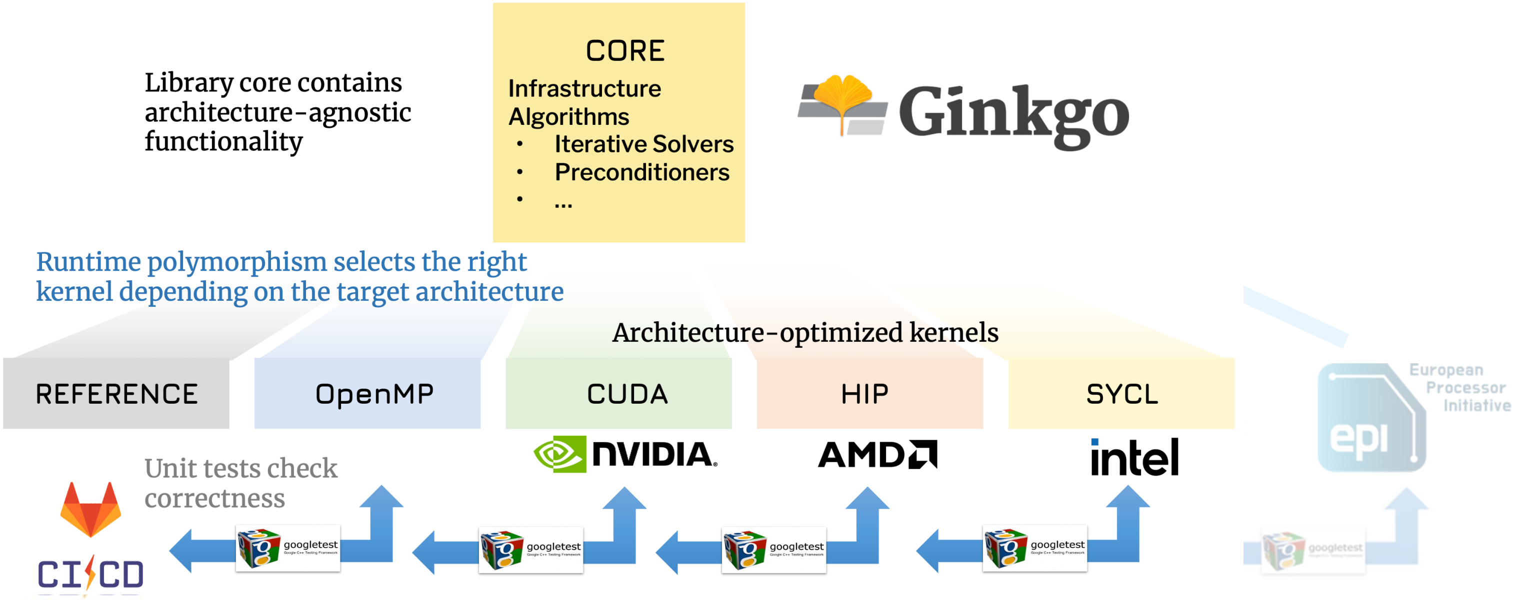
From DPCPP AoT documentation, not clear:

- 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++/L0) - 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



Portability as central design principle



This software design gives portability, performance, and sustainability.

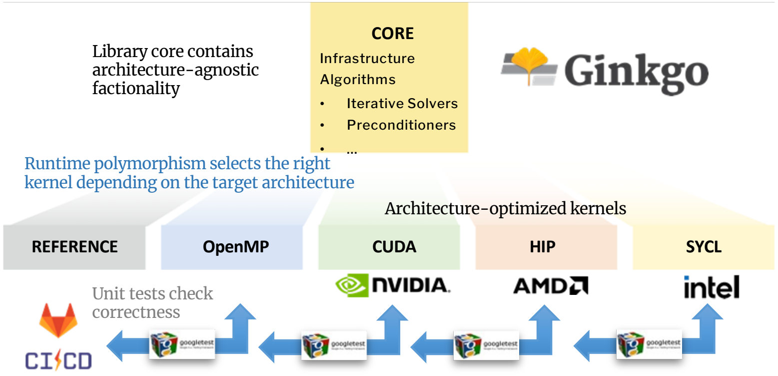
Focus efforts as lightweight tool in ECP to address challenges

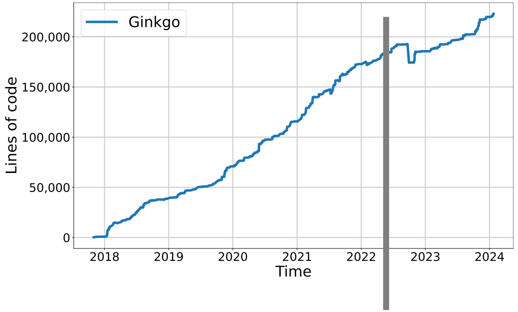
xSDK

Focus efforts

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

- Batched Routines
  - *Address application requirements*





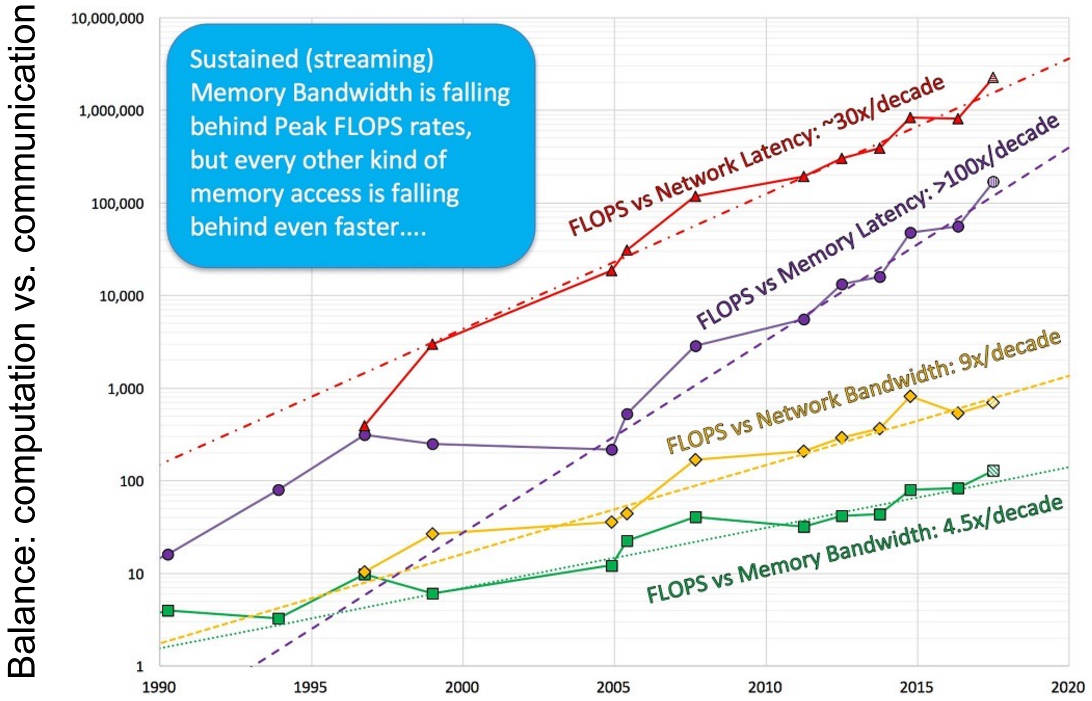
	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate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

