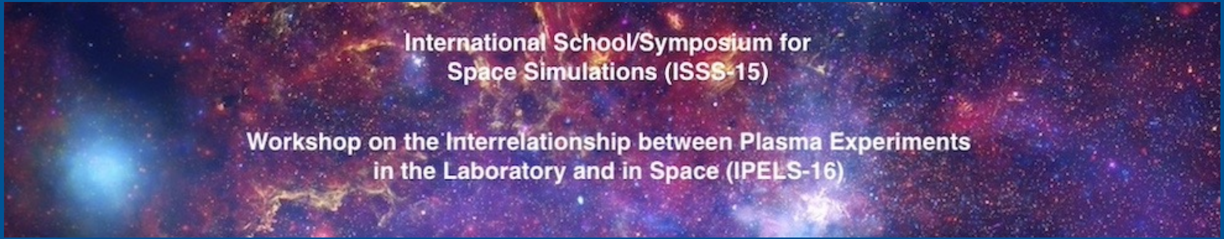
TUM School of Computation, Information and Technology
Technische Universität München

Accelerating Fusion Plasma Collision Operator Solves with Portable, Batched Iterative Solvers on GPUs

How to make Research Software Faster Better Harder Stronger - Lessons learnt from the US Exascale Computing Project

Hartwig Anzt



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).

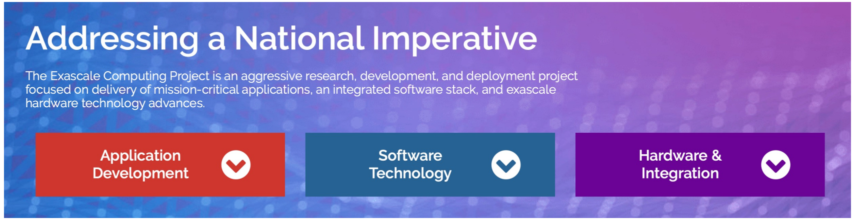
Strengthening National Security

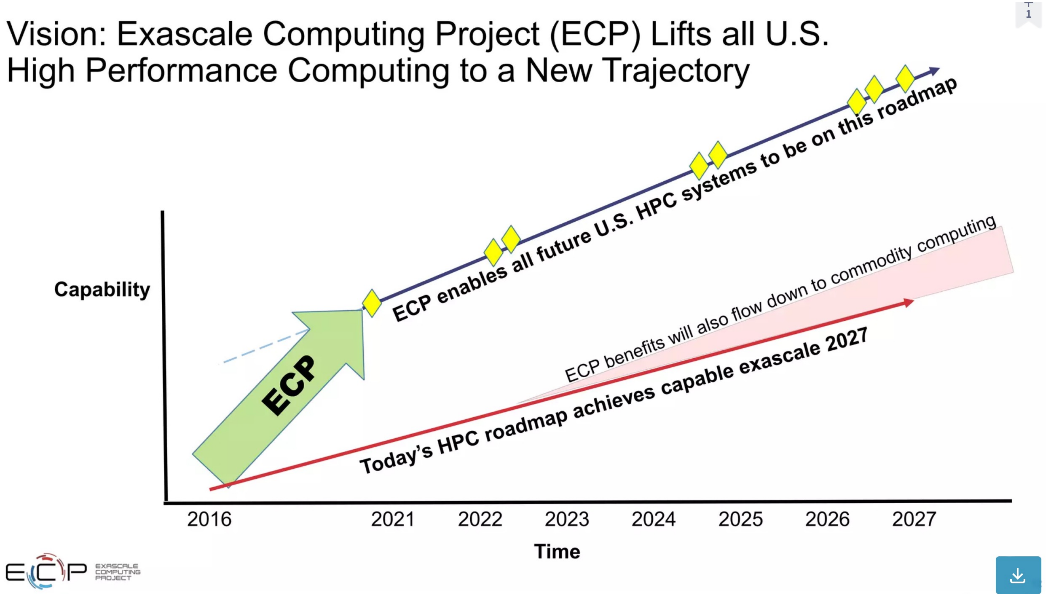
The ECP teams are also developing new applications for supporting the NNSA Stockpile Stewardship Program, which is responsible for maintaining the readiness and reliability of our nuclear weapons systems—without underground testing. Assessing the performance of weapons systems subject to hostile environments 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.

