TUM School of Computation, Information and Technology
Technische Universität München

IGHASC 2024: Indo-German Workshop on Hardware-aware Scientific Computing
Heidelberg, Oct 2024

It is all in the GPUs - How the Hardware Architecture impacted Scientific Software in the US Exascale Computing Project

Hartwig Anzt
Chair of Computational Mathematics, TU Munich
Adjunct Professor, University of Tennessee

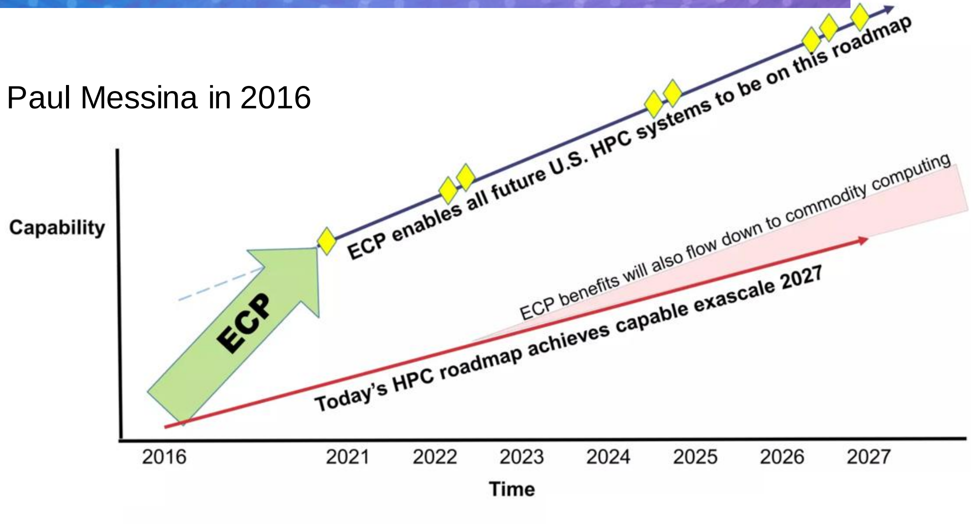
The US Exascale Computing Project

Advancing Scientific Discovery

Strengthening National Security

Improving Industrial Competitiveness

© Paul Messina in 2016



The US Exascale Computing Project

- 3 computers. (~2B)
  - $600M each
  - $400M to vendors for Design, Path, Fast - Forward

AMD Based
(Up & running)
20 MW

Intel Based
(Up & running)
40 MW

AMD APU Based
(being commissioned)

The US Exascale Computing Project

- 3 computers. (~2B)
    - $600M each
    - $400M to vendors for Design, Path, Fast - Forward

AMD Based
(Up & running)
20 MW

Intel Based
(Up & running)
40 MW

AMD APU Based
(being commissioned)

TOP 500 The List.

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
6	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194

98% of FLOP/s in GPUs
2% of FLOP/s in CPUs

92% of FLOP/s in GPUs
8% of FLOP/s in CPUs

The US Exascale Computing Project

- 3 computers. (~2B)
  - $600M each
  - $400M to vendors for Design, Path, Fast - Forward
- Application and Software Development(~2B)

AMD Based
(Up & running)
20 MW

Intel Based
(Up & running)
40 MW

AMD APU Based
(being commissioned)

LatticeQCD
Validate Fundamental Laws of Nature
Objective: Validate Fundamental Laws

NWChemEx
Tackling Chemical, Materials, and Biomolecular Challenges in Exascale

GAMESS
General Atomic and Molecular Electronic Structure System

ExaStar
Exascale Models of Stellar Explosions
Objective: Demystify Origin of Chemical Elements

ExaSky
Computing at the Extreme Scales
Objective: Cosmological Probe of the Standard Model of Particle Physics

EQSIM
High-Performance, Multidisciplinary Simulations for Regional-Scale Earthquake Hazard/ Risk Assessments

EXAALT
Molecular Dynamics at Exascale
Objective: Simultaneously address time

ExaAM
Transforming Additive Manufacturing through Exascale Simulation

QMCPACK
Quantum Mechanics at Exascale

WDMApp
High-fidelity Whole Device Modeling of Magnetically Confined Fusion Plasmas

ExaSMR
Coupled Monte Carlo Neutronics and Fluid Flow Simulation of Small Modular Reactors

WarpX
Exascale Modeling of Advanced Particle Accelerators

ExaSGD
Optimizing Stochastic Grid Dynamics at Exascale

CANDLE
Exascale Deep Learning–Enabled Precision Medicine for Cancer

ExaBiome
Exascale Solutions for Microbiome Analysis

ExaWind
Exascale Predictive Wind Plant Flow Physics Modeling

Combustion-PELE
High-efficiency, Low- emission Combustion Engine Design

MFIX-Exa
Performance Prediction of Multiphase Energy Conversion Device

Ristra
Multi-physics simulation tools for weapons-relevant applications

MAPP
Multi-physics simulation tools for High Energy Density Physics (HEDP) and weapons- relevant applications for DOE and DoD.

EMPIRE AND SPARC
EMPIRE addresses electromagnetic plasma physics, and SPARC addresses reentry aerodynamics

Subsurface
Exascale Subsurface Simulator of Coupled Flow, Transport, Reactions, and Mechanics

E3SM-MMF
Cloud-Resolving Climate Modeling of the Earth's Water Cycle

ExaFEL
Data Analytics at Exascale for Free Electron Lasers

Adaptive Mesh Refinement

Efficient Exascale Discretizations

Online Data Analysis and Reduction at the Exascale

Particle-Based Applications

Efficient Implementation of Key Graph Algorithms

Exascale Machine Learning Technologies

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

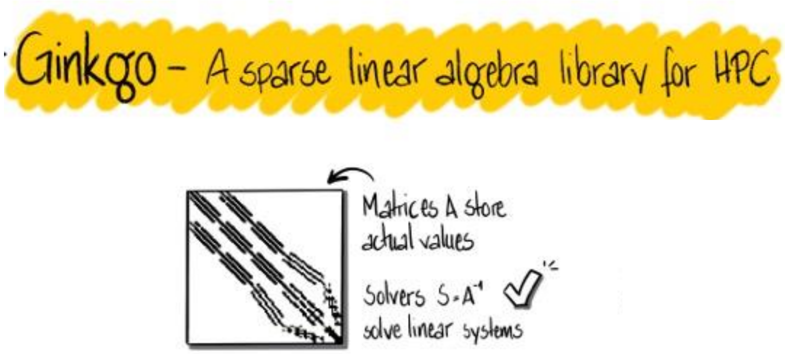
- Born and raised in Karlsruhe
- PhD in Numerical Mathematics from KIT in 2012
- Focus on computational linear algebra and high performance computing (HPC)
- Linear solvers, preconditioners, …
- During my PostDoc at the University of Tennessee, I developed MAGMA sparse

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

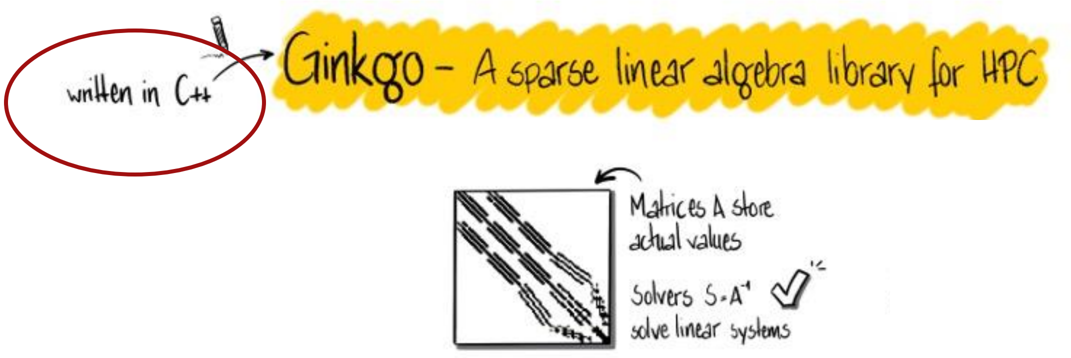
MATLAB

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

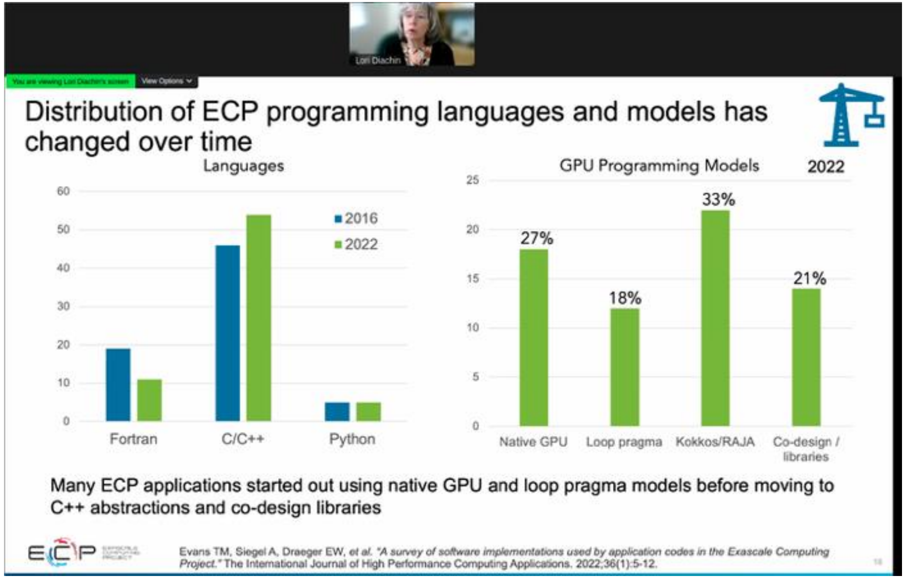
Designing a math toolset for ECP applications

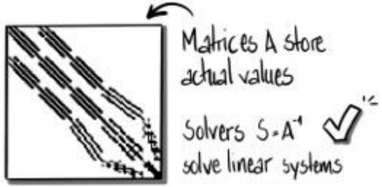


Designing a math toolset for ECP applications

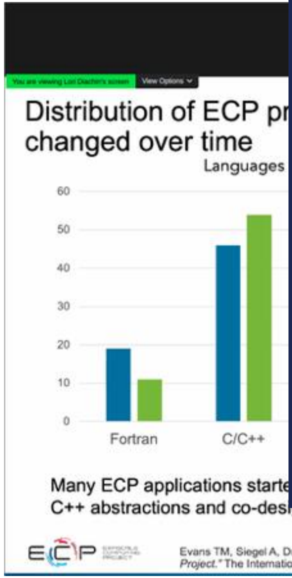


Designing a math toolset for ECP applications



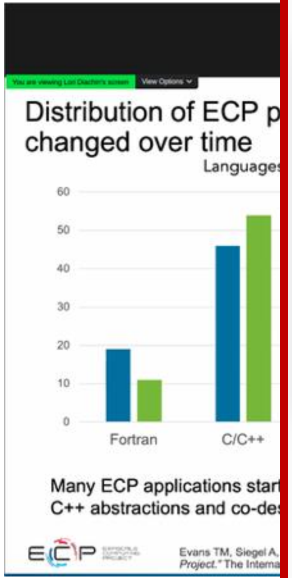


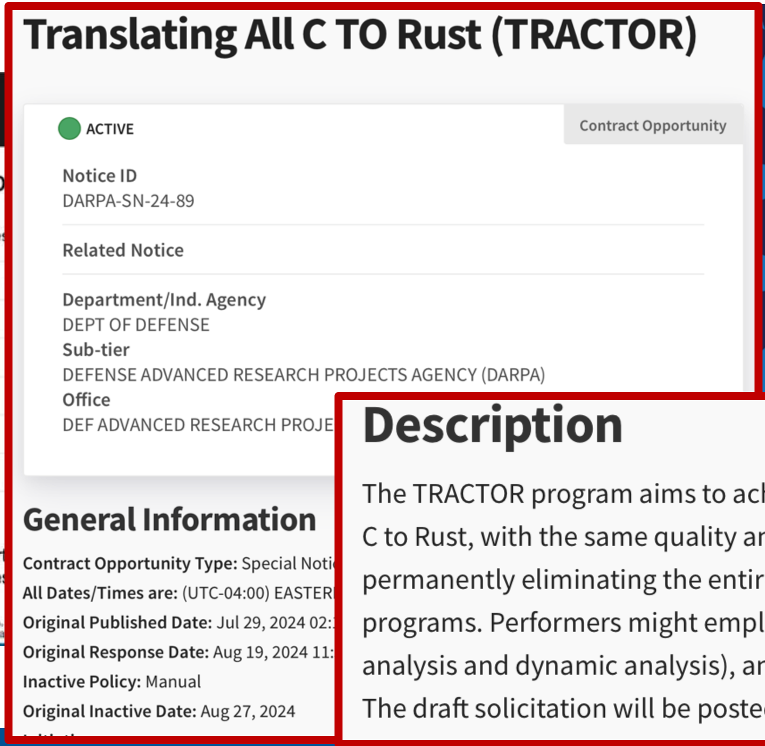
Designing a math toolset for ECP applications



Memory safety vulnerabilities are a class of vulnerability affecting how memory can be accessed, written, allocated, or deallocated in unintended ways.[iii] Experts have identified a few programming languages that both lack traits associated with memory safety and also have high proliferation across critical systems, such as C and C++.[iv] Choosing to use memory safe programming languages at the outset, as recommended by the Cybersecurity and Infrastructure Security Agency's (CISA) Open-Source Software Security Roadmap is one example of developing software in a secure-by-design manner.[v]

Designing a math toolset for ECP applications





Translating All C TO Rust (TRACTOR)

ACTIVE

Contract Opportunity

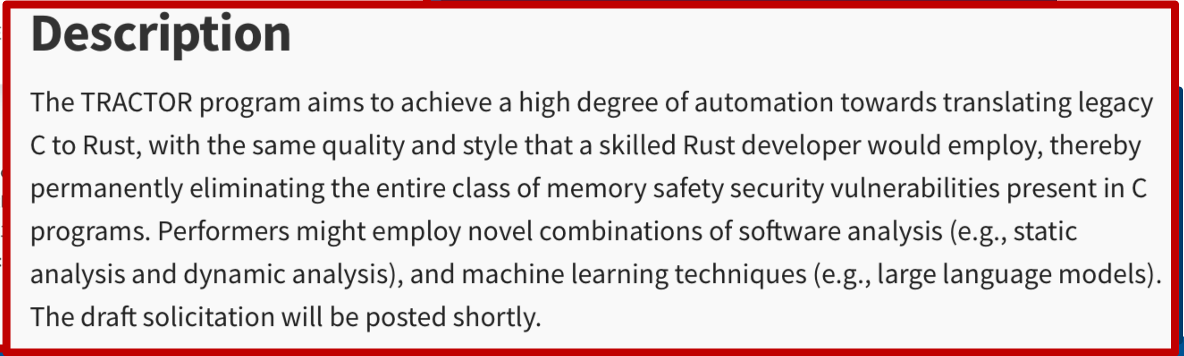
Notice ID
DARPA-SN-24-89

Related Notice

Department/Ind. Agency
DEPT OF DEFENSE
Sub-tier
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (DARPA)
Office
DEF ADVANCED RESEARCH PROJE