Mixed precision focus effort

NVIDIA

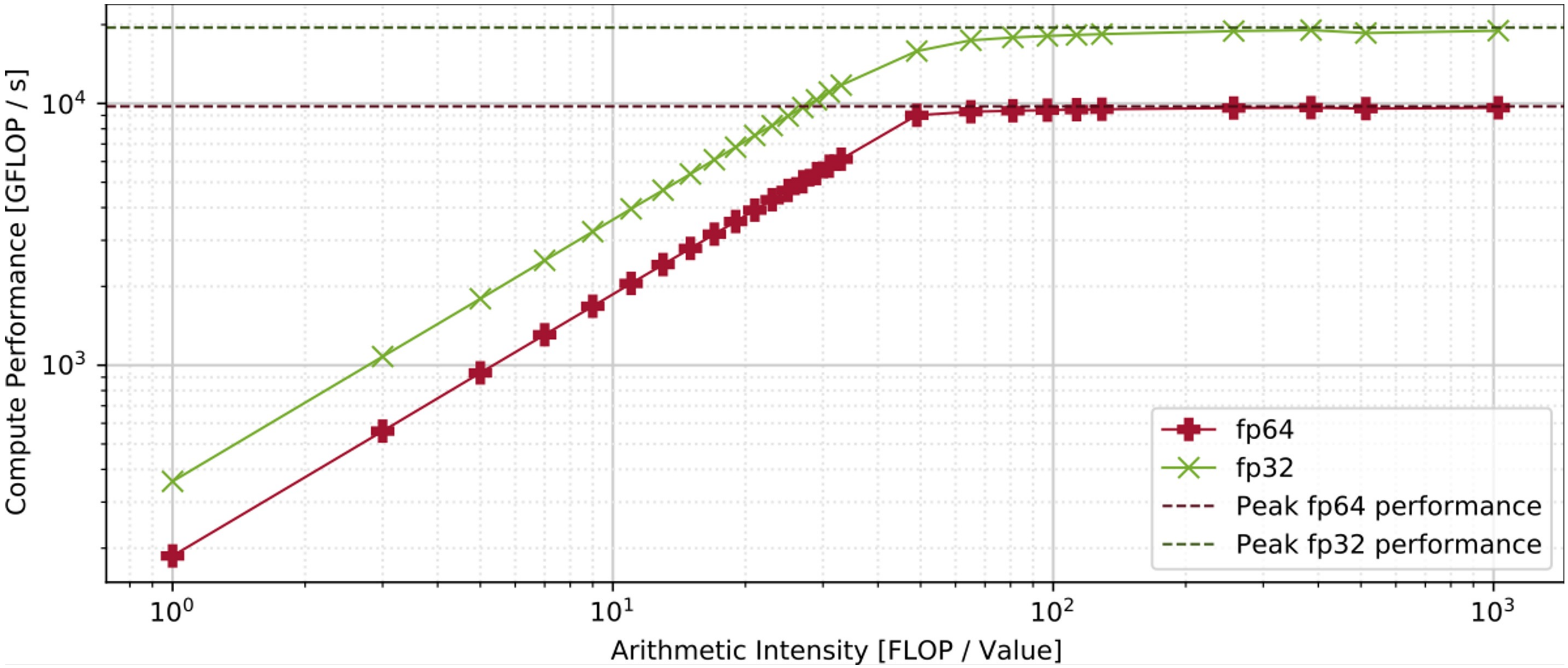
Form Factor	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core	989 teraFLOPS²
BFLOAT16 Tensor Core	1,979 teraFLOPS²
FP16 Tensor Core	1,979 teraFLOPS²
FP8 Tensor Core	3,958 teraFLOPS²
INT8 Tensor Core	3,958 TOPS²
GPU memory	80GB
GPU memory bandwidth	3.35TB/s

- (Dense) Matrix Performance
  > Vector Operation Performance
- Low Precision Performance
  > High Precision Performance



Trends in the relative performance of floating-point arithmetic and several classes of data access for select HPC servers over the past 25 years. Source: John McCalpin

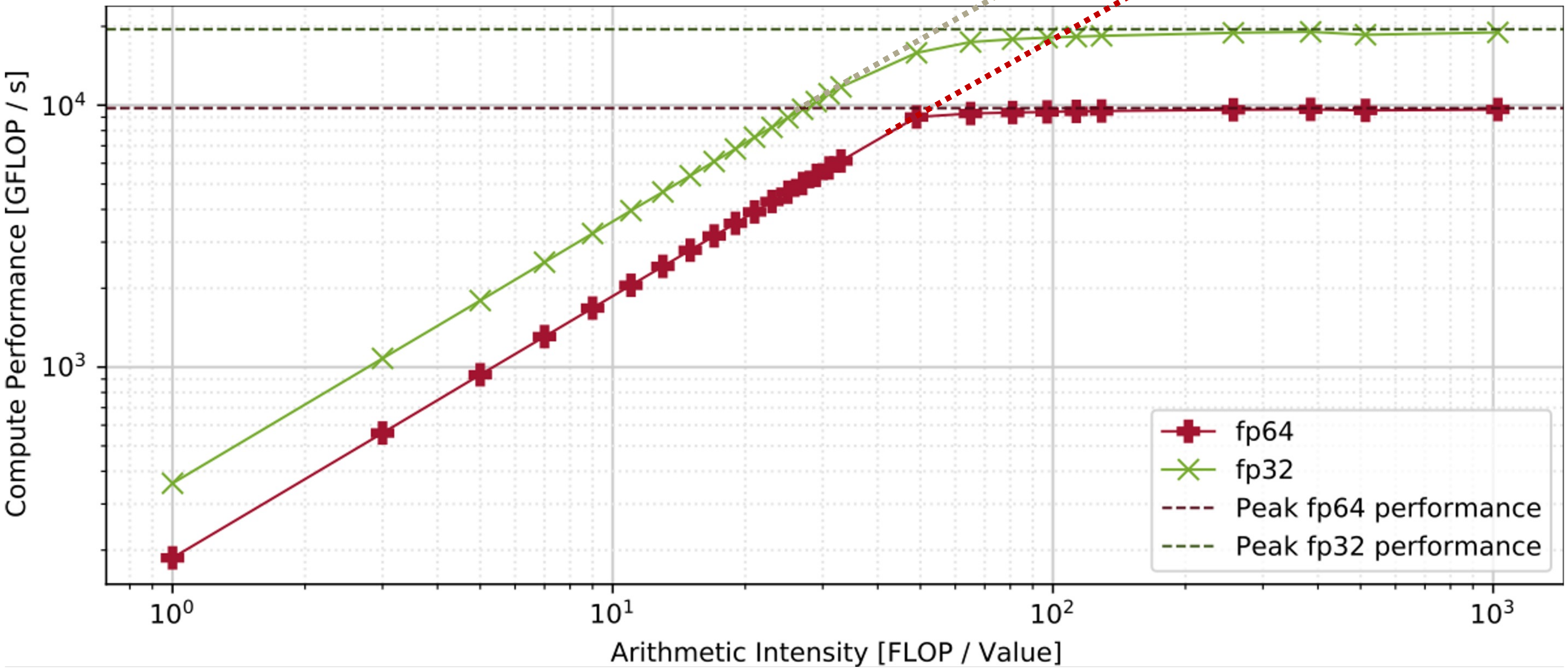
NVIDIA A100



Compute Performance [GFLOP / s]
10^4
10^3
10^0
10^1
10^2
10^3
Arithmetic Intensity [FLOP / Value]
fp64
fp32
Peak fp64 performance
Peak fp32 performance