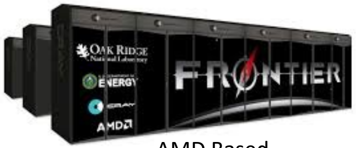
The US Exascale Computing Project





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

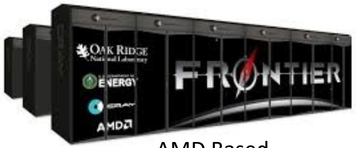
AMD APU Based
(panned)

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

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
(panned)

- **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**

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 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)*

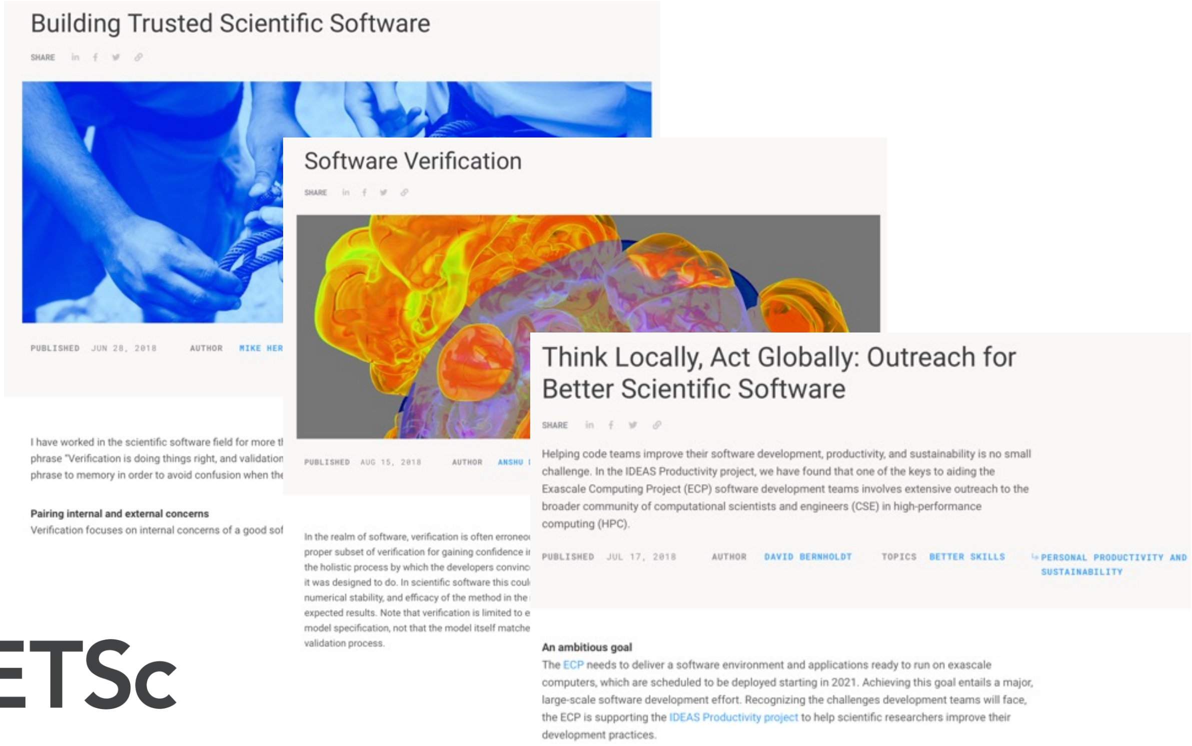
Learn from your peers…

IDEAS productivity

better scientific software

TRILINOS

PETSc



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

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

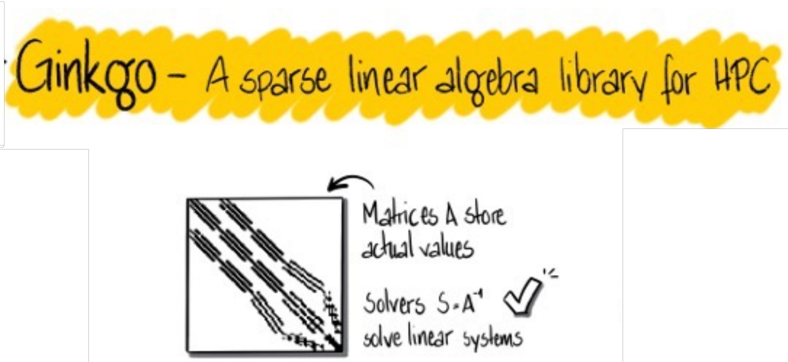
Helping code teams improve their software development, productivity, and sustainability is no small challenge. In the IDEAS Productivity project, we have found that