General Information

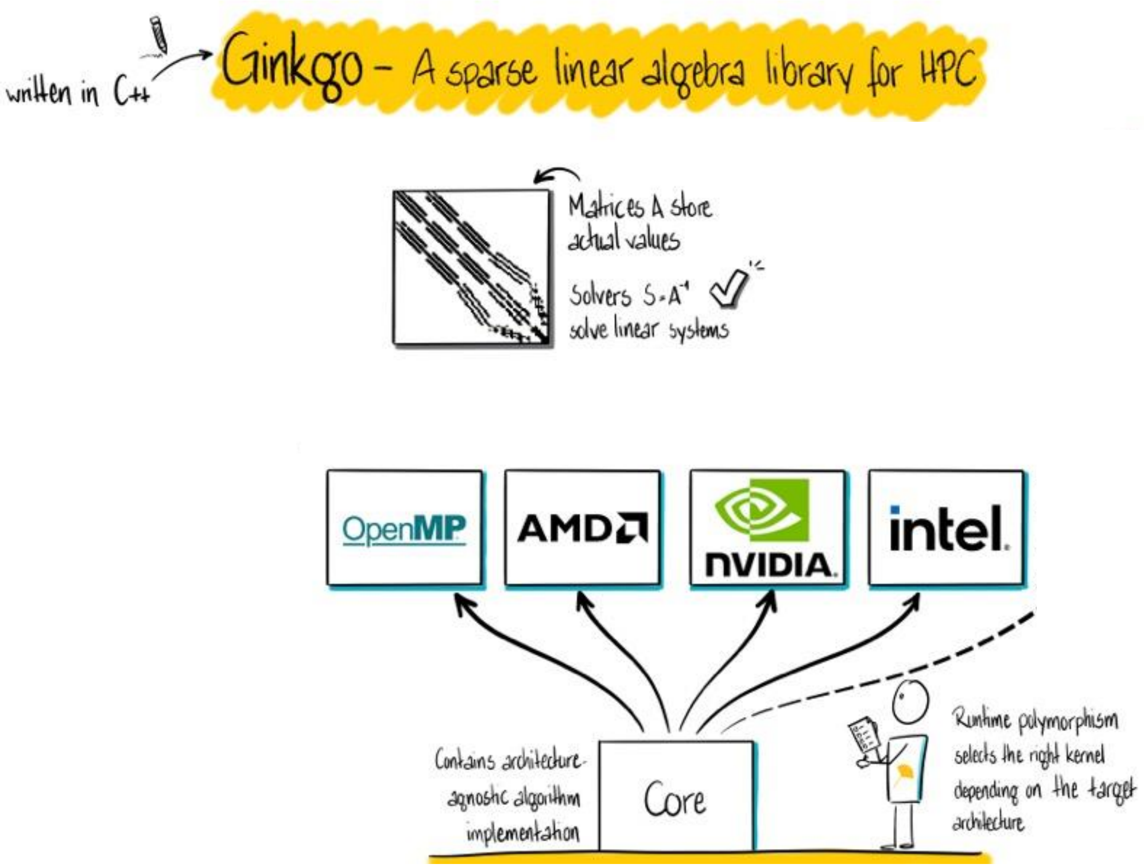
Contract Opportunity Type: Special Noti
All Dates/Times are: (UTC-04:00) EASTER
Original Published Date: Jul 29, 2024 02:
Original Response Date: Aug 19, 2024 11:
Inactive Policy: Manual
Original Inactive Date: Aug 27, 2024



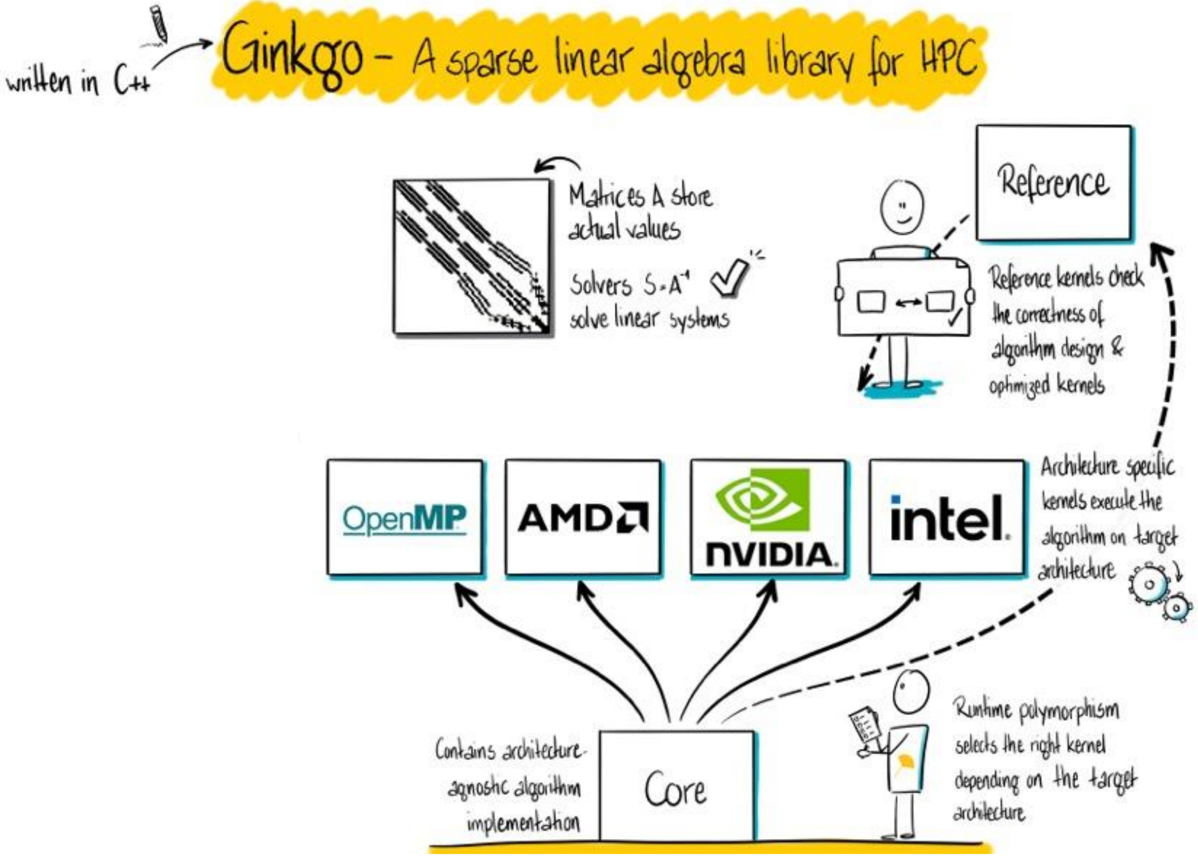
Description

The TRACTOR program aims to achieve a high degree of automation towards translating legacy C to Rust, with the same quality and style that a skilled Rust developer would employ, thereby permanently eliminating the entire class of memory safety security vulnerabilities present in C programs. Performers might employ novel combinations of software analysis (e.g., static analysis and dynamic analysis), and machine learning techniques (e.g., large language models). The draft solicitation will be posted shortly.

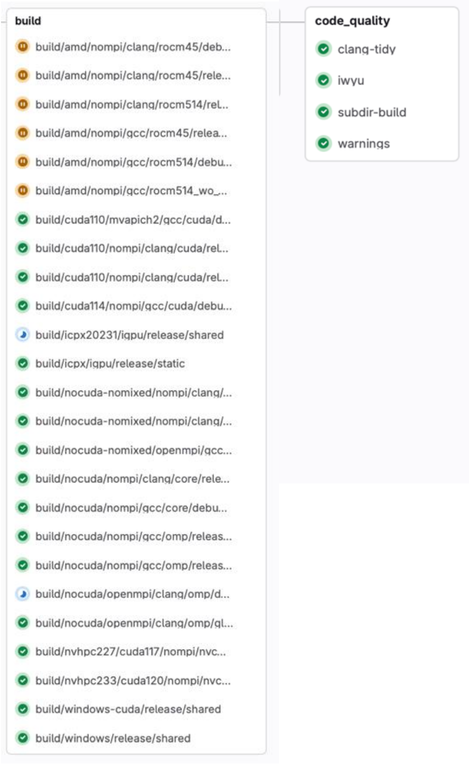
Designing a math toolset for ECP applications

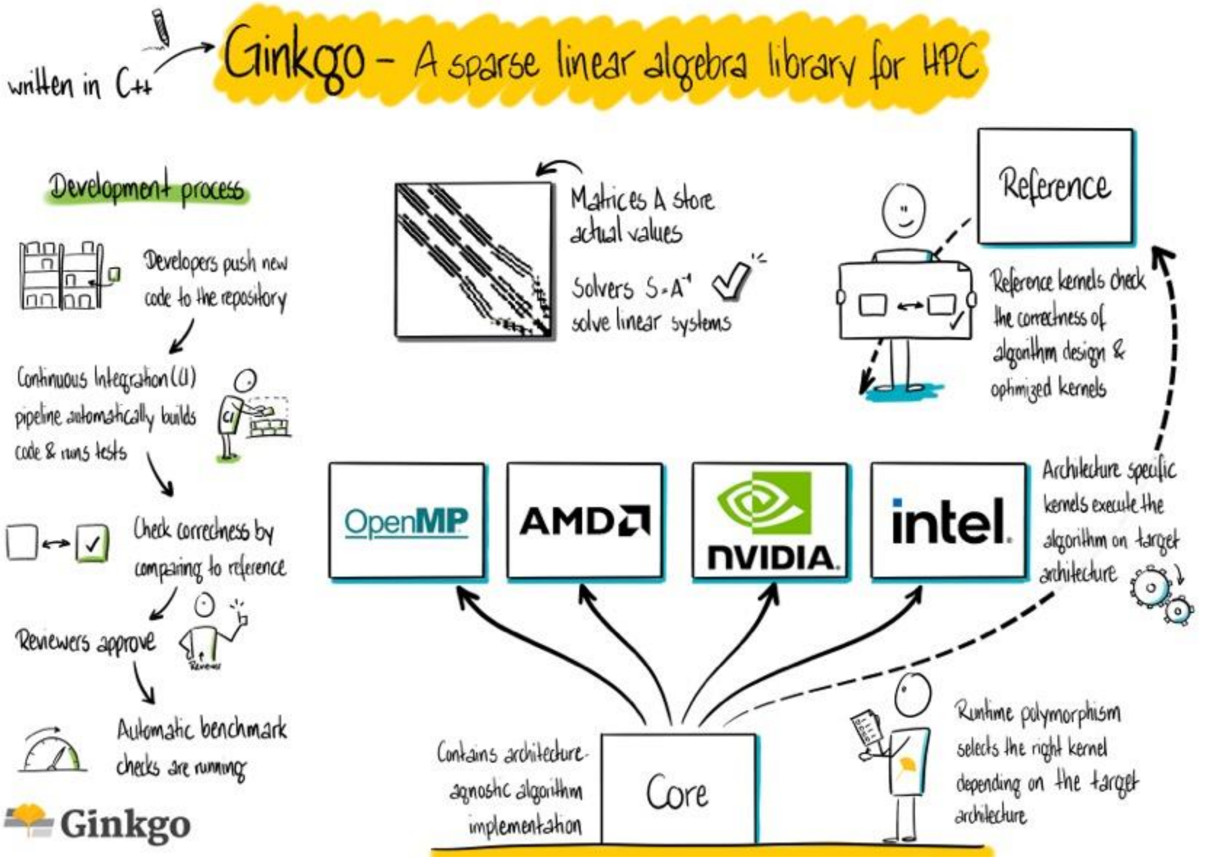


Designing a math toolset for ECP applications

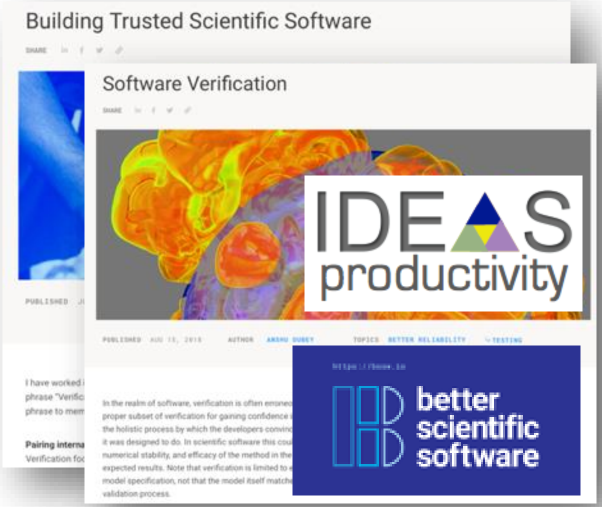


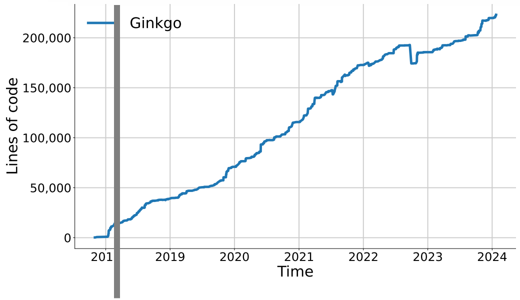
Designing a math toolset for ECP applications