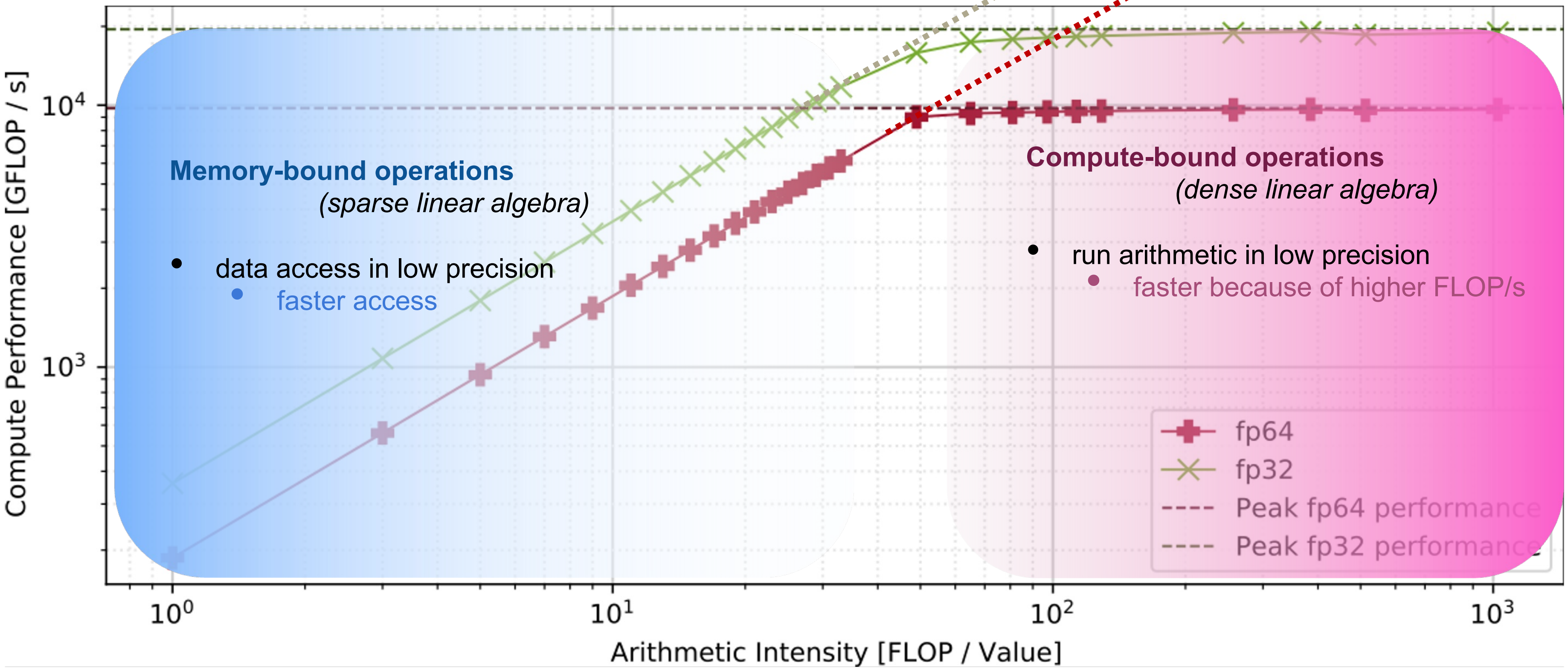
Matrix fp32

Matrix fp64



Matrix fp32

Matrix fp64



Linear System Ax=b with cond(A) ≈ 10^7

(apache2 from SuiteSparse) **NVIDIA V100 GPU**

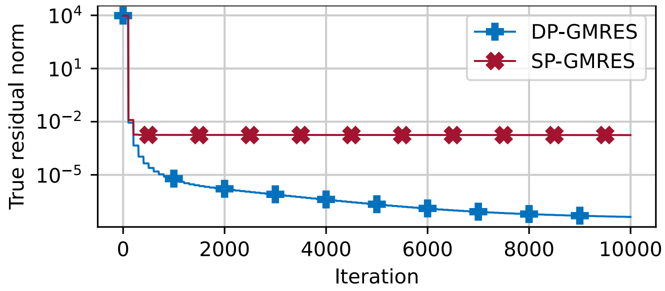
```
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms
```

Relative residual ~10^{-12}

```
Single precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 0.00175464
GMRES iteration count: 25000
GMRES execution time: 27376 ms
```

Relative residual ~10^{-7}

~2x faster!

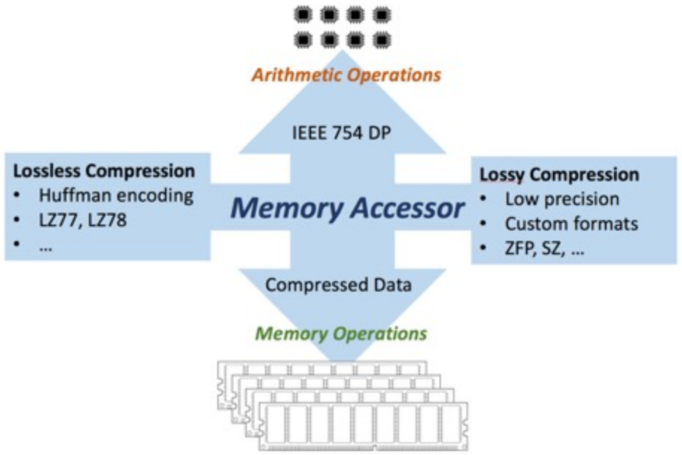


forward error ≈ (unit round-off) * (linear system's condition number)

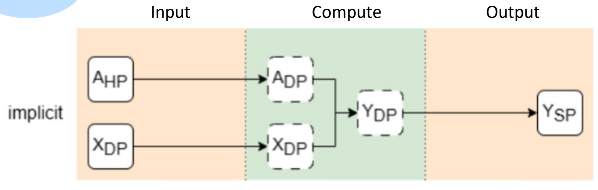
N. Higham: Accuracy and stability of numerical algorithms. SIAM, 2002.

Mixed precision focus effort

- Traditionally, we use a strong coupling between the precision formats used for arithmetic operations and storing data.
- *We should compute in fp64*
- *Data should be compressed for main memory access (low precision/compression)*
- *Compression / Conversion needs to happen on-the-fly*

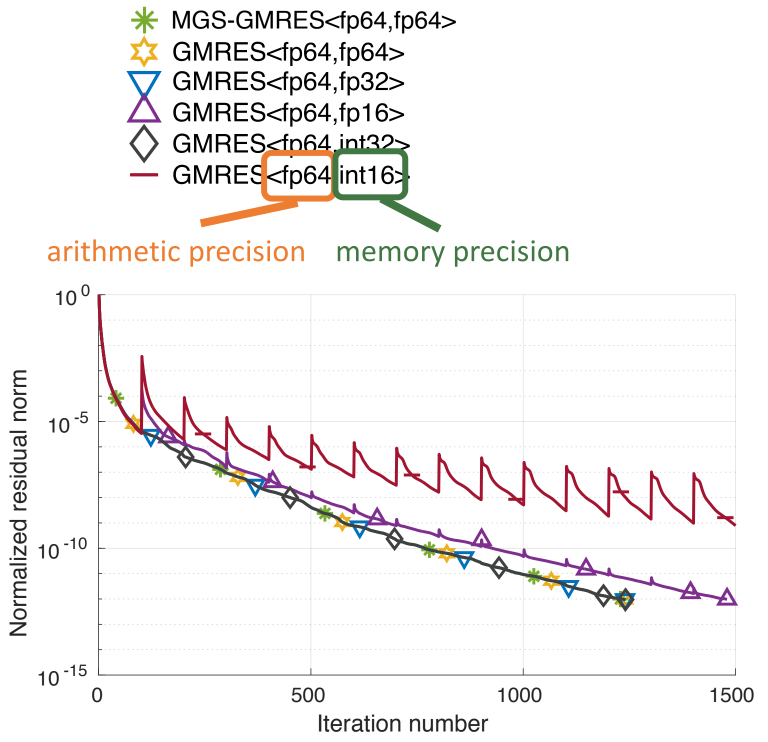


See Felix Liu's thesis



Compressed Basis (CB-) GMRES

- Use double precision in all arithmetic operations;
- Store Krylov basis vectors in lower precision;
  - Search directions are no longer DP-orthogonal;
  - Hessenberg system maps solution to “perturbed” Krylov subspace;
  - Additional iterations may be needed;
  - As long as the loss-of-orthogonality is moderate, we should see moderate convergence degradation;



Linear System Ax=b with cond(A) ≈ 10^7

(apache2 from SuiteSparse) **NVIDIA V100 GPU**

```
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms
```

Relative residual ~10^{-12}

```
Single precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 0.00175464
GMRES iteration count: 25000
GMRES execution time: 27376 ms
```