one of the keys to aiding the Exascale Computing Project (ECP) software development teams involves extensive outreach to the broader community 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

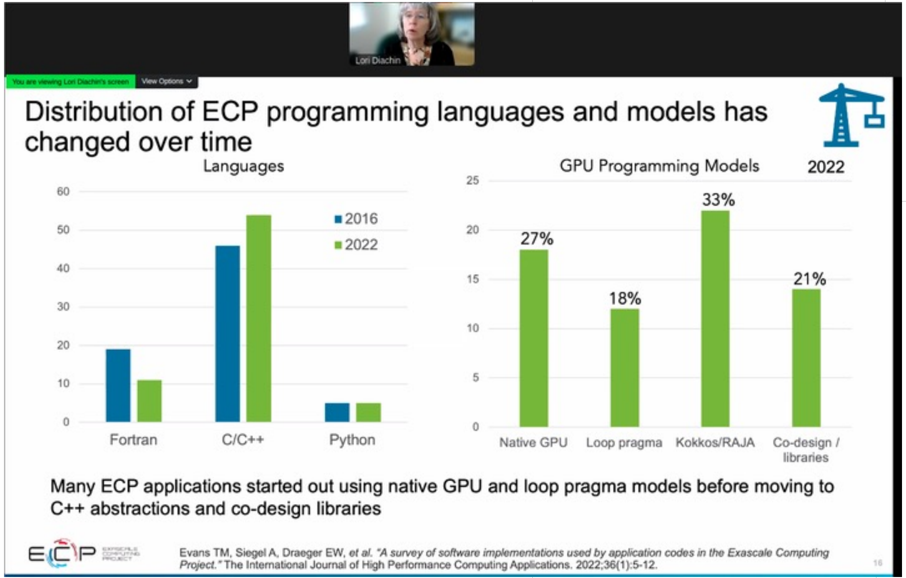
The ECP needs to deliver a software environment and applications ready to run on exascale computers, which are scheduled to be deployed starting in 2021. Achieving this goal entails a major, large-scale software development effort. Recognizing the challenges development teams will face, the ECP is supporting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

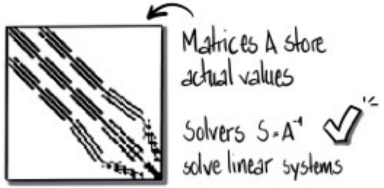
Designing a math toolset for ECP applications



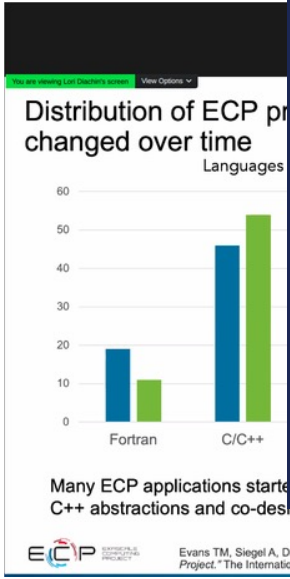
Designing a math toolset for ECP applications

written in C++ Ginkgo – A sparse linear algebra library for HPC





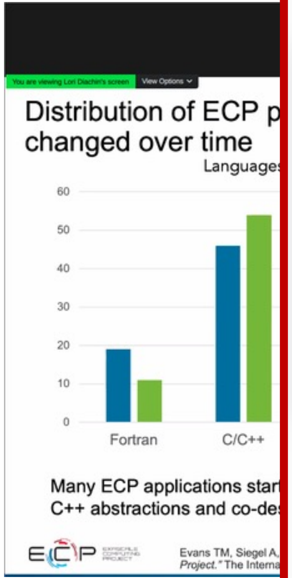
Designing a math toolset for ECP applications