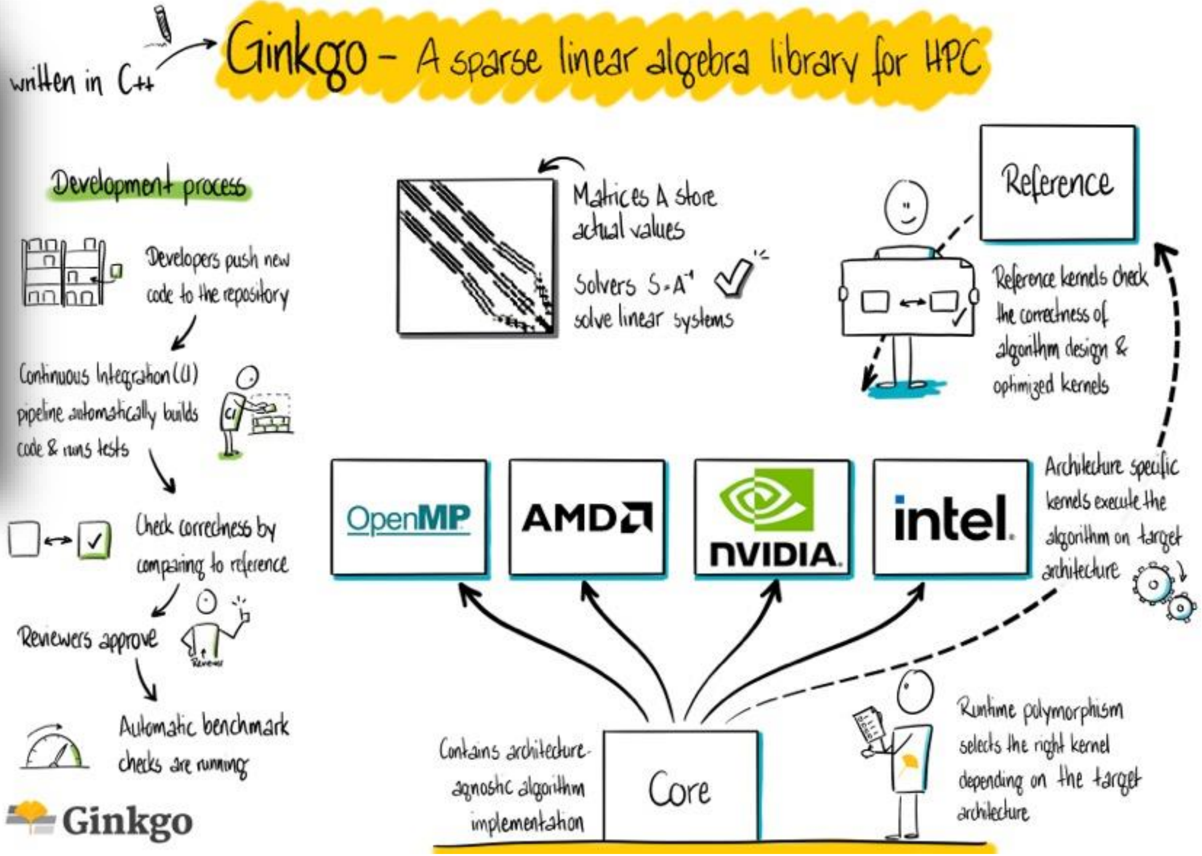




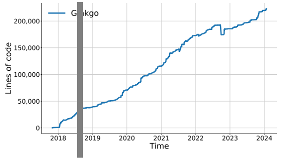
Designing a math toolset for ECP applications

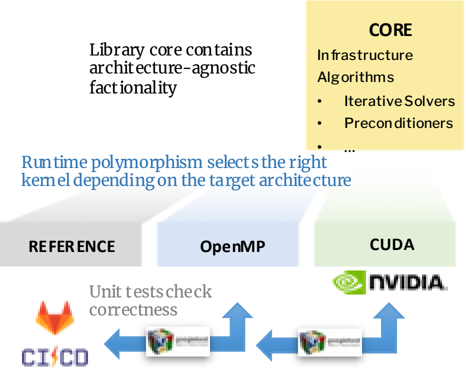






Starting with the CUDA backend





	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	✔	✔

Linear Operator Interface

We express everything as Linear Operator.

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

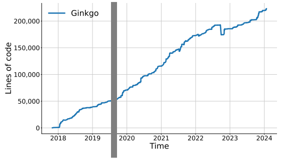
Matrix-Vector Product	Preconditioner (for matrix A)	Solver (for system $Ax = b$)
$x := A \cdot b$	$x := M^{-1} \cdot b$	$x := S \cdot b$
	$M^{-1} \approx A^{-1}$	$S \approx A^{-1}$
	$M^{-1} = \Pi(A)$	$S = \Sigma(A)$

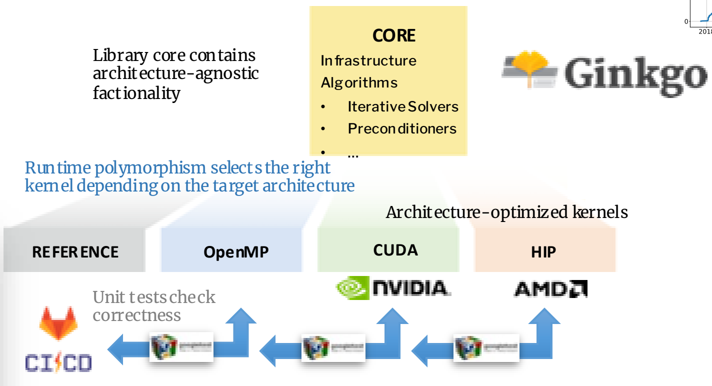
All of them can be expressed as

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

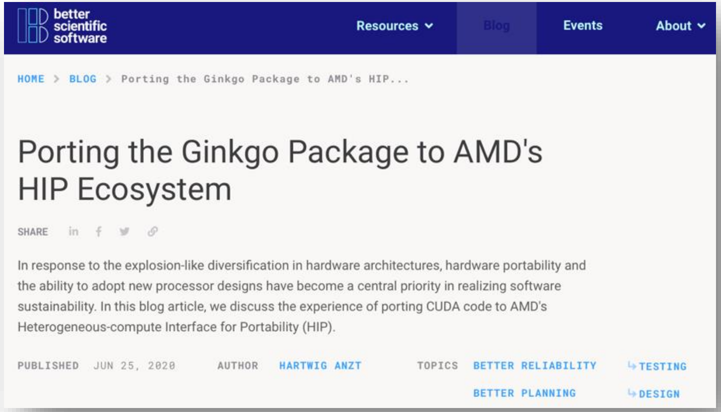
Extending to AMD GPUs

~2 months





	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	✔	✔	✔



better scientific software — Resources | Blog | Events | About

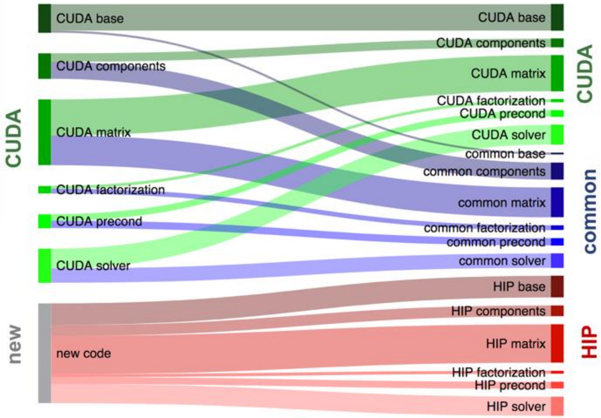
HOME > BLOG > Porting the Ginkgo Package to AMD's HIP...

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

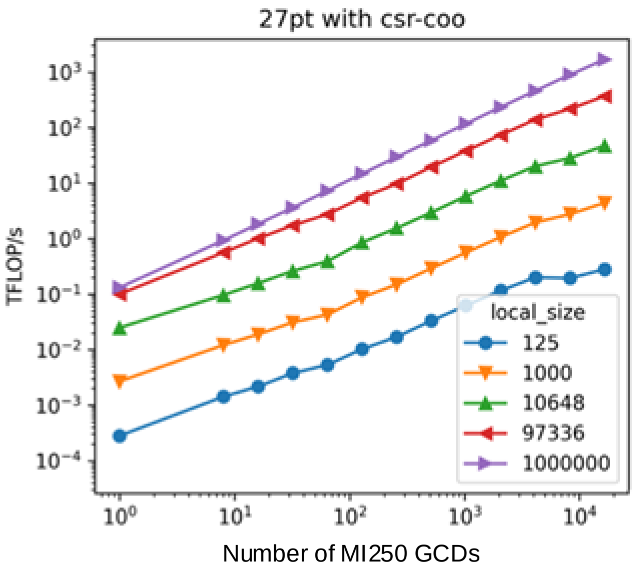


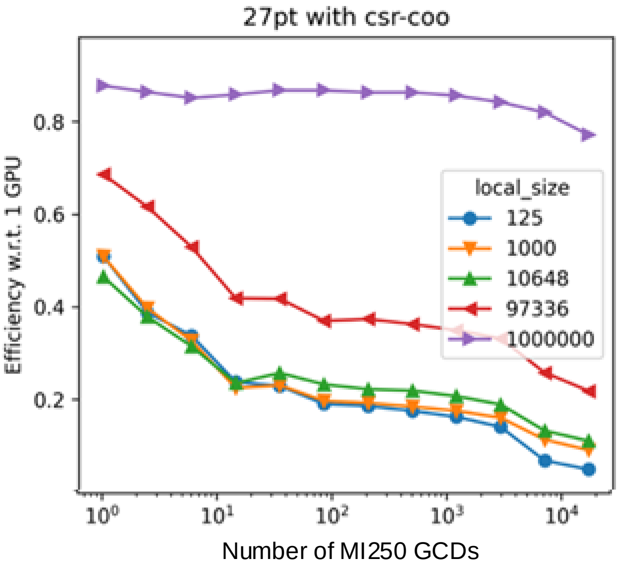
Weak and strong Scalability

Frontier (#1 TOP500)

SpMV Weak scaling: problem size increases with parallel resources

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

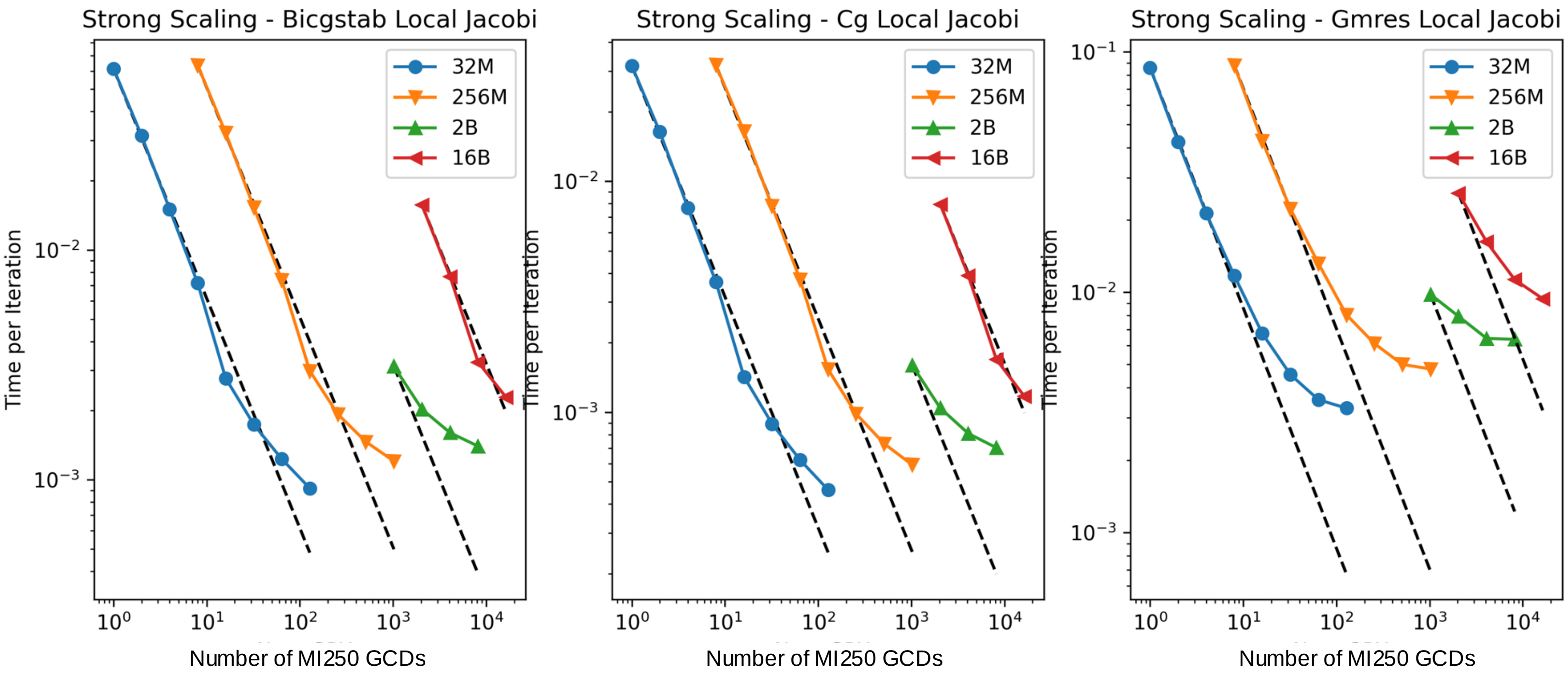




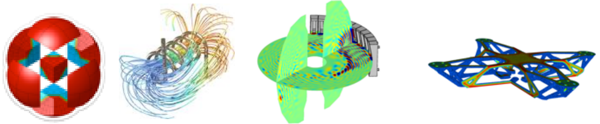
Weak and strong Scalability

Strong scaling: problem size constant, parallel resources increase

Frontier (#1 TOP500)



We “forgot” the customer on the way…



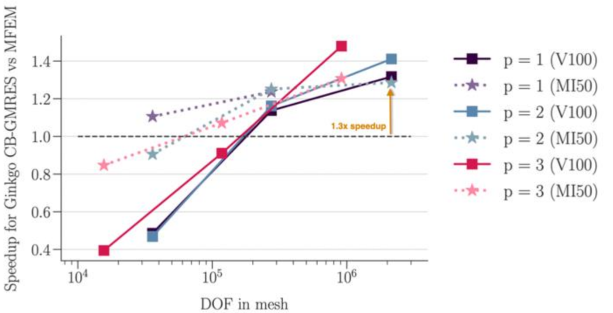
MFEM is a *free*, *lightweight*, *scalable* C++ library for finite element methods.

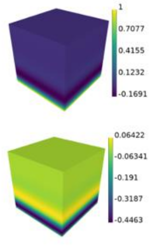
Speeding up MFEM’s “example 22” on GPUs

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:

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

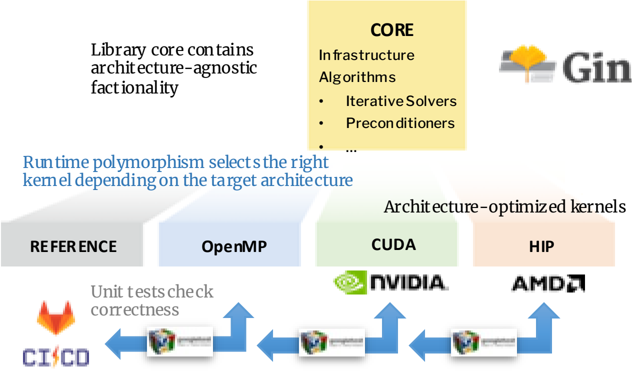
with a = 1, b = 1, ω = 10, c = 20





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.