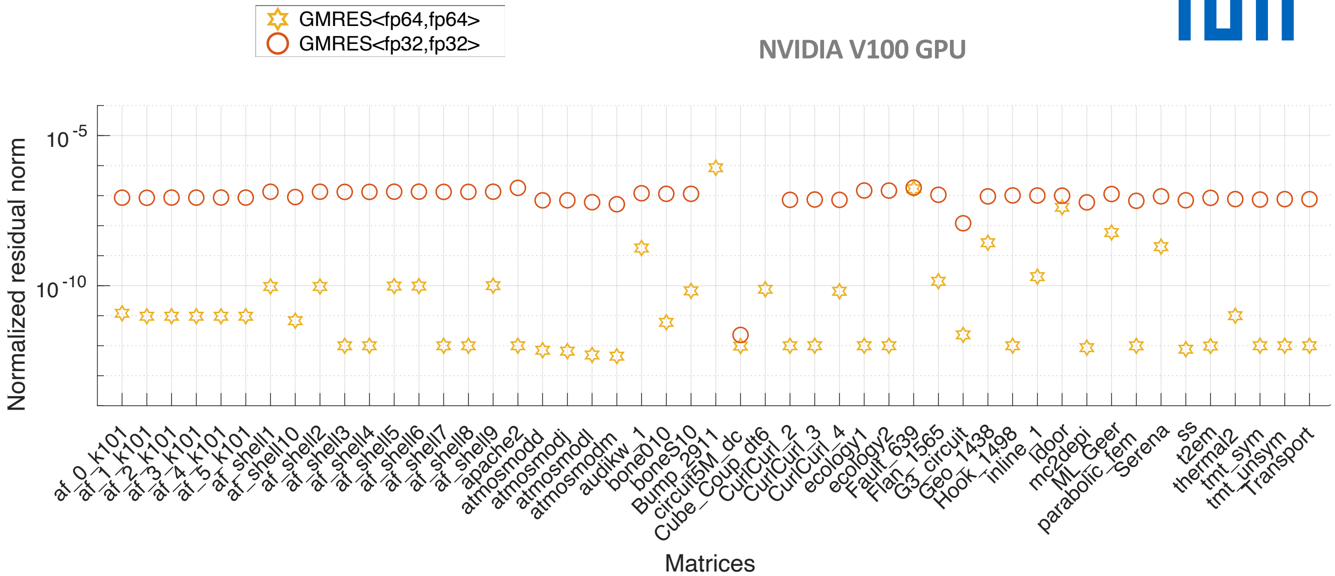
Relative residual ~10^{-7}

```
Compressed Basis GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6591e-09
GMRES iteration count: 23271
GMRES execution time: 29369 ms
```

Relative residual ~10^{-12}

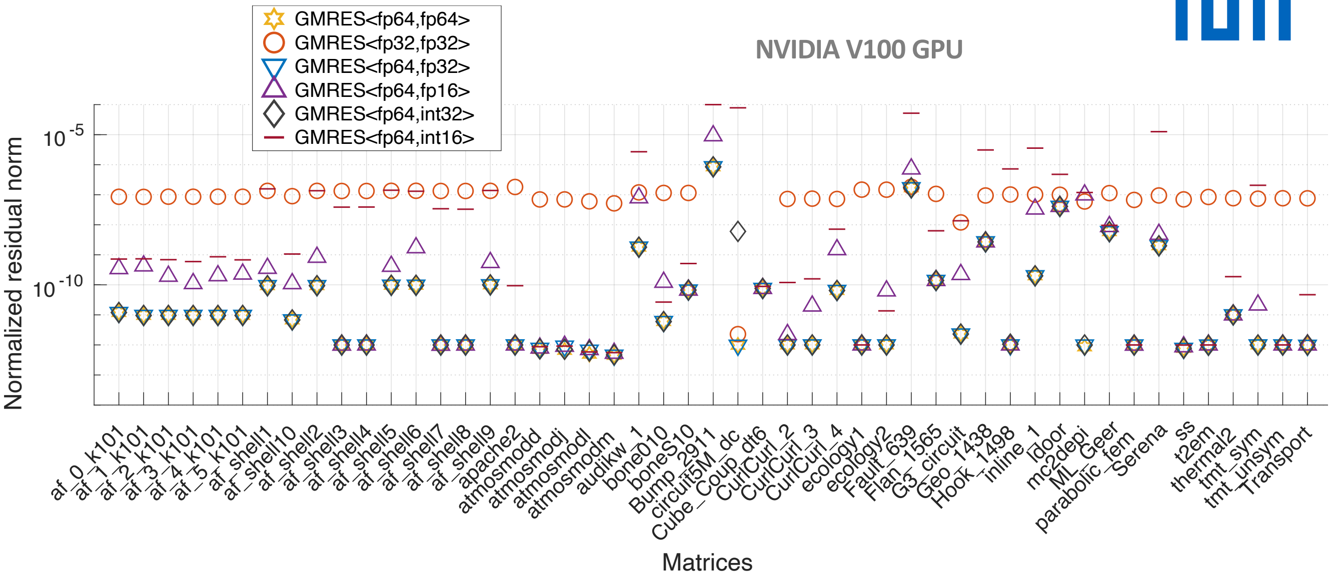
Accuracy of DP GMRES

Performance similar to SP GMRES



NVIDIA V100 GPU

- CB-GMRES using 32-bit storage preserves DP accuracy (SP-GMRES does not)

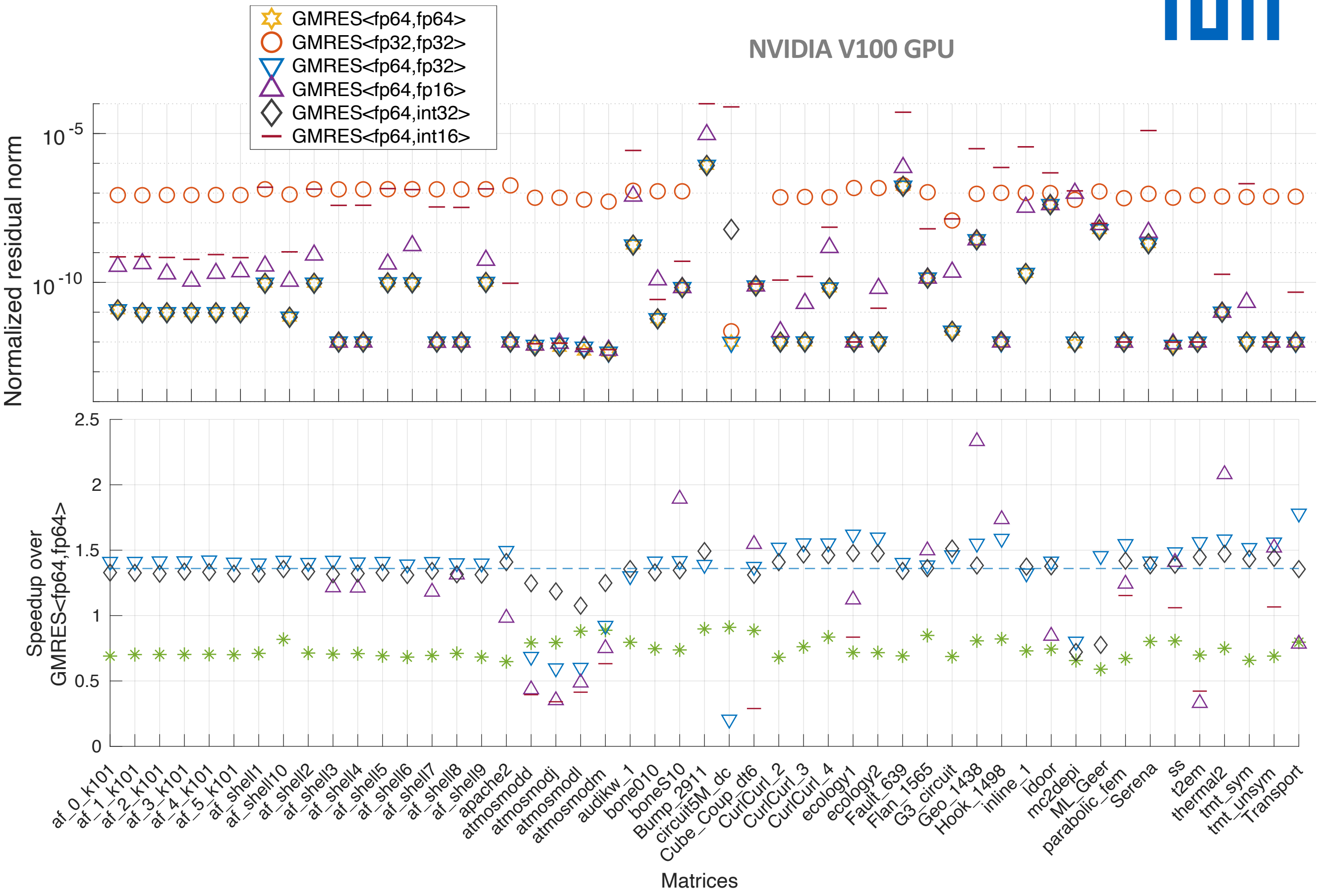


- CB-GMRES using 32-bit storage preserves DP accuracy (SP-GMRES does not)