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

written in C++ → Ginkgo – A sparse linear algebra library for HPC



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 Notic
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

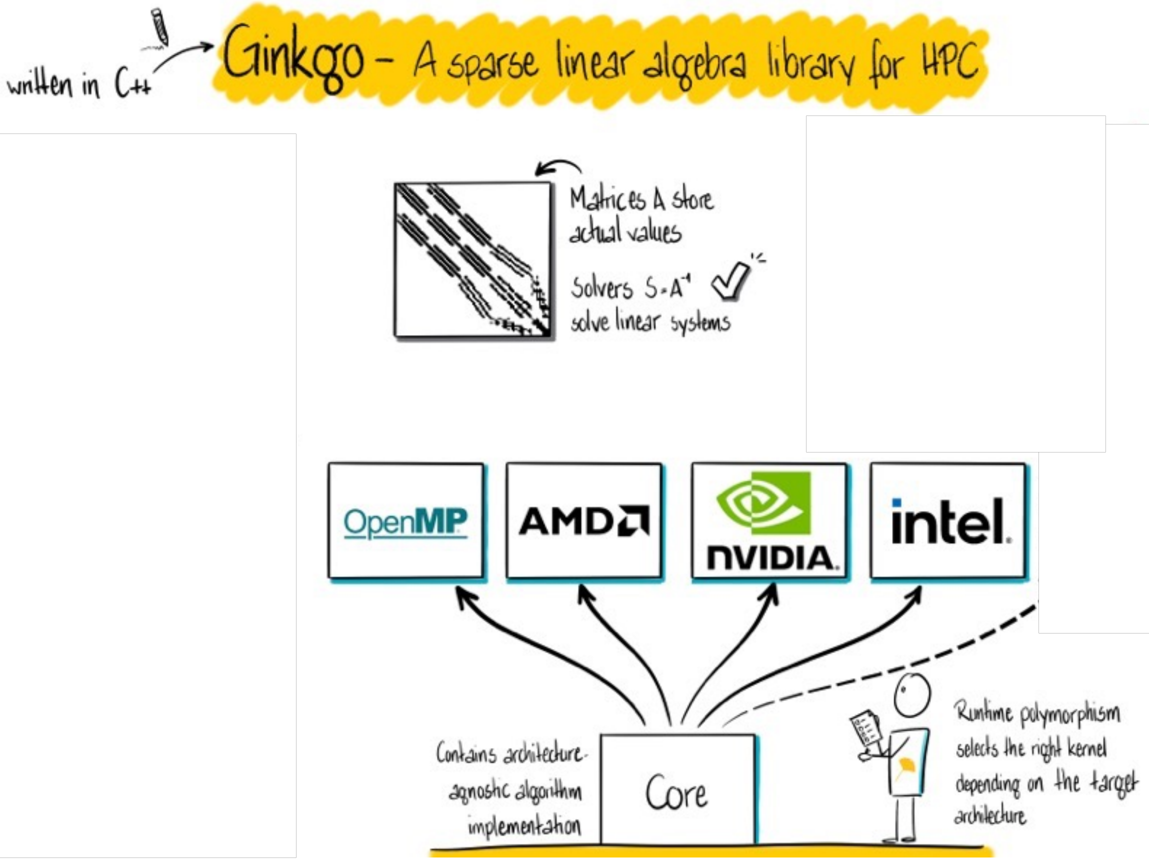