Ginkgo

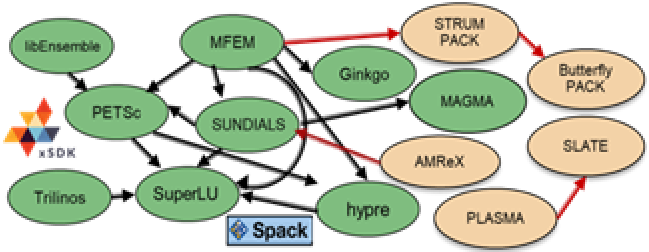
	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓

Natalie Beams

Tzanio Kolev

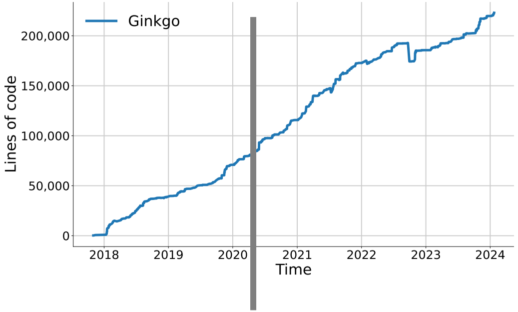
	Functionality	OMP	CUDA	HIP
Utilities	On-Device Matrix Assembly	✓	✓	✓
	MC64/RCM reordering	✓		
	Wrapping user data	✓	✓	✓
	Logging	✓	✓	✓
	PAPI counters	✓	✓	✓

Contribute and benefit from community



	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	✓	✓	✓

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.

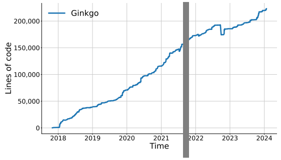


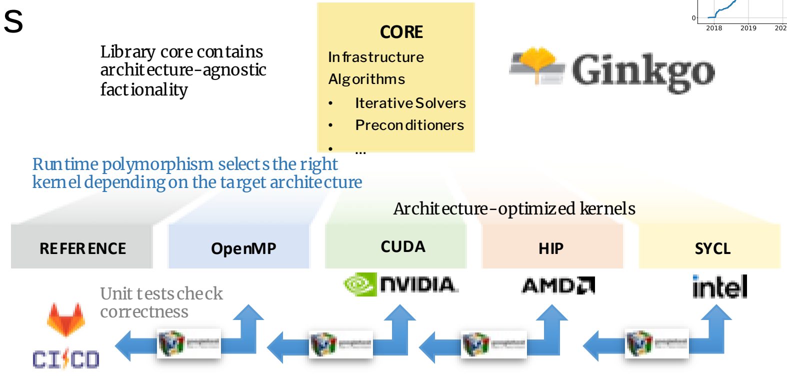
November 2022
- 26 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

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

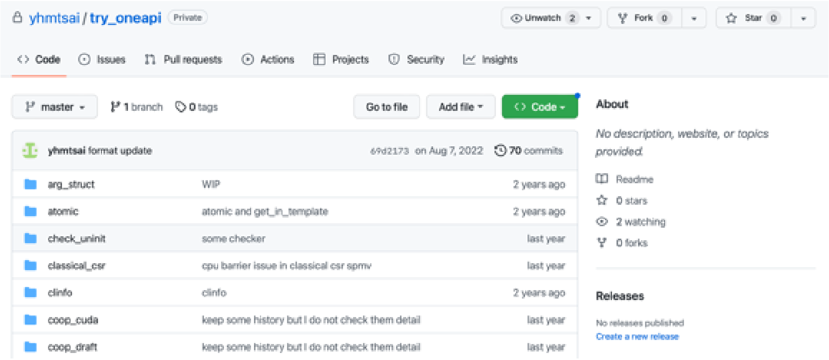
Extending to Intel GPUs

~18 months

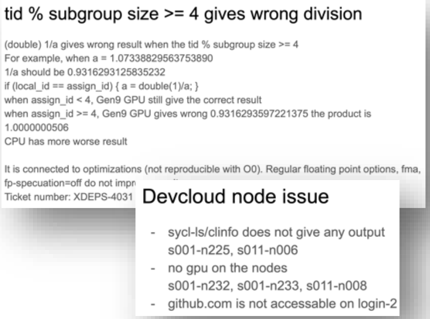




	Functionality	OMP	CUDA	HIP	DPC++
Basic	SpMV	✓	✓	✓	✓
	SpMM	✓	✓	✓	✓
	SpGeMM	✓	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓	✓
	BiCGSTAB	✓	✓	✓	✓
	CG	✓	✓	✓	✓
	CGS	✓	✓	✓	✓
	GMRES	✓	✓	✓	✓
	IDR	✓	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓	✓
	ILU/IC		✓	✓	✓
	Parallel ILU/IC	✓	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓	✓
	Sparse Approximate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



Mike Tsai



tid % subgroup size >= 4 gives wrong division

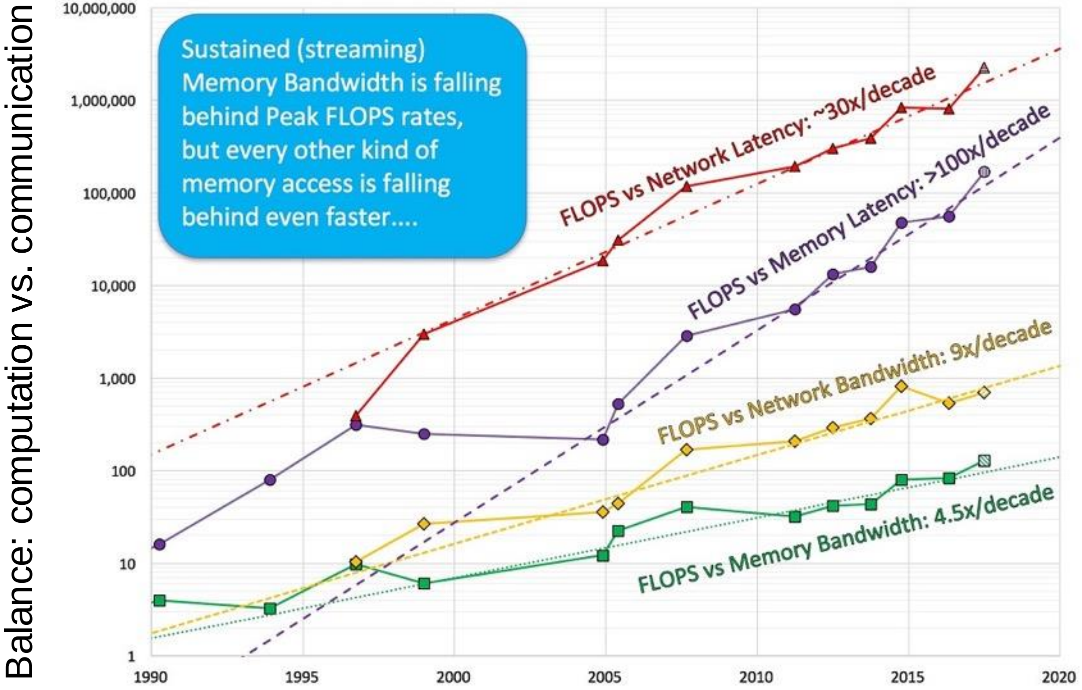
(double) 1/a gives wrong result when the tid % subgroup size >= 4
For example, when a = 1.07338829563753890
1/a should be 0.9316293125835232
if (local_id == assign_id) { a = double(1)/a; }
when assign_id < 4, Gen9 GPU still give the correct result
when assign_id >= 4, Gen9 GPU gives wrong 0.9316293597221375 the product is 1.0000000506
CPU has more worse result

It is connected to optimizations (not reproducible with O0). Regular floating point options, fma, fp-specuation=off do not impr
Ticket number: XDEPS-4031

Devcloud node issue

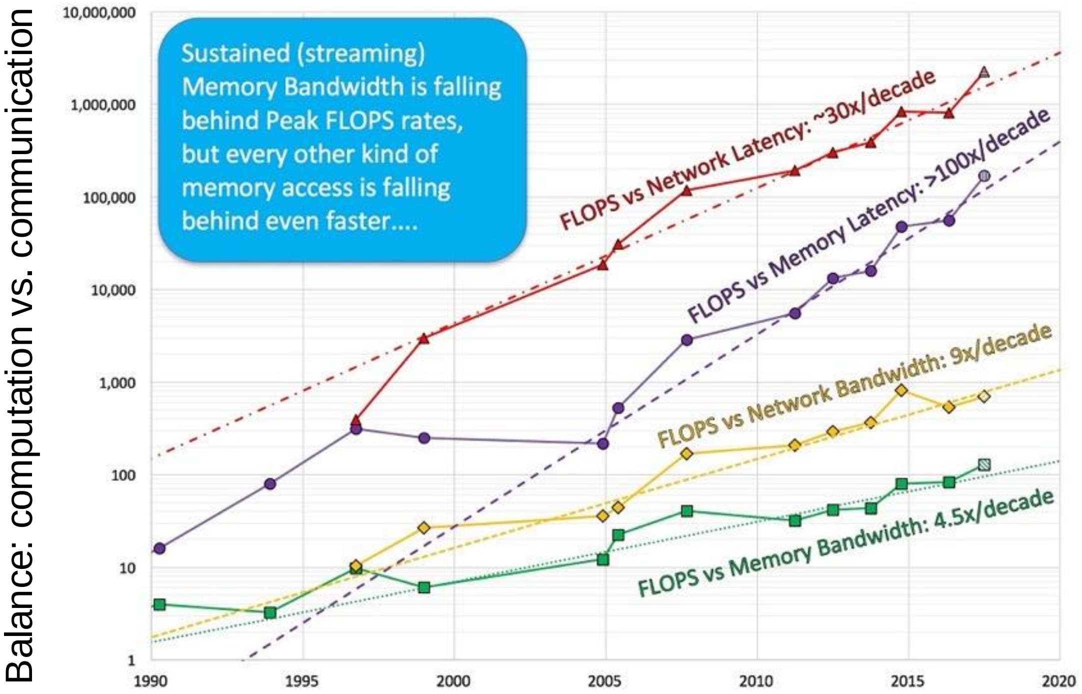
- sycl-ls/clinfo does not give any output s001-n225, s011-n006
- no gpu on the nodes s001-n232, s001-n233, s011-n008
- github.com is not accessable on login-2

Hardware Trends – ECP Mixed Precision Focus Effort



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

Hardware Trends – ECP Mixed Precision Focus Effort

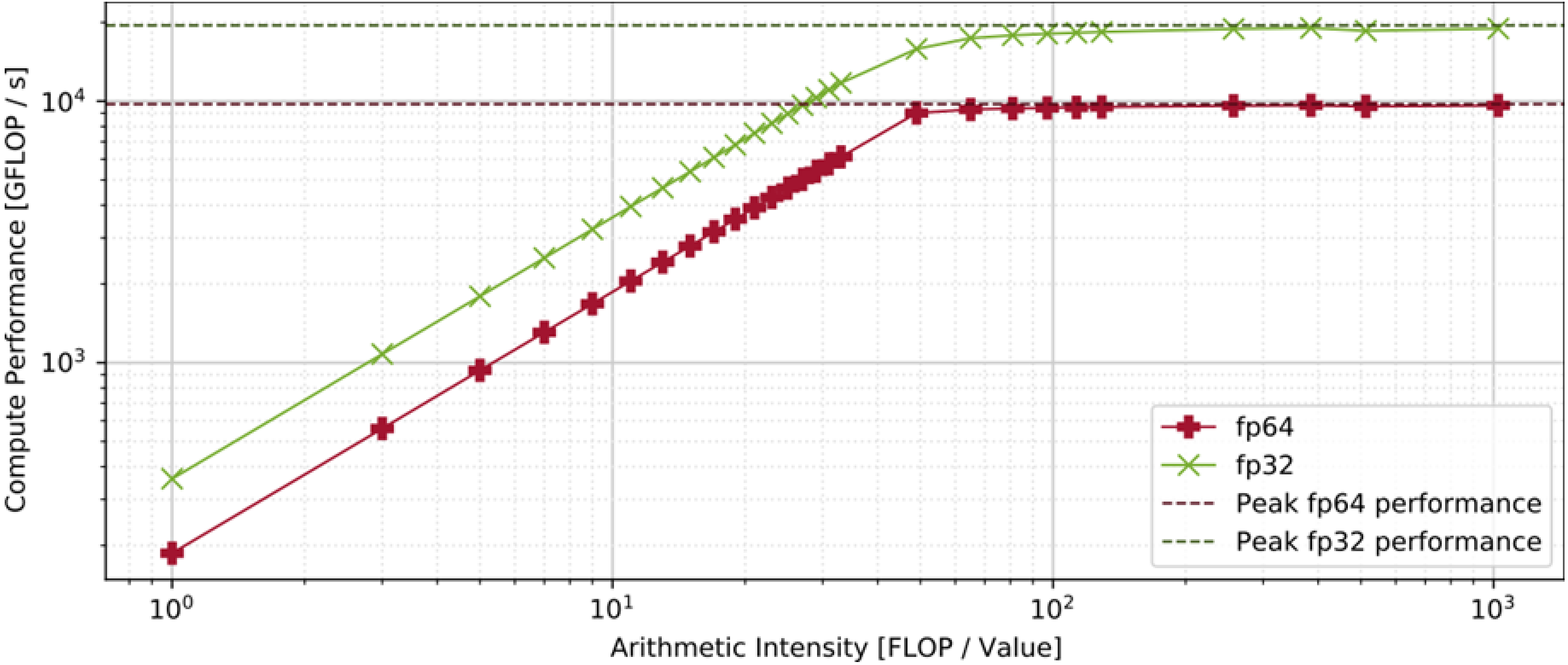


Trends in the relative performance of floating-point arithmetic and several classes of data access for select HPC servers over the past 25 years. Source: John McCalpin

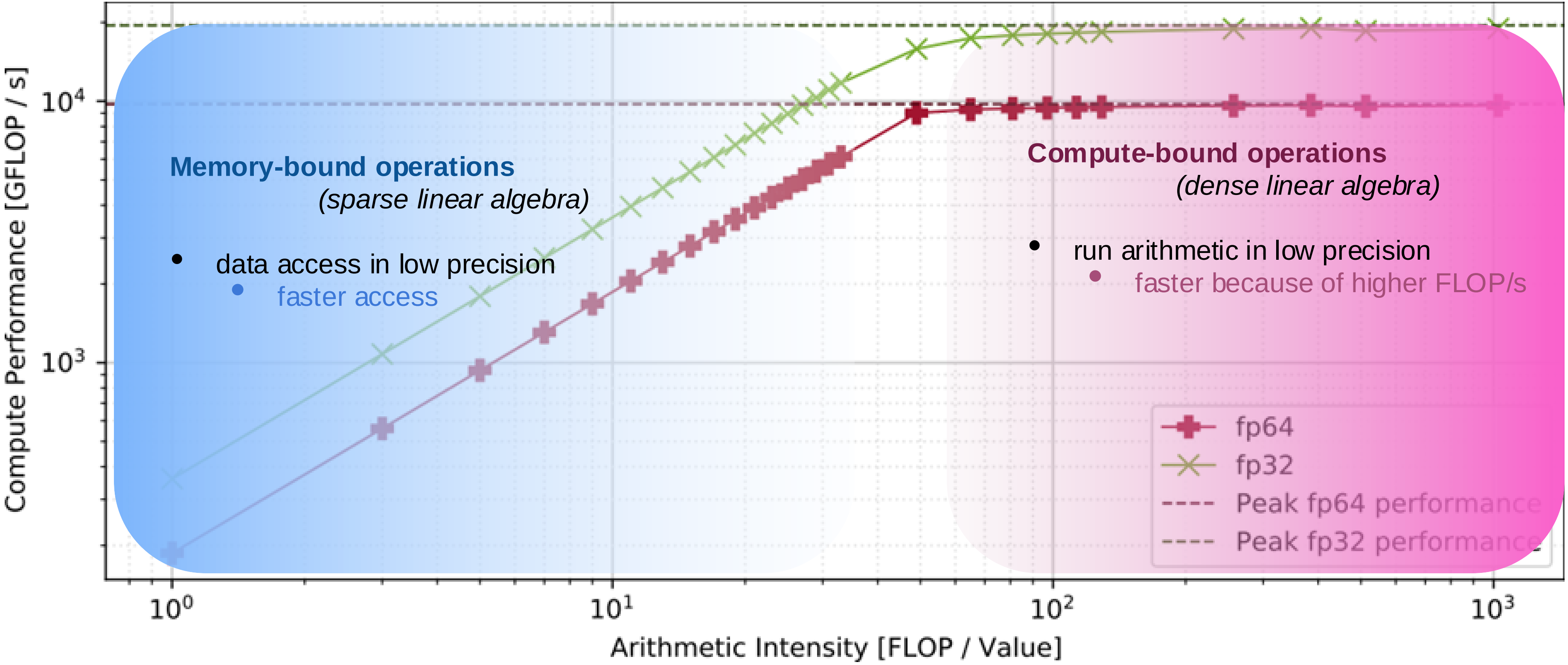
Form Factor	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core	989 teraFLOPS[2]
BFLOAT16 Tensor Core	1,979 teraFLOPS[2]
FP16 Tensor Core	1,979 teraFLOPS[2]
FP8 Tensor Core	3,958 teraFLOPS[2]
INT8 Tensor Core	3,958 TOPS[2]