- Speedups problem-dependent
- Speedup Ø1.4x (for restart 100)
- 16-bit storage mostly inefficient

Ginkgo

Aliaga JI, Anzt H, Grützmacher T, Quintana-Ortí ES, Tomás AE. Compressed basis GMRES on high-performance graphics processing units. *The International Journal of High Performance Computing Applications*. 2022;0(0). doi:10.1177/10943420221115140



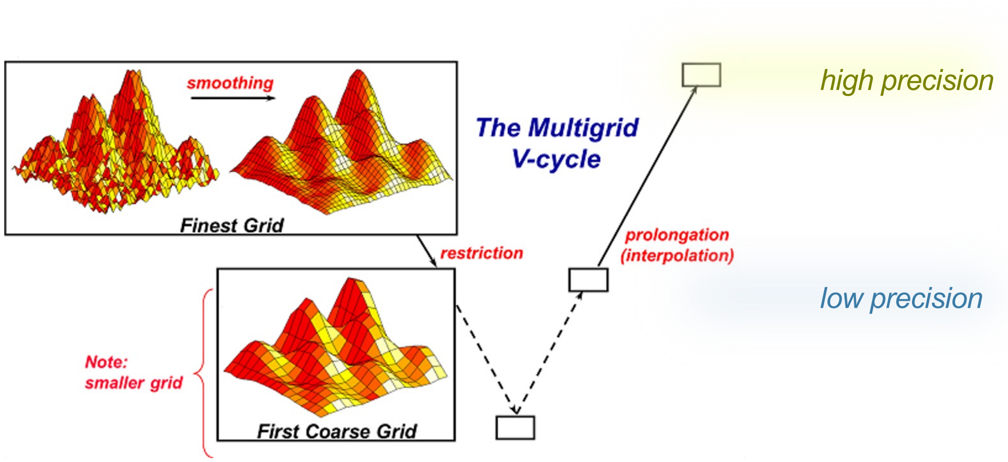
Mixed precision AMG on GPUs

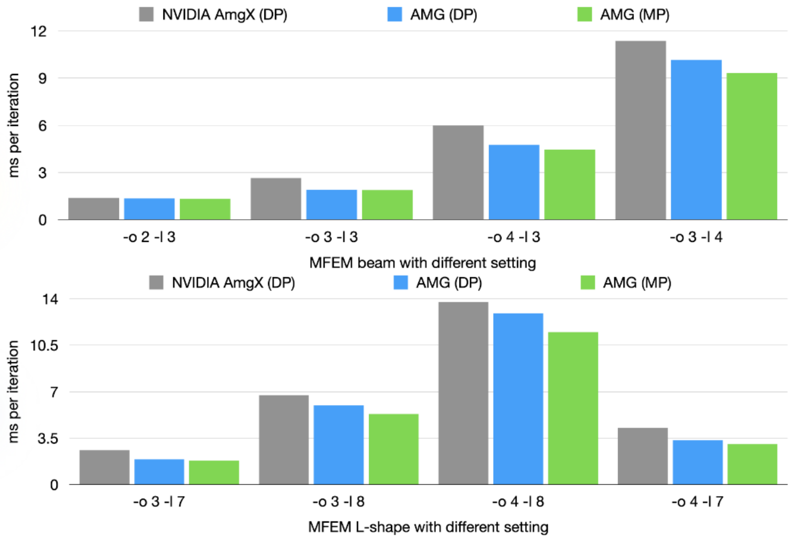
Mike Tsai

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

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

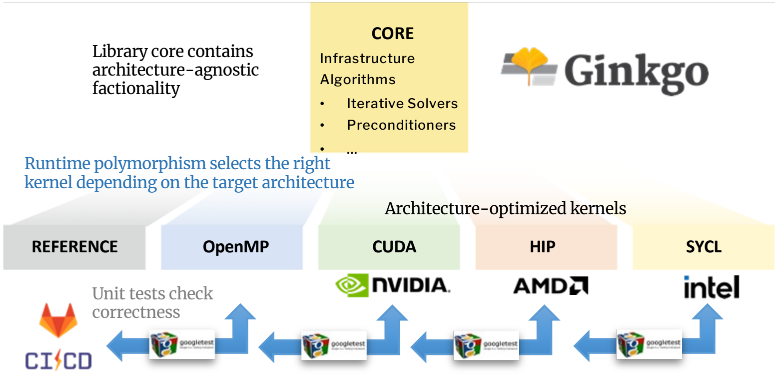
- **Mixed Precision Multigrid Preconditioner**

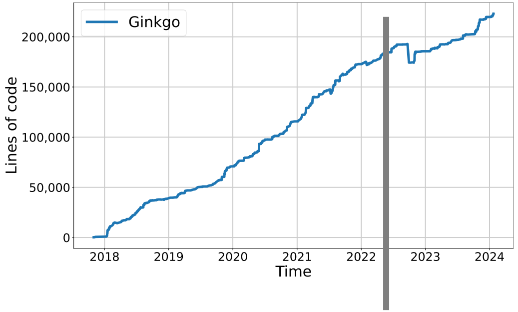




Stephen F. McCormick, Joseph Benzaken, Rasmus Tamstorf: Algebraic error analysis for mixed-precision multigrid solvers, https://arxiv.org/abs/2007.06614

Mixed precision focus effort





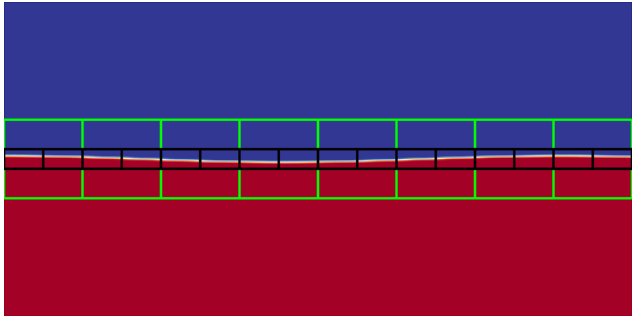
	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
Basic	SpMM	✓	✓	✓	✓
Basic	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
Krylov solvers	BiCGSTAB	✓	✓	✓	✓
Krylov solvers	CG	✓	✓	✓	✓
Krylov solvers	CGS	✓	✓	✓	✓
Krylov solvers	GMRES	✓	✓	✓	✓
Krylov solvers	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
Preconditioners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

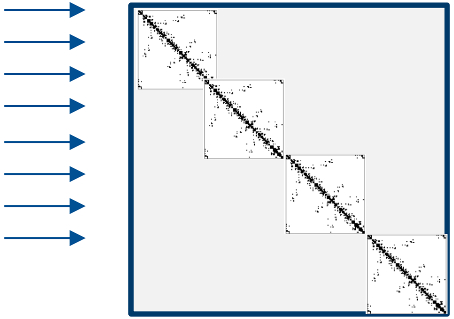
Batched focus effort – Combustion Simulations

Batched iterative solvers for SUNDIALS / PeleLM

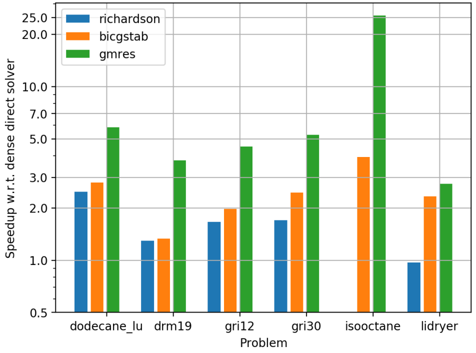
PeleLM is a parallel, adaptive mesh refinement (AMR) code that solves the reacting Navier-Stokes equations in the low Mach number regime. The core libraries for managing the subcycling AMR grids and communication are found in the AMReX source code.