Matrices A store actual values

Solvers $S \approx A^{-1}$ solve linear systems

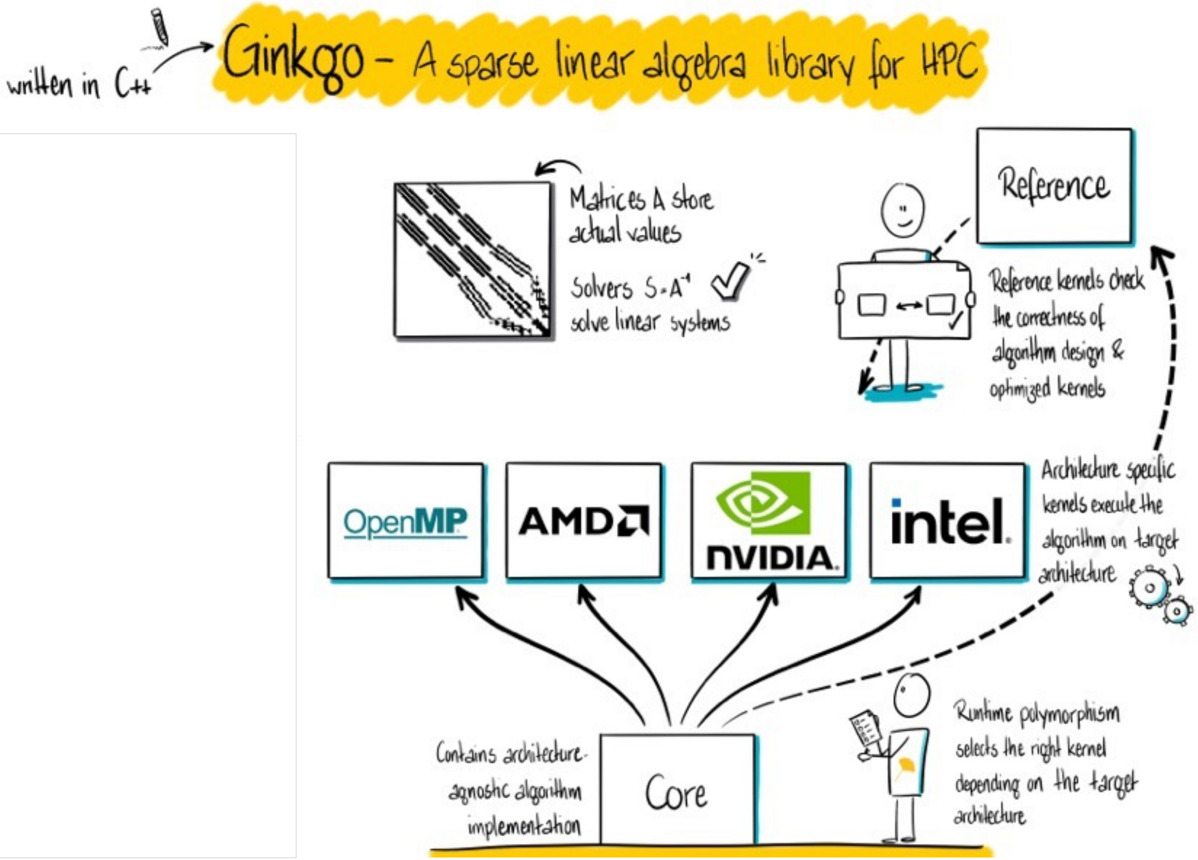
Description

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.

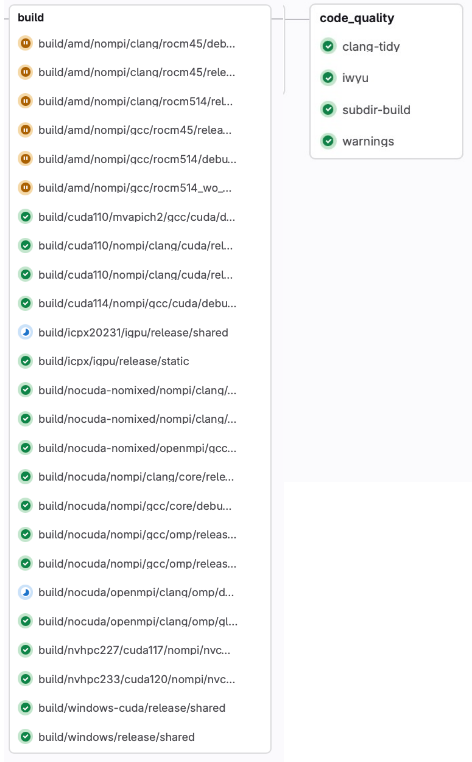
Designing a math toolset for ECP applications