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

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

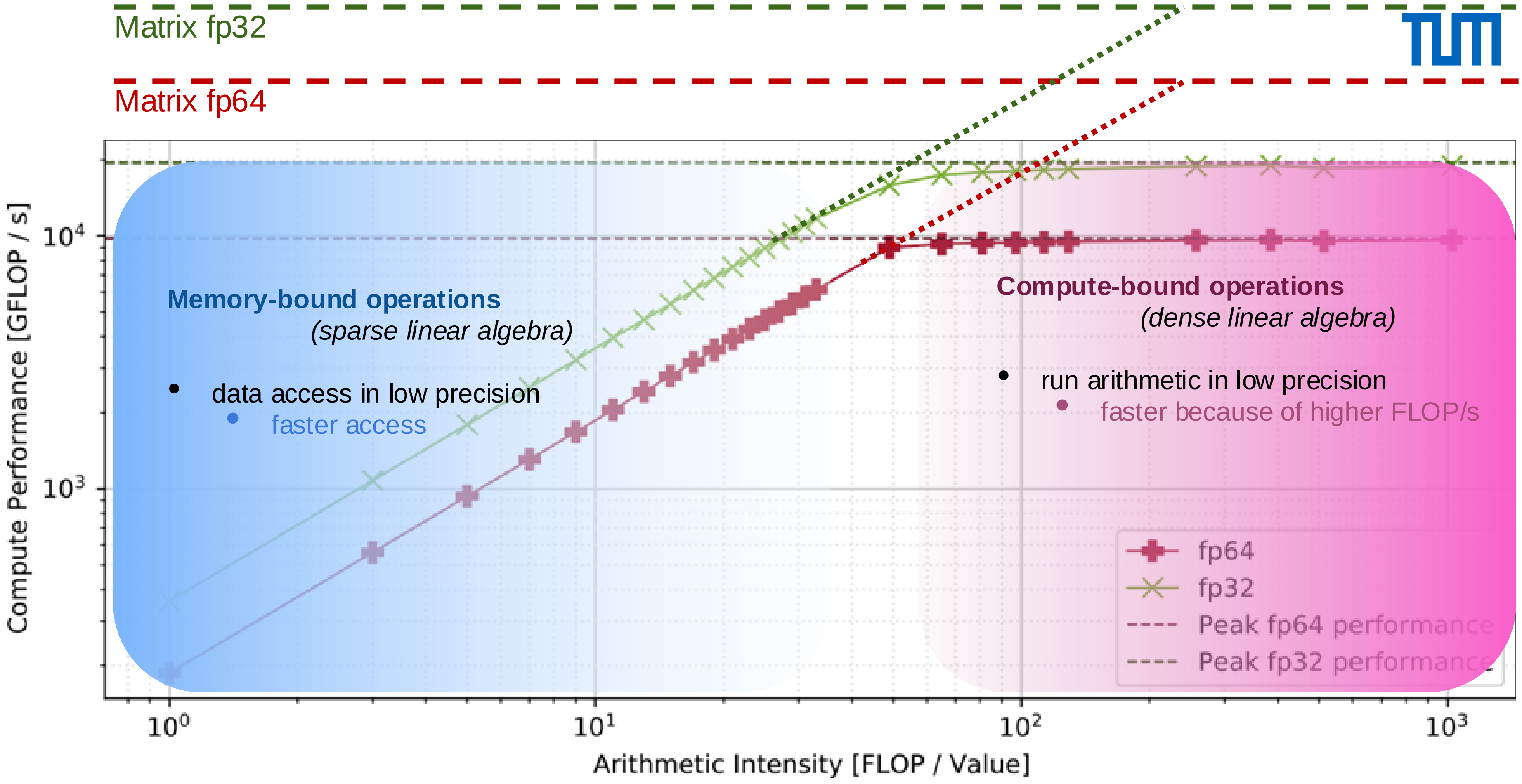


Memory-bound operations
(sparse linear algebra)

- data access in low precision
  - faster access

Compute-bound operations
(dense linear algebra)

- run arithmetic in low precision
  - faster because of higher FLOP/s



Matrix fp32
Matrix fp64
Memory-bound operations
(sparse linear algebra)
data access in low precision
faster access
Compute-bound operations
(dense linear algebra)
run arithmetic in low precision
faster because of higher FLOP/s
fp64
fp32
Peak fp64 performance
Peak fp32 performance
Compute Performance [GFLOP / s]
10^4
10^3
10^0
10^1
10^2
10^3
Arithmetic Intensity [FLOP / Value]

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

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

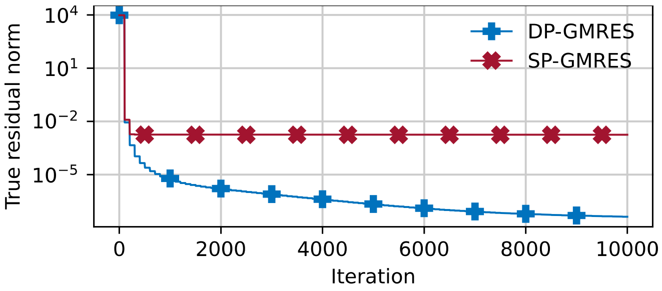
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms

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)

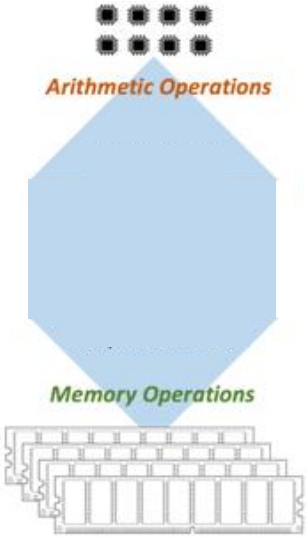
N. Higham: Accuracy and stability of numerical algorithms. SIAM, 2002.

ECP Focus Effort Mixed Precision

- Traditionally, we use a strong coupling between the

  precision formats used for arithmetic operations

  the precision format handling data in main memory.

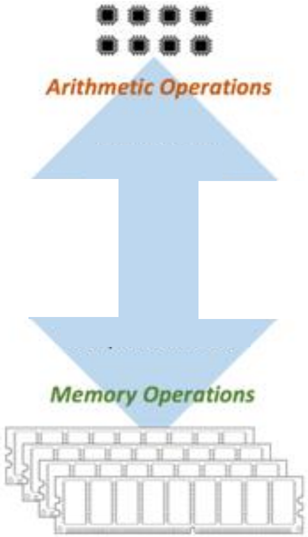


ECP Focus Effort Mixed Precision

- Traditionally, we use a strong coupling between the

  precision formats used for arithmetic operations

  the precision format handling data in main memory.



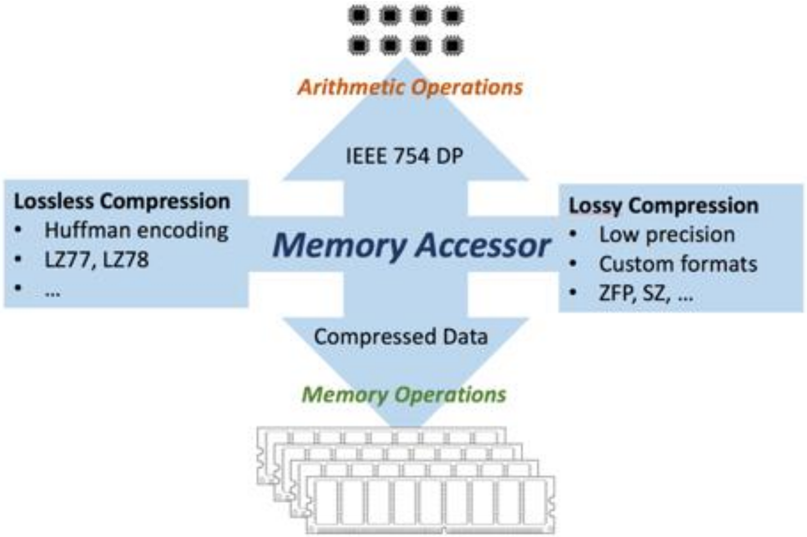
ECP Focus Effort Mixed Precision

- Traditionally, we use a strong coupling between the

  precision formats used for arithmetic operations

  the precision format handling data in main memory.

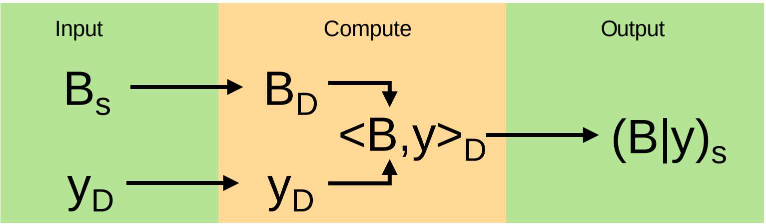
- *We should compute in fp64*
- *Data should be compressed for main memory access (low precision/compression)*
- *Compression / Conversion needs to happen on-the-fly*

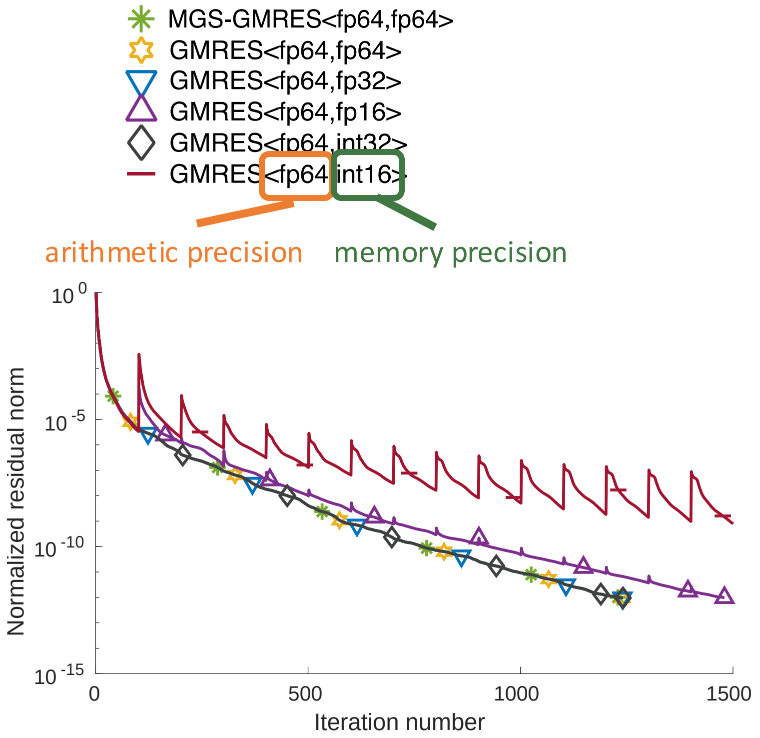


ECP Focus Effort Mixed Precision

Compressed Basis (CB-) GMRES

- Use double precision in all arithmetic operations;
- Store Krylov basis vectors **B** 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**

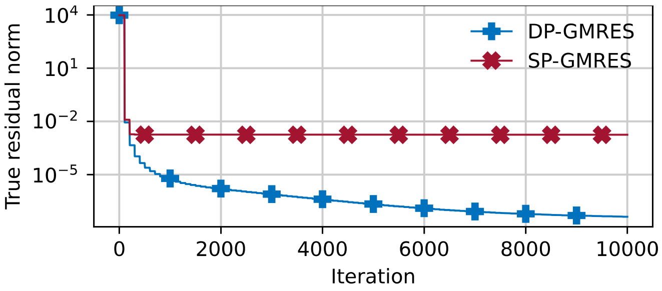
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms

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}

~2x faster!