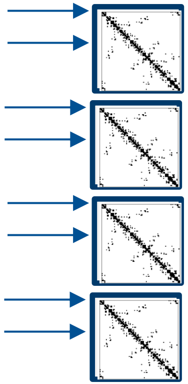
https://amrex-combustion.github.io/PeleLM/overview.html





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





Batched Sparse Iterative Solvers for Computational Chemistry Simulations on GPUs

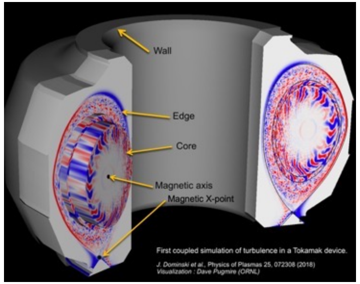
Publisher: IEEE | Cite This | PDF

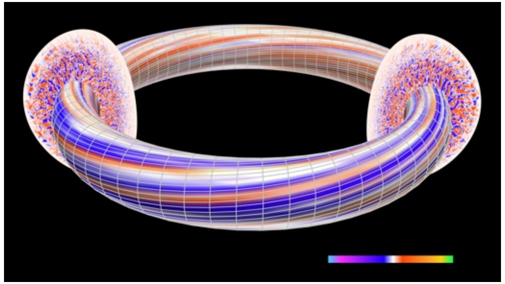
Isha Aggarwal ; Aditya Kashi ; Pratik Nayak ; Cody J. Balos ; Carol S. Woodward ; Hartwig Anzt All Authors

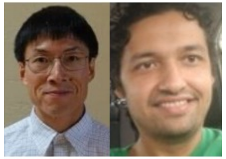
Batched focus effort – Fusion Plasma Simulations

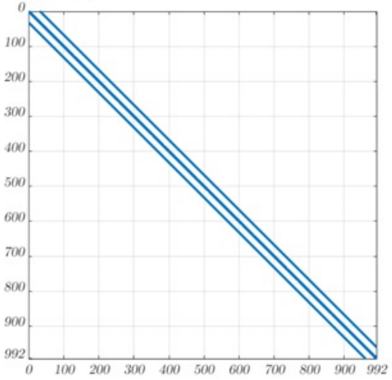
XGC is a gyrokinetic particle-in-cell code, which specializes in the simulation of the edge region of magnetically confined thermonuclear fusion plasma. The simulation domain can include the magnetic separatrix, magnetic axis and the biased material wall. XGC can run in total-delta-f, and conventional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

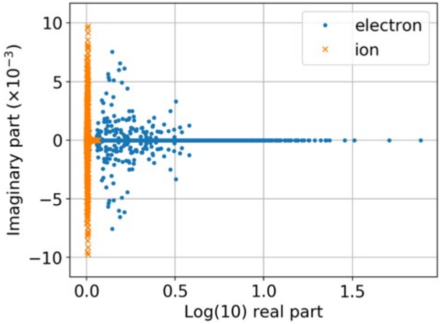
Source: https://xgc.pppl.gov/html/general_info.html





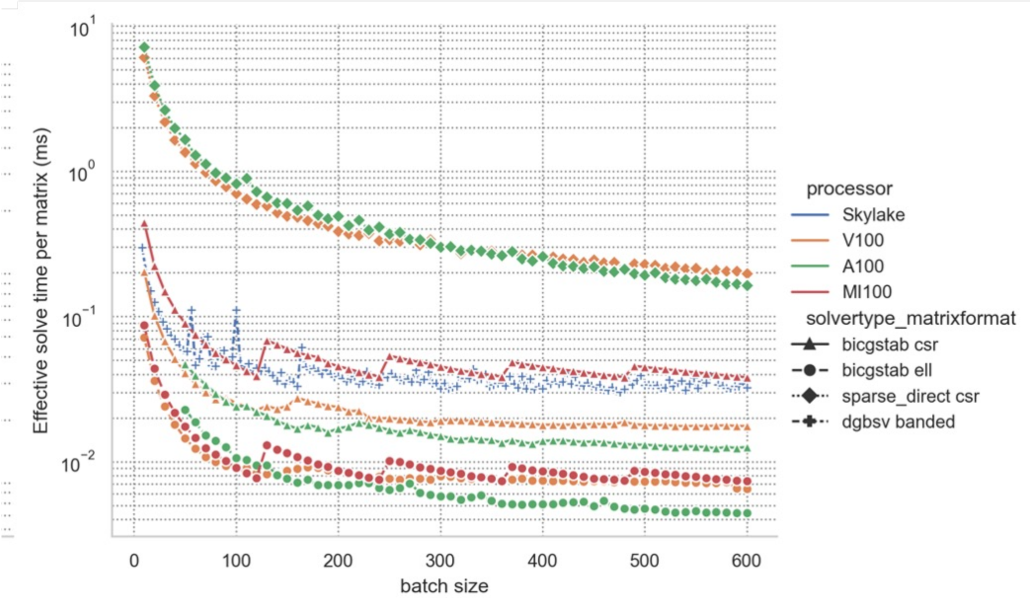


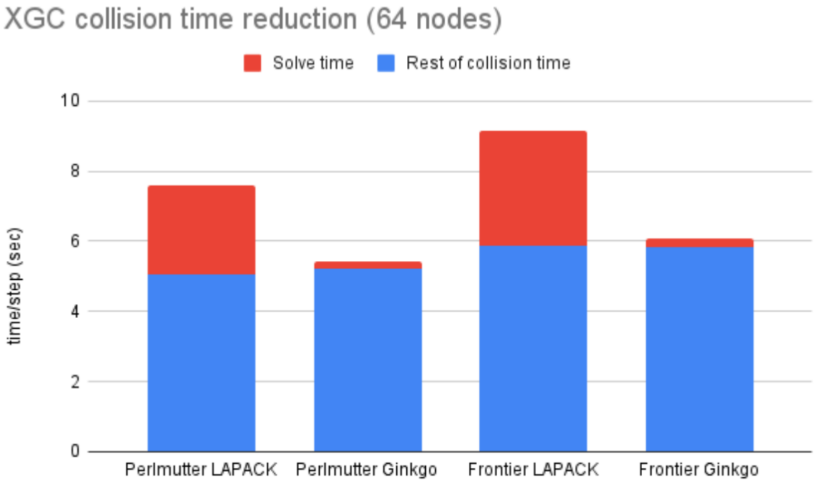




- Two species
- Ions easy to solve
- Electrons hard to solve
- Banded matrix structure
- Non-symmetric, need BiCGSTAB
- n = ~1,000
- nz = ~9,000

Batched focus effort – Fusion Plasma Simulations





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.

Sparse direct solvers for power grid simulations

Mathematical Formulation of the ExaSGD Core Challenge

Security constrained multiperiod AC optimal power flow analysis

Posed as an optimization problem:

Find

$$\min_{x_t, y_{tsk}} \left(\sum_t F_t(x_t) + \right.$$ generator fuel cost

$$\left. + \sum_{tsk} G_{tsk}(x_t + y_{tsk})\right)$$ wind curtailment, load shedding, power imbalance, etc.

Subject to:

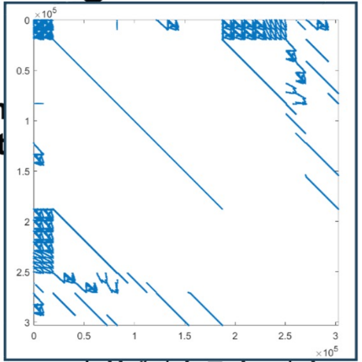
$H_{tsk}(x_t, y_{tsk}) = 0$ — flow definitions, power balance

$Q_{tsk}(x_t, y_{tsk}) \leq 0$ — bounds: generator power, voltage, branch flow

$R_t(x_t, x_{t+1}) \leq 0$ — generator ramping limit

The optimization problem ... the underlying linear syst...

$$\begin{bmatrix} K_1 & & & & & \\ & K_2 & & & & \\ & & \boxed{K_3} & & & \\ & & & \dots & & \dots \\ & & & & K_N & B_N \\ B_1^T & B_2^T & B_3^T & \dots & B_N^T & K_0 \end{bmatrix} \begin{bmatrix} \vdots \\ \dots \\ y_N \\ x \end{bmatrix} = \begin{bmatrix} \vdots \\ \dots \\ r_N \\ r_0 \end{bmatrix}$$



- The characteristic block-arrow coupling structure can be exploited to decompose the optimization problem, nevertheless there is no solver that can tackle this on a GPU-based architecture.

EXASGD

Sparse direct solvers for power grid simulations

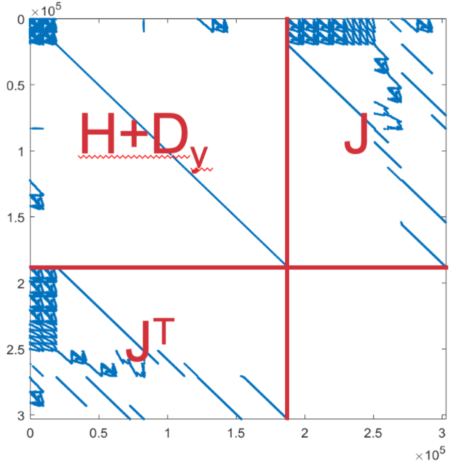
Underlying KKT Linear System Properties

See Felix Liu's thesis

- Security constrained optimal power flow analysis.
- The interior method strategy leads to symmetric indefinite linear systems

$$\overbrace{\begin{bmatrix} H + D_y & J \\ J^T & 0 \end{bmatrix}}^{K_k} \overbrace{\begin{bmatrix} \Delta y \\ \Delta \lambda \end{bmatrix}}^{\Delta x_k} = \overbrace{\begin{bmatrix} r_y \\ r_\lambda \end{bmatrix}}^{r_k},$$

  - J – sparse constraints Jacobian,
  - H – sparse Hessian,
  - D_y – arises from log-barrier function

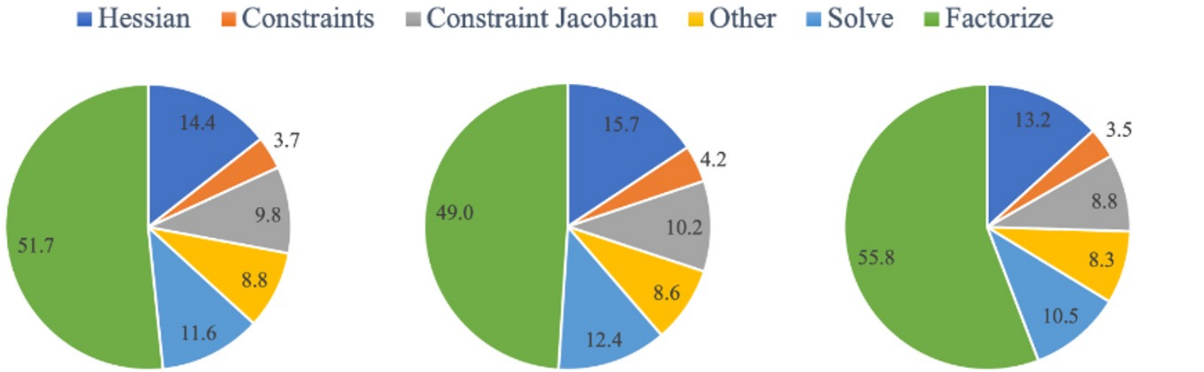


Typical sparsity pattern of optimal power flow matrices: No obvious structure that can be used by linear solver.

- The challenge: we need to solve a long sequences of such systems.

Sparse direct solvers for power grid simulations

Grid	Buses	Generators	Lines	$N(K_k)$	$nnz(K_k)$
Northeastern US	25 K	4.8 K	32.3 K	108 K	1.19 M
Eastern US	70 K	10.4 K	88.2 K	296 K	3.20 M
Western and Eastern US	82 K	13.4 K	104.1 K	340 K	3.73 M



(a) Northeast U.S. grid (b) Eastern U.S. grid (c) Eastern and Western U.S. grids

EXASGD

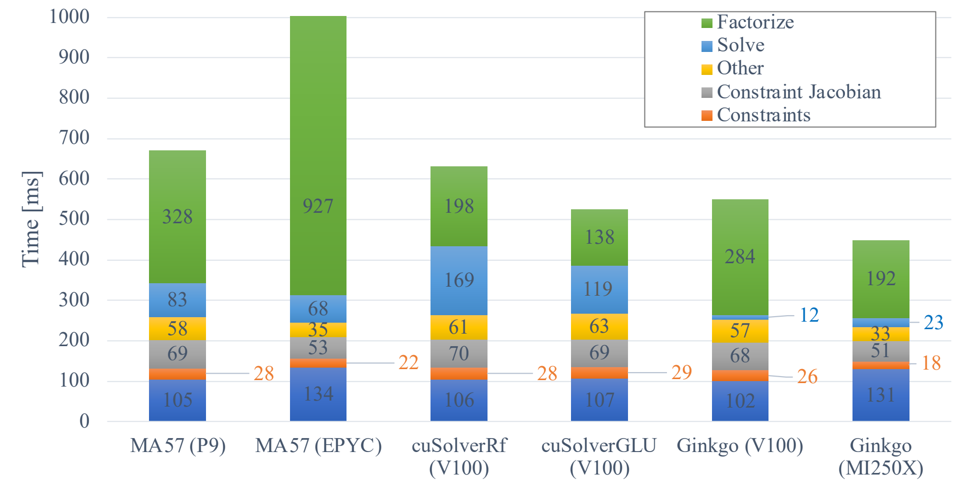
Sparse direct solvers for power grid simulations

You are viewing Presentation's screen View Options

Liner Solver Performance within Optimization Algorithm

Average per iteration times (including first iteration on CPU)

- Each GPU solution outperforms all CPU baselines.
- Ginkgo performance improves on a better GPU.
- Iterative refinement configuration affects linear solver performance and optimization solver convergence.
- Ginkgo is the first GPU-resident sparse direct linear solver.



ECP EXASCALE COMPUTING PROJECT

Multiple promising GPU-resident sparse linear solvers

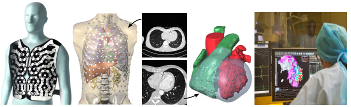
EXASGD

Now, after the completion of ECP

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

EuroHPC Joint Undertaking

MICROCARD

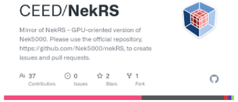


https://www.microcard.eu

- BMBF PDExa and ExaSIM projects use Ginkgo

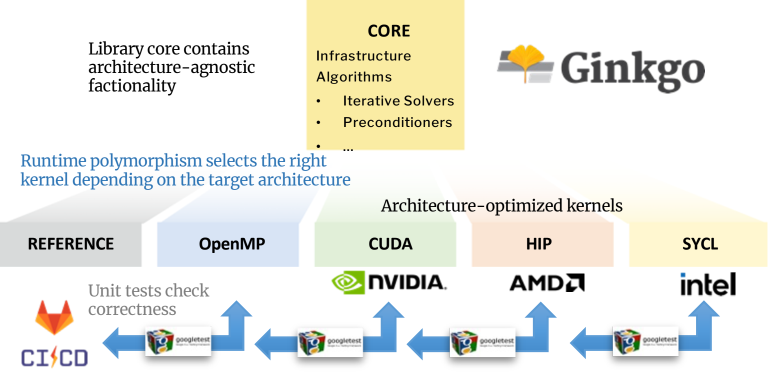
OpenFOAM
The Open Source CFD Toolbox

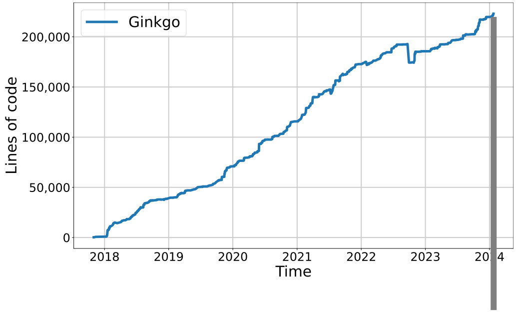
deal.II



- Companies are evaluating Ginkgo

MathWorks · HUAWEI · aws





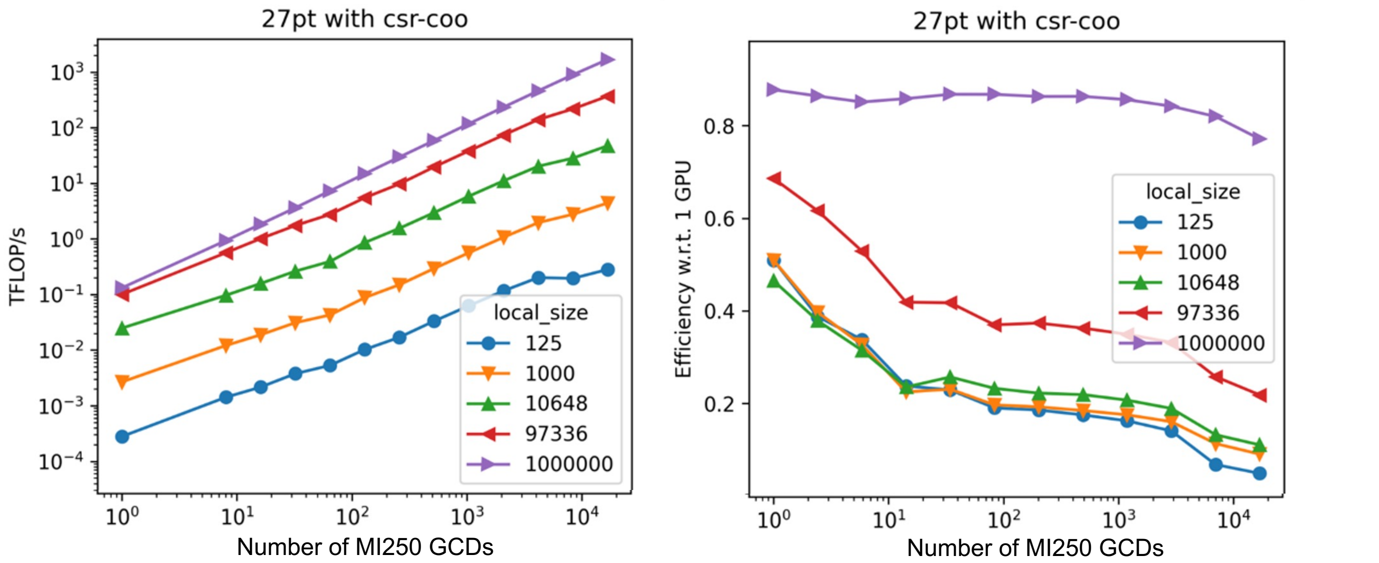
FUNCTIONALITY		OMP	CUDA	HIP	DPC++
Basic	SpMV	✓ (MPI)	✓ (MPI)	✓ (MPI)	✓ (MPI)
	SpMM	✓ (MPI)	✓ (MPI)	✓ (MPI)	✓ (MPI)
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓ (MPI)	✓ (MPI)	✓ (MPI)	✓ (MPI)
	CG	✓ (MPI)	✓ (MPI)	✓ (MPI)	✓ (MPI)
	CGS	✓	✓	✓	✓
	GCR	✓ (MPI)	✓ (MPI)	✓ (MPI)	✓ (MPI)
	GMRES	✓ (MPI)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

✓ (MPI) *MPI Support* ✓ *Single-GPU Support*

Scalability of Ginkgo on Frontier (#1 TOP500, AMD MI250)

Weak scaling: problem size increases with parallel resources

Weak scaling up to 8k AMD MI250 GPUs (16k GCDs)

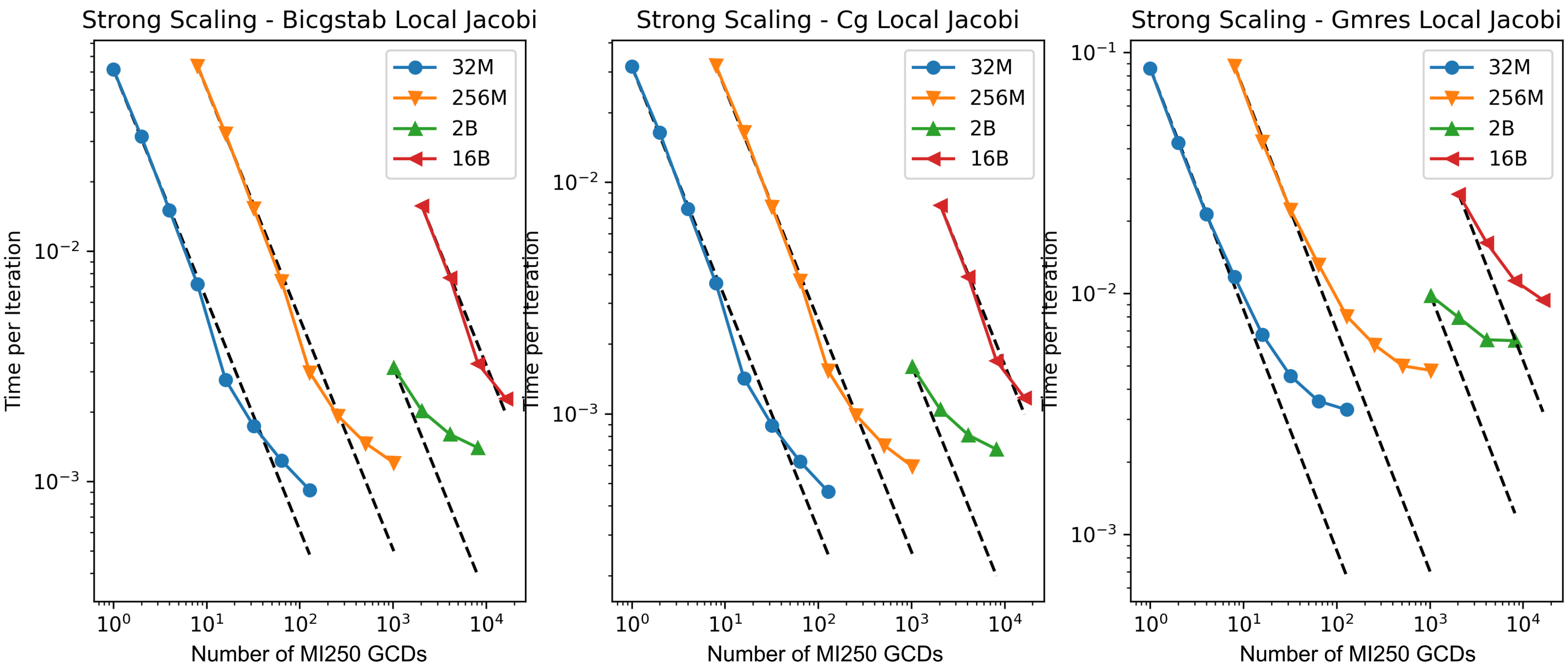


Significant Compute Waste!

Scalability of Ginkgo on Frontier (#1 TOP500, AMD MI250)

Strong scaling: problem size increases with parallel resources

Strong scaling up to 8k AMD MI250 GPUs (16k GCDs)



Lessons learnt over the years

- **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, companies, 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. Industry career paths.**

- **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.**

EXASCALE COMPUTING PROJECT