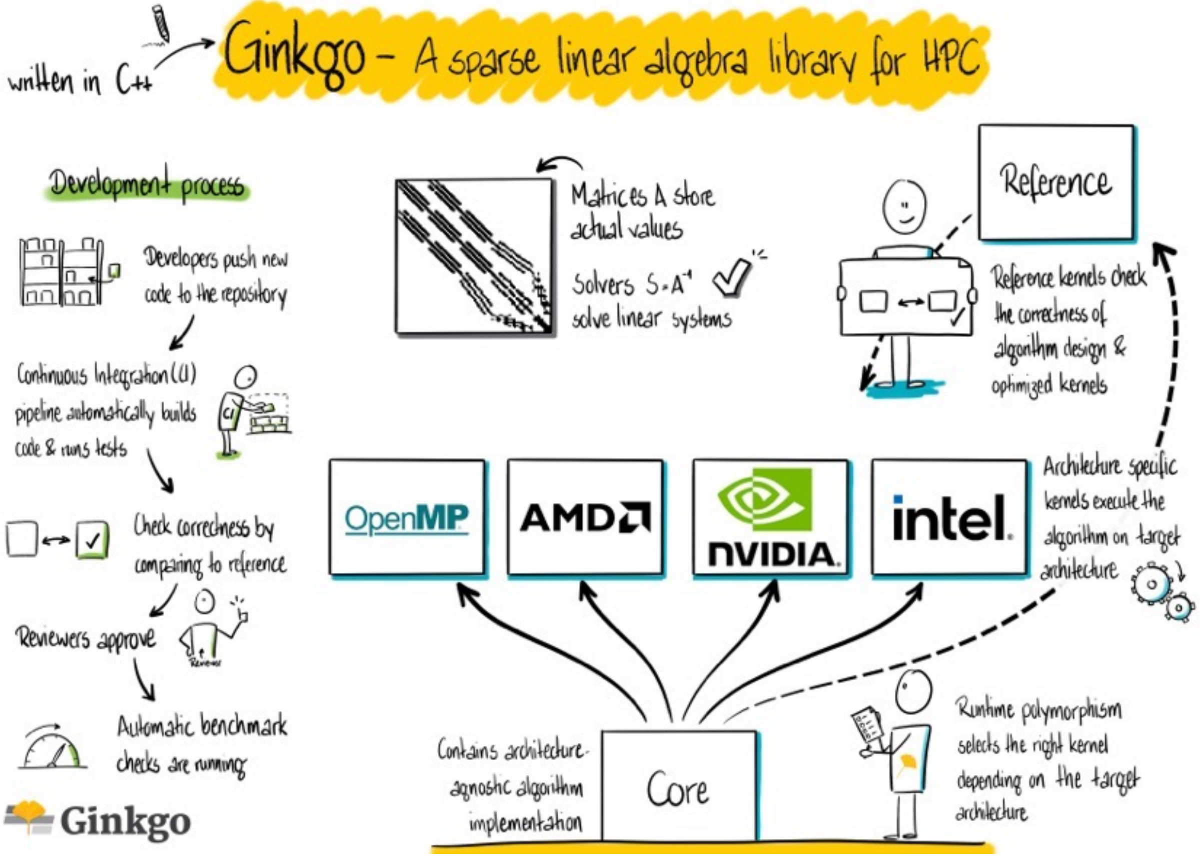


Designing a math toolset for ECP applications

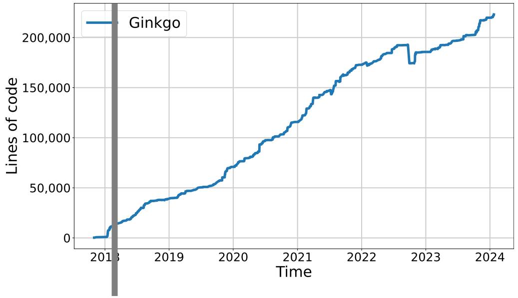