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

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

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

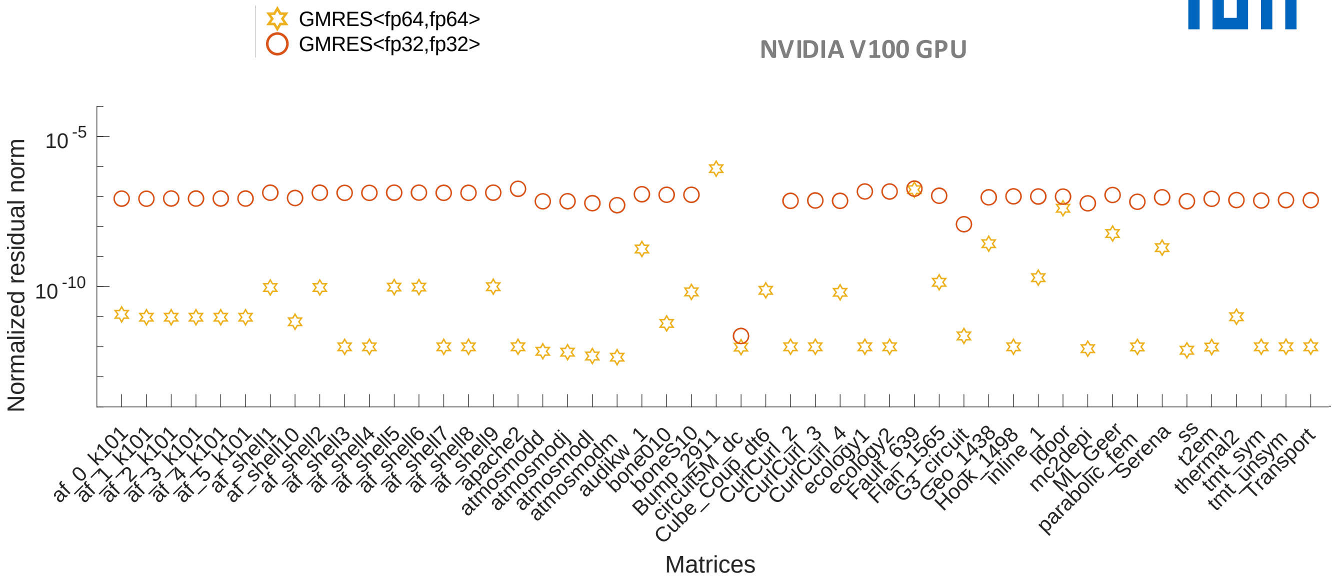
Relative residual ~10^{-7}

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}

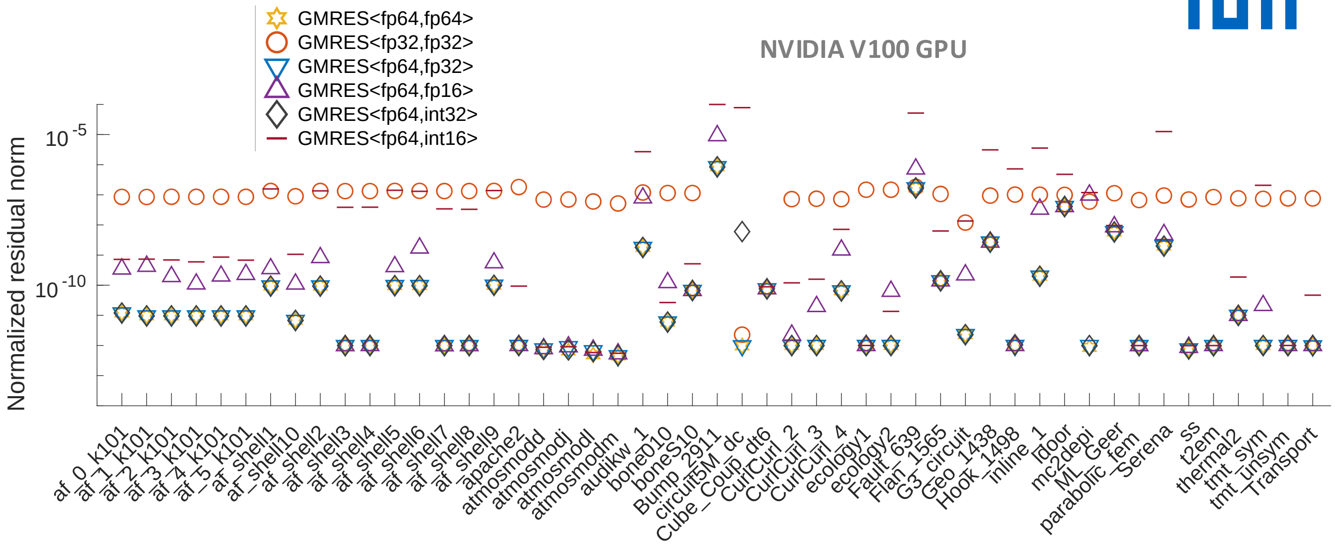
Accuracy of DP GMRES
Performance similar to SP GMRES

NVIDIA V100 GPU



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

NVIDIA V100 GPU



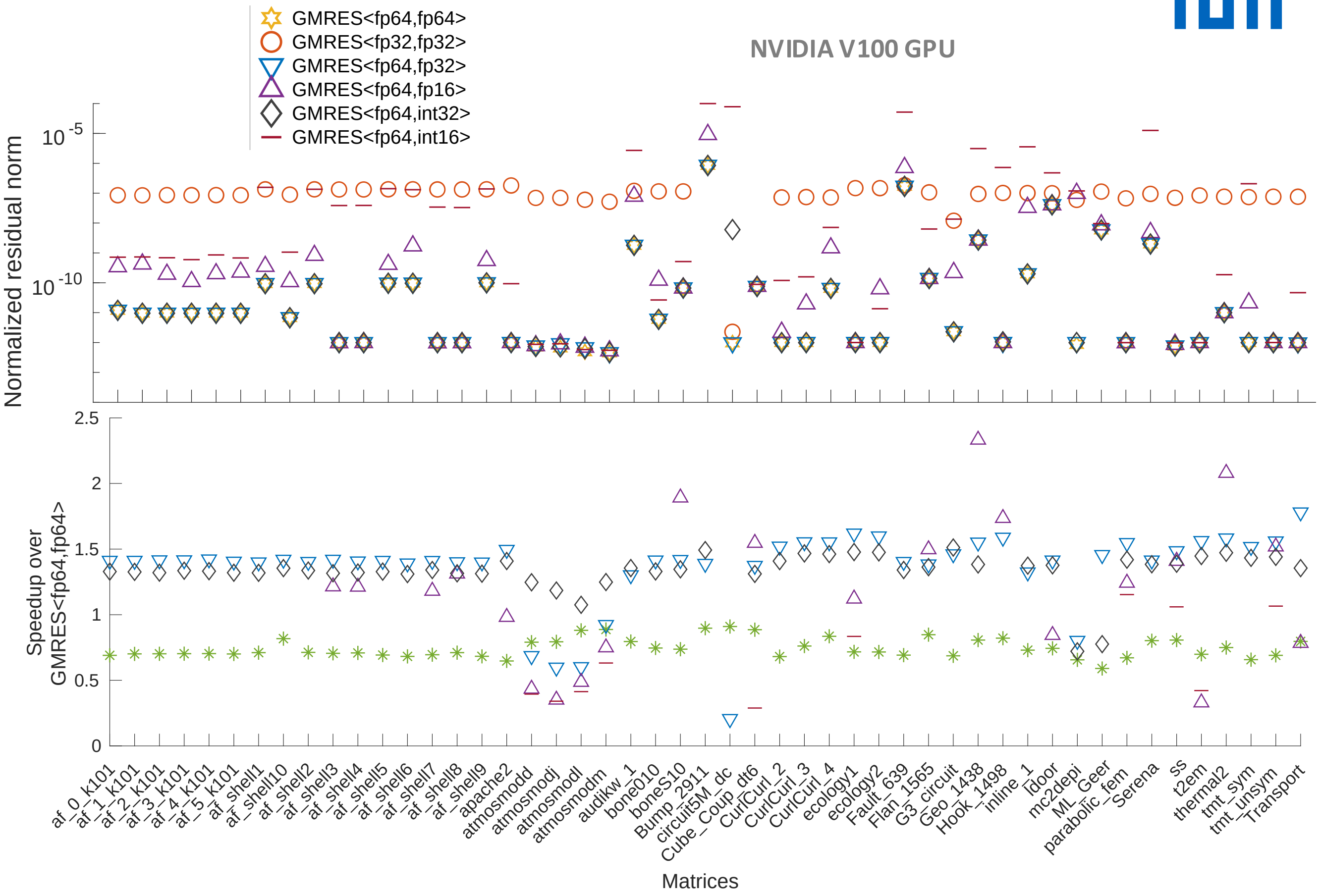
- CB-GMRES using 32-bit storage preserves DP accuracy (SP-GMRES does not)

- 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

NVIDIA V100 GPU

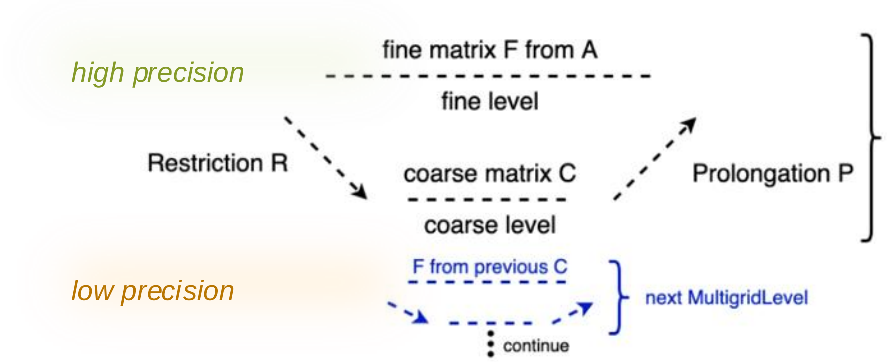


- **Preconditioning iterative solvers**
    - Idea: Approximate inverse of system matrix to make the system “easier to solve”:

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

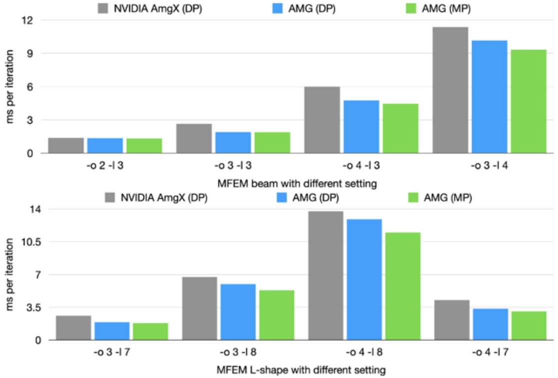
- **Mixed Precision Multigrid Preconditioner**

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



Ginkgo

Mike Tsai

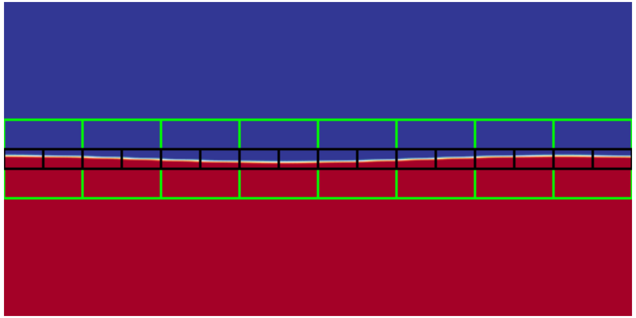


Application Needs – ECP Batched Focus Effort

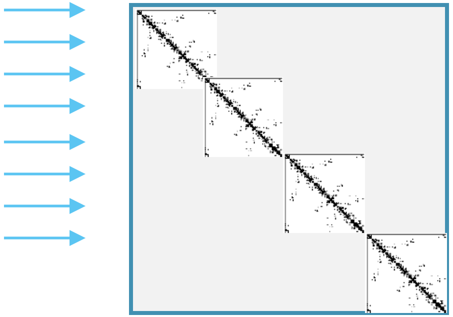
Batched iterative solvers for SUNDIALS / PeleLM

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)

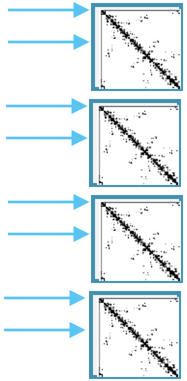
https://amrex-combustion.github.io/PeleLM/overview.html

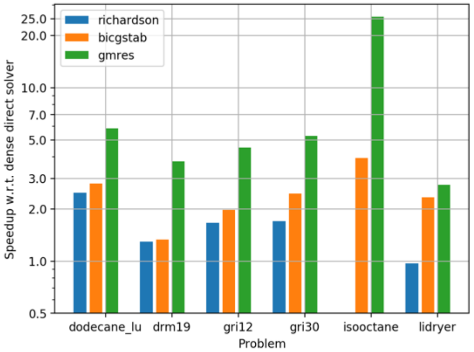


Problem	Size	Non-zeros (A)	Non-zeros (L+U)
dodecane_lu	54	2,332 (80%)	2,754 (94%)
drm19	22	438 (90%)	442 (91%)
gri12	33	978 (90%)	1,018 (93%)
gri30	54	2,560 (88%)	2,860 (98%)
isooctane	144	6,135 (30%)	20,307 (98%)
lidryer	10	91 (91%)	91 (91%)

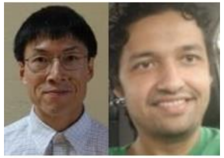


Carol Woodward   Cody Balos





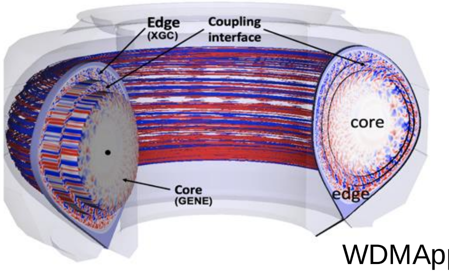
Batched Functionality for the Collision Operator



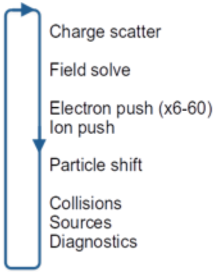
Paul Lin   Dhruva Kulkarni

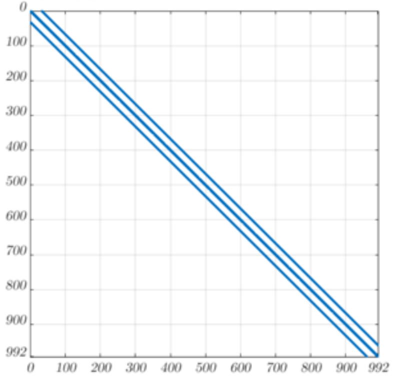
<u>XGC</u> is a gyrokinetic particle-in-cell code, which specializes in the simulation of the edge region of magnetically confined thermonuclear fusion plasma. The simulation domain can include the magnetic separatrix, magnetic axis and the biased material wall. XGC can run in total-delta-f, and conventional 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

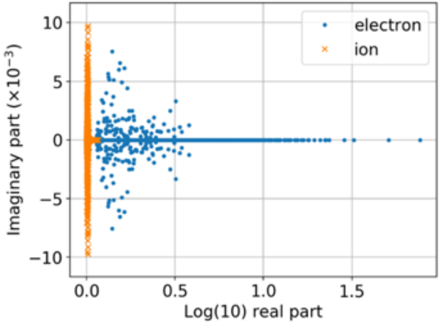


WDMApp





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

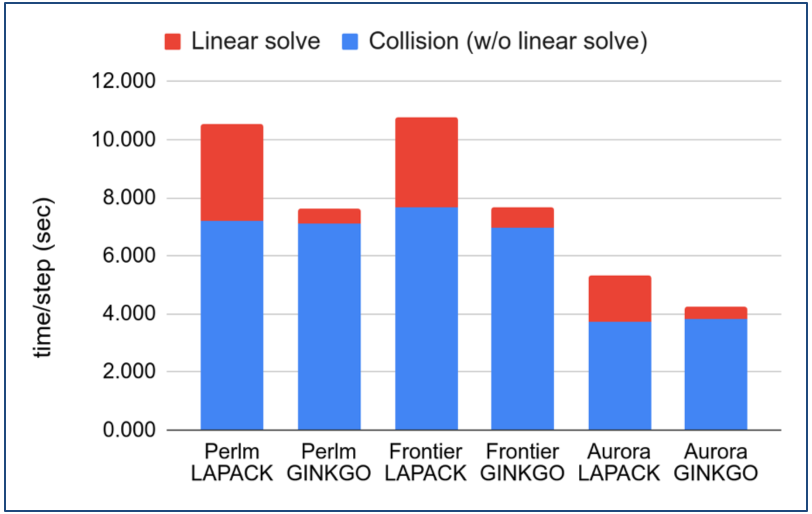


XGC collision operator: fully nonlinear multi-species Fokker-Planck-Landau
For each mesh vertex:

- Outer nonlinear solver: Picard method with inner linear solves
- Linear solve: discretize velocity space with approx 35x35 velocity grid
- direct solve on CPU using LAPACK banded solver **dgbsv**
- After GPU porting of XGC, this is the remaining CPU intensive kernel for collision operator

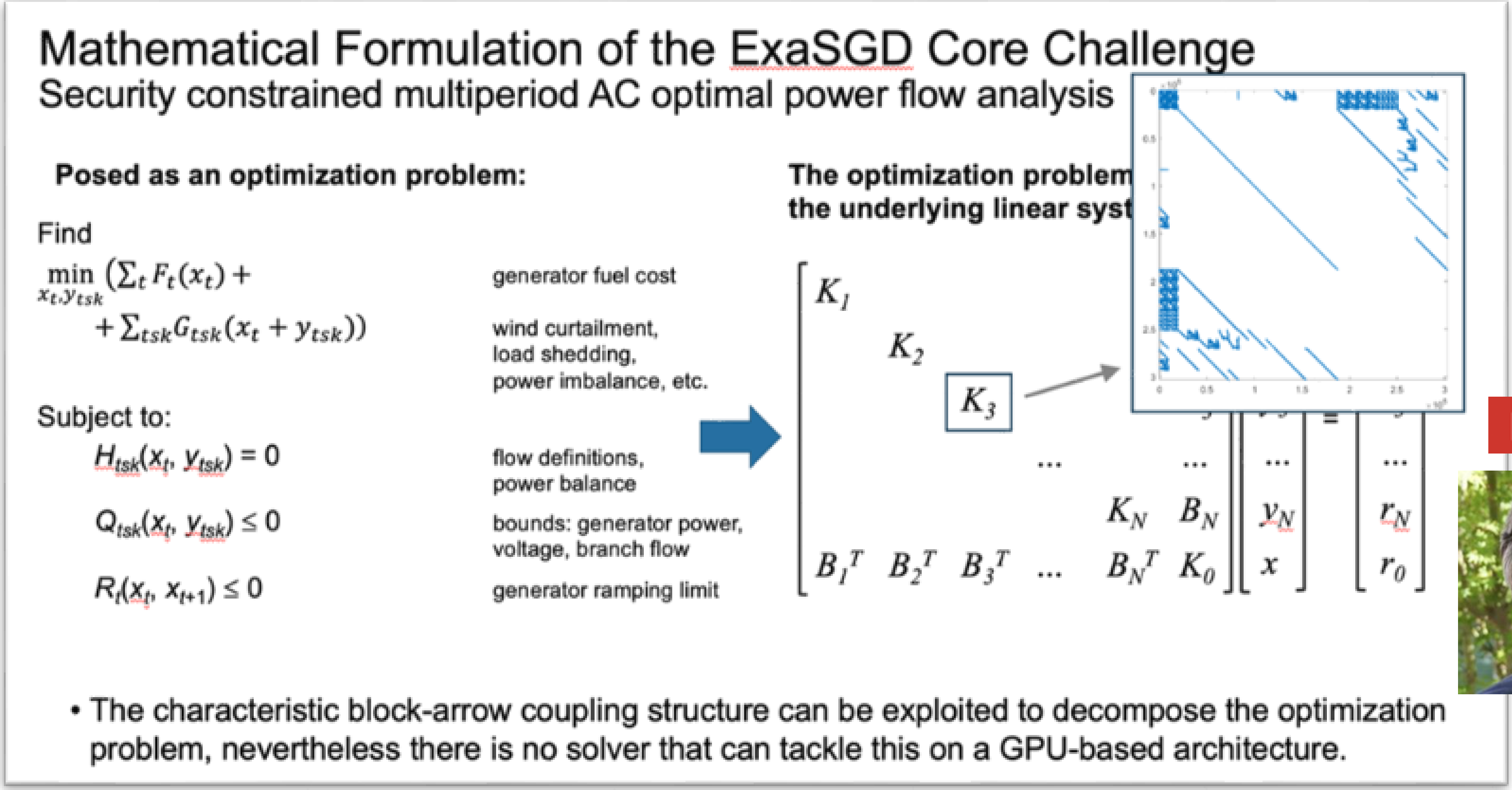
Batched Functionality for the Collision Operator

- XGC DIII-D National Fusion Facility tokamak electromagnetic (EM) test case
- 8 nodes of NERSC Perlmutter: 32 A100s, 1 MPI per GPU; single socket 64-core AMD EPYC
- 8 nodes OLCF Frontier: 32 MI250X, 64 GCDs, 1 MPI per GCD; single socket 64-core AMD EPYC
- 8 nodes ALCF Aurora: 48 Intel Data Center Max 1550, 96 tiles, 1 MPI per tile; dual socket 52-core Intel CPU Max 9470C SPR



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.

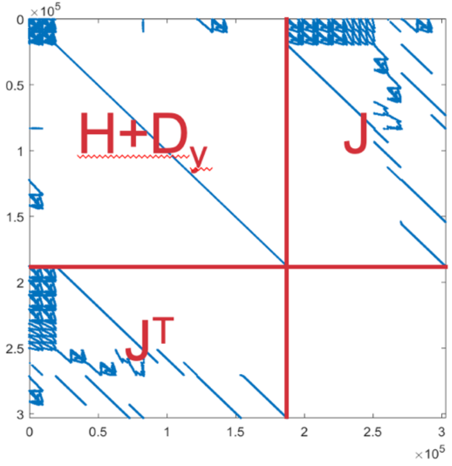
Power Grid Simulations



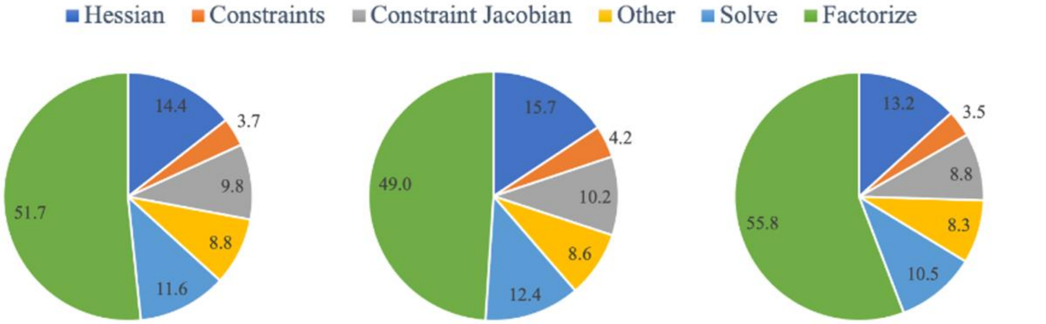
EXASGD

© Slaven Peles

Sparse Direct Solvers

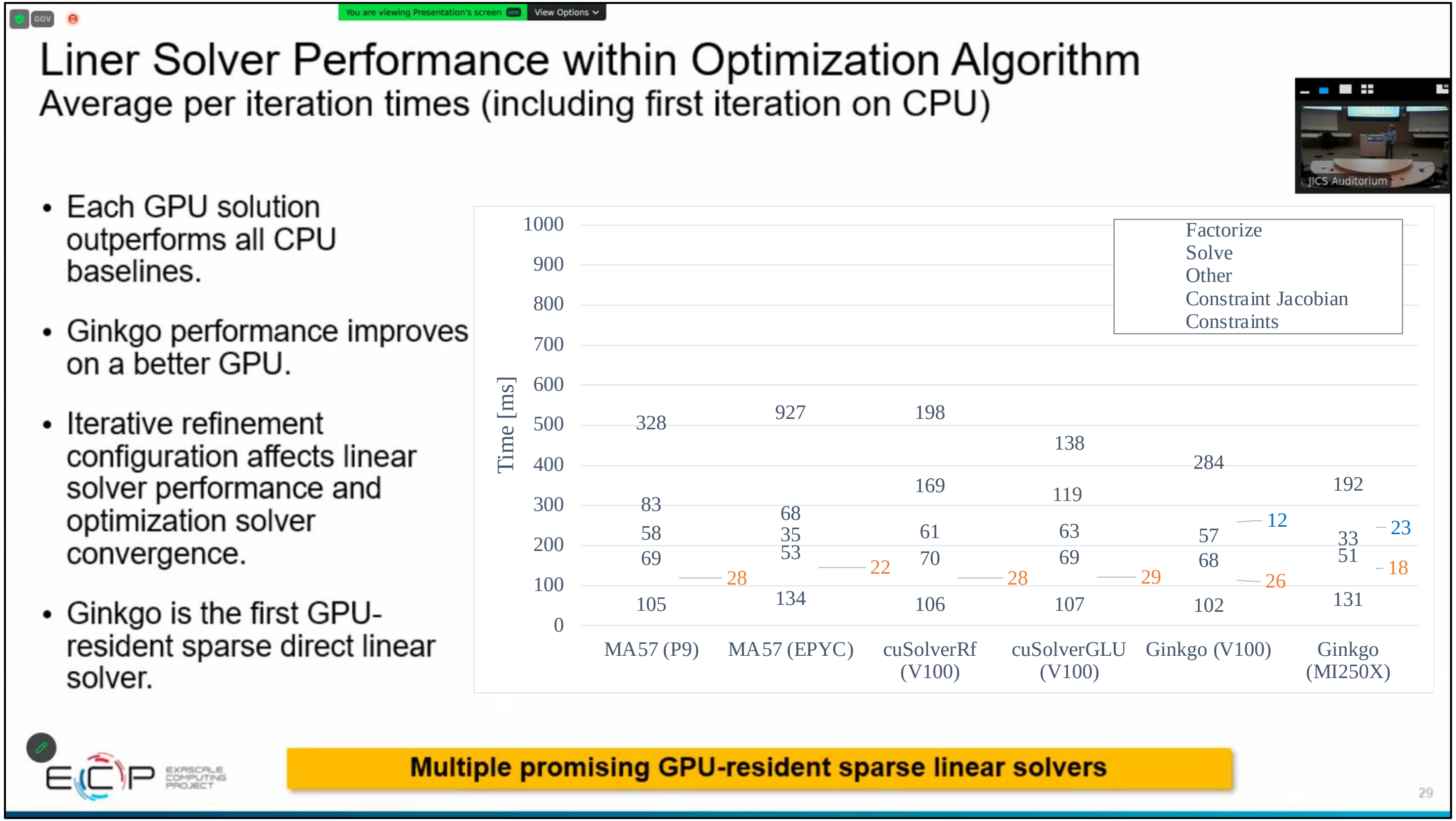


Grid	Buses	Generators	Lines	$N(K_k)$	$\text{nnz}(K_k)$
Northeastern US	25 K	4.8 K	32.3 K	108 K	1.19 M
Eastern US	70 K	10.4 K	88.2 K	296 K	3.20 M
Western and Eastern US	82 K	13.4 K	104.1 K	340 K	3.73 M

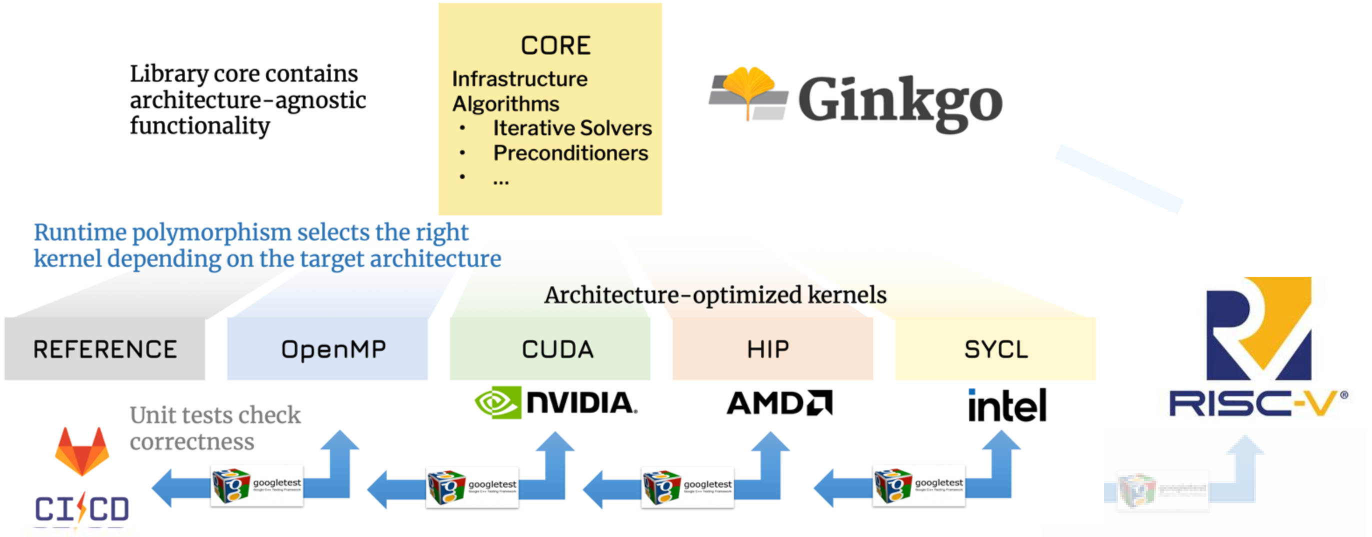


(a) Northeast U.S. grid (b) Eastern U.S. grid (c) Eastern and Western U.S. grids

Sparse Direct Solvers



After 6 years of development



	FUNCTIONALITY	OMP	CUDA	HIP	DPC++
Basic	SpMV	☑	☑	☑	☑
	SpMM	☑	☑	☑	☑
	SpGeMM	✔	✔	✔	✔
Krylov solvers	BiCG	✔	✔	✔	✔
	BiCGSTAB	☑	☑	☑	☑
	CG	☑	☑	☑	☑
	CGS	✔	✔	✔	✔
	GCR	☑	☑	☑	☑
	GMRES	☑	☑	☑	☑
	FCG	☑	☑	☑	☑
	FGMRES	☑	☑	☑	☑
	IR	☑	☑	☑	☑
	IDR	✔	✔	✔	✔
Preconditioners	Block-Jacobi	☑	☑	☑	☑
	ILU/IC	✔	✔	✔	✔
	Parallel ILU/IC	✔	✔	✔	✔
	Parallel ILUT/ICT	✔	✔	✔	✔
	ISAI	✔	✔	✔	✔
Batched	Batched BiCGSTAB	☑	☑	☑	☑
	Batched CG	☑	☑	☑	☑
	Batched GMRES	☑	☑	☑	☑
	Batched ILU	☑	☑	☑	☑
	Batched ISAI	☑	☑	☑	☑
	Batched Block-Jacobi	☑	☑	☑	☑
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	☑	☑	☑	☑
	Logging	✔	✔	✔	✔
	PAPI counters	✔	✔	✔	✔

☑ MPI Support ✔ Single-GPU Support

Lessons learnt

- **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.**

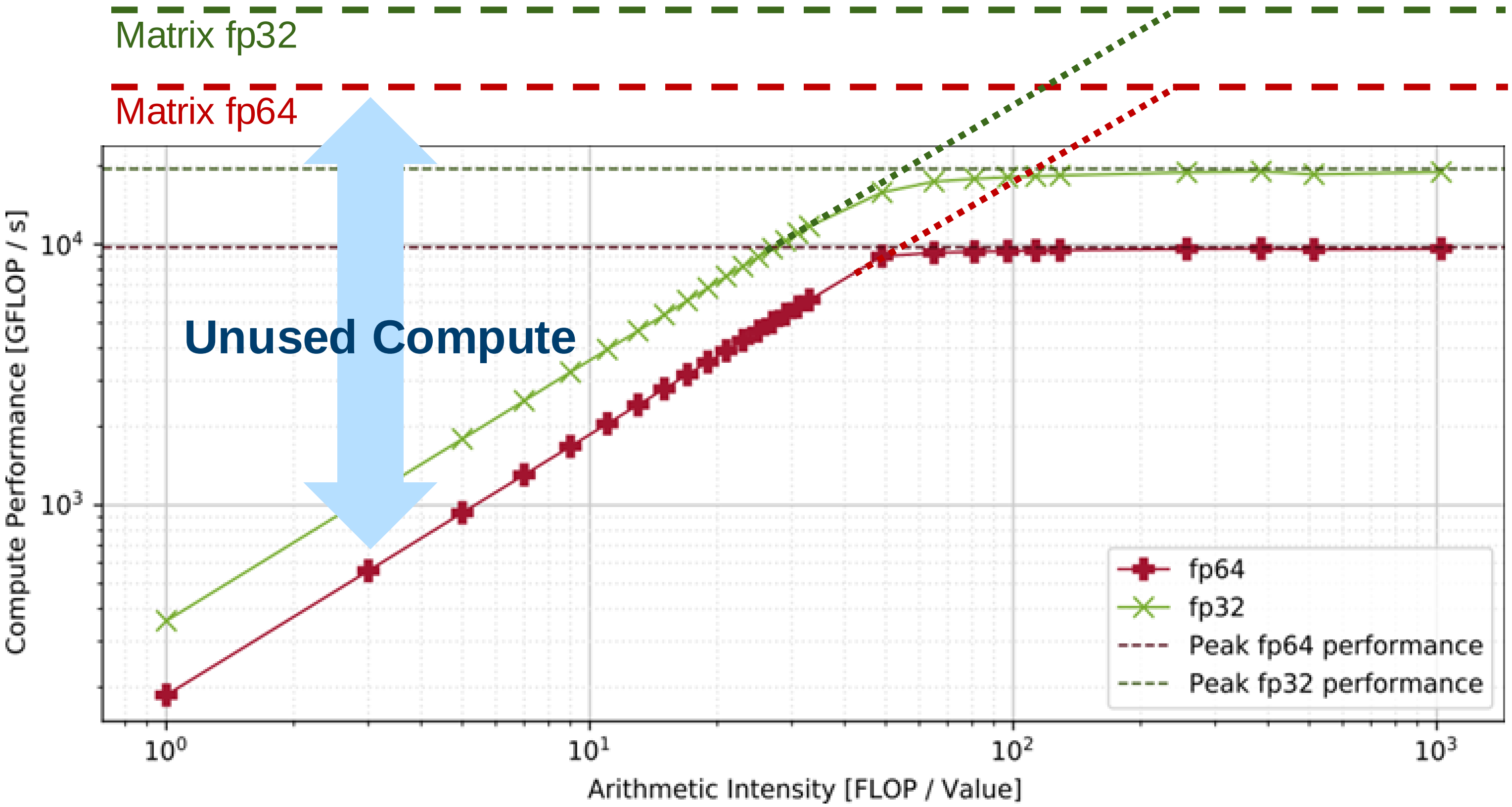
- **Anticipating the future in hardware development accelerates the porting process.**
  - **Blueprints** and **early access systems** both useful.
  - **Interaction with industry** is mutually beneficial.

- **Strategic initiatives, interaction and collegial behavior are important.**
  - **Strategic focus groups, conferences,** and **meetings** bring experts together and **create collaboration**.
  - **Listen to the application needs. Value input and acknowledge collaborators.**

Lessons learnt

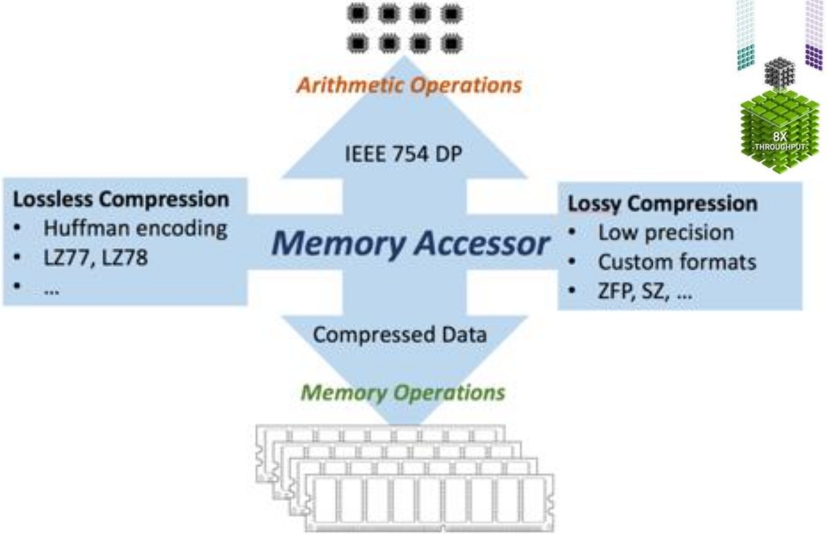
- **AI is dominating the hardware market.**
  - **Low precision will become the major format supported by hardware** (16 bit? 8bit?).
  - We need likely have to **emulate higher precision** if we require it in applications.
- **Bandwidth and Latency can not keep up with growth in compute power.**
  - **We need use compression on all levels** – likely hierarchically combined.
  - Optimizations need to focus on replacing communication with repetitive computation.
- **Resilience has become more a “secret discussion” than a showstopper.**
  - We know that MTF rates are in the hours for some machines.
  - Often, the topic is silenced for political reason.
- **AI is here to stay.**
  - **AI is an alternative to classical simulations and can enhance classical simulations**.
  - Like classical simulations, AI needs highly optimized matrix and vector operations, communication, and algorithms.

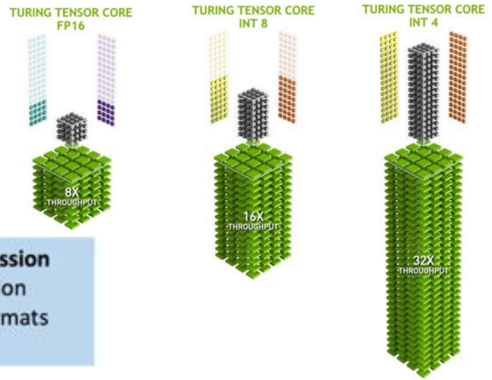
Activities going forward: There's plenty of room at the Top



Activities going forward

Using the Tensor cores for better in-register compression

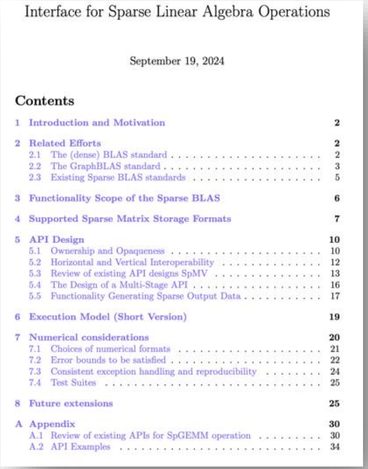




Activities going forward

Sparse BLAS working group

- On the path to defining a standard for sparse BLAS operations

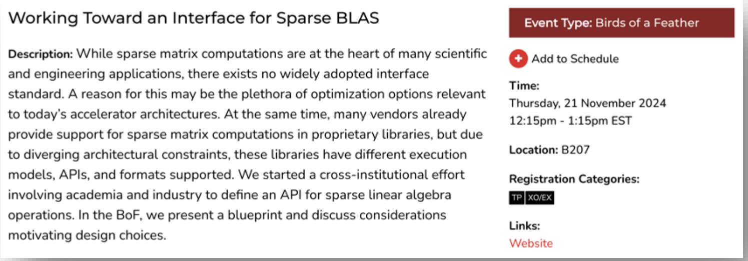

Interface for Sparse Linear Algebra Operations

September 19, 2024

Contents

1st Sparse BLAS workshop, 2023

2nd Sparse BLAS workshop, 2024


Working Toward an Interface for Sparse BLAS

Description: While sparse matrix computations are at the heart of many scientific and engineering applications, there exists no widely adopted interface standard. A reason for this may be the plethora of optimization options relevant to today's accelerator architectures. At the same time, many vendors already provide support for sparse matrix computations in proprietary libraries, but due to diverging architectural constraints, these libraries have different execution models, APIs, and formats supported. We started a cross-institutional effort involving academia and industry to define an API for sparse linear algebra operations. In the BoF, we present a blueprint and discuss considerations motivating design choices.

Event Type: Birds of a Feather

Add to Schedule

Time:
Thursday, 21 November 2024
12:15pm - 1:15pm EST

Location: B207

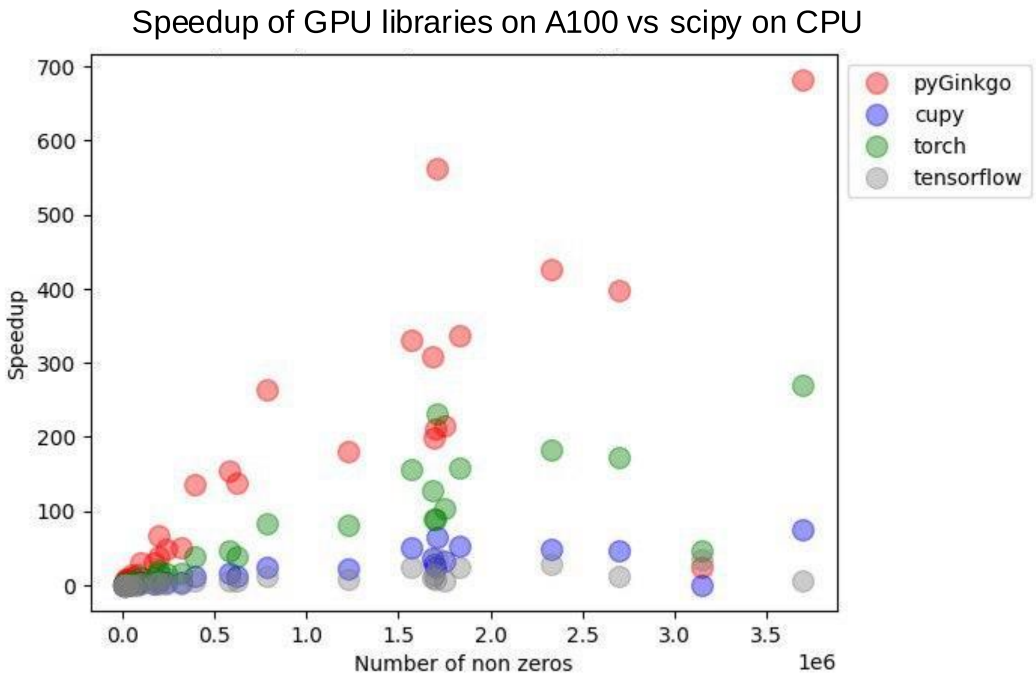
Registration Categories:
TP XO/EX

Links:
Website

Intel, NVIDIA, AMD, IBM, EVIDEN, Arm, MathWorks,
LLNL, LBNL, SNL, MIT, UC Berkeley, UTK, KIT, TUM

Activities going forward

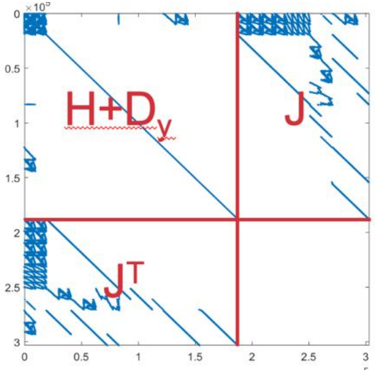
Linear algebra functionality for AI software stacks



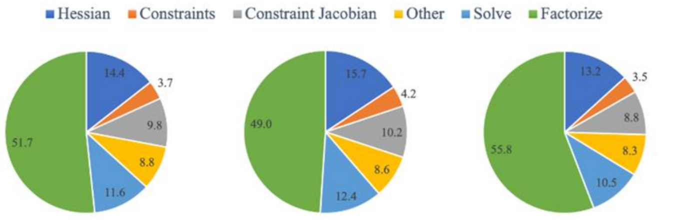
Activities going forward

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

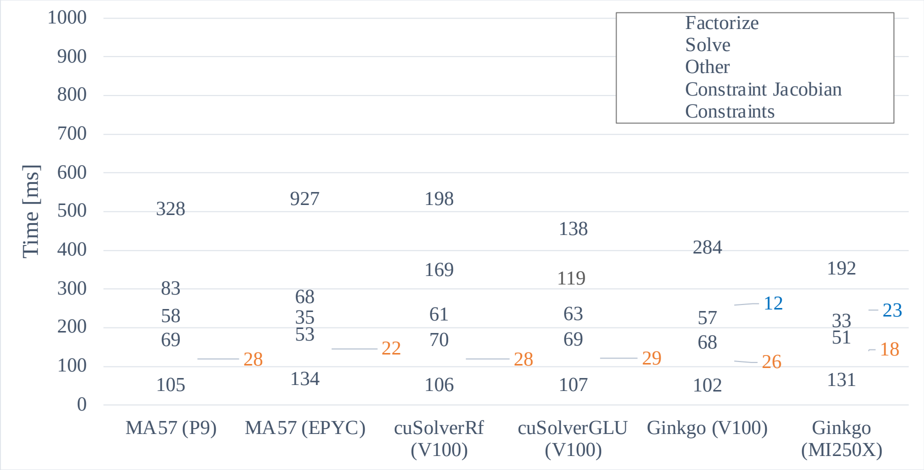
- J – sparse constraints Jacobian,
- H – sparse Hessian,
- D_y – arises from log-barrier function



Grid	Buses	Generators	Lines	$N(K_k)$	$\mathrm{nnz}(K_k)$
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



Activities going forward

EuroHPC
Joint Undertaking

MICROCARD

- *Cardiac disease is the #1 cause of death in Europe and half of these deaths are caused by electrical malfunctions.*
- *Structural muscle damage is crucial in most of these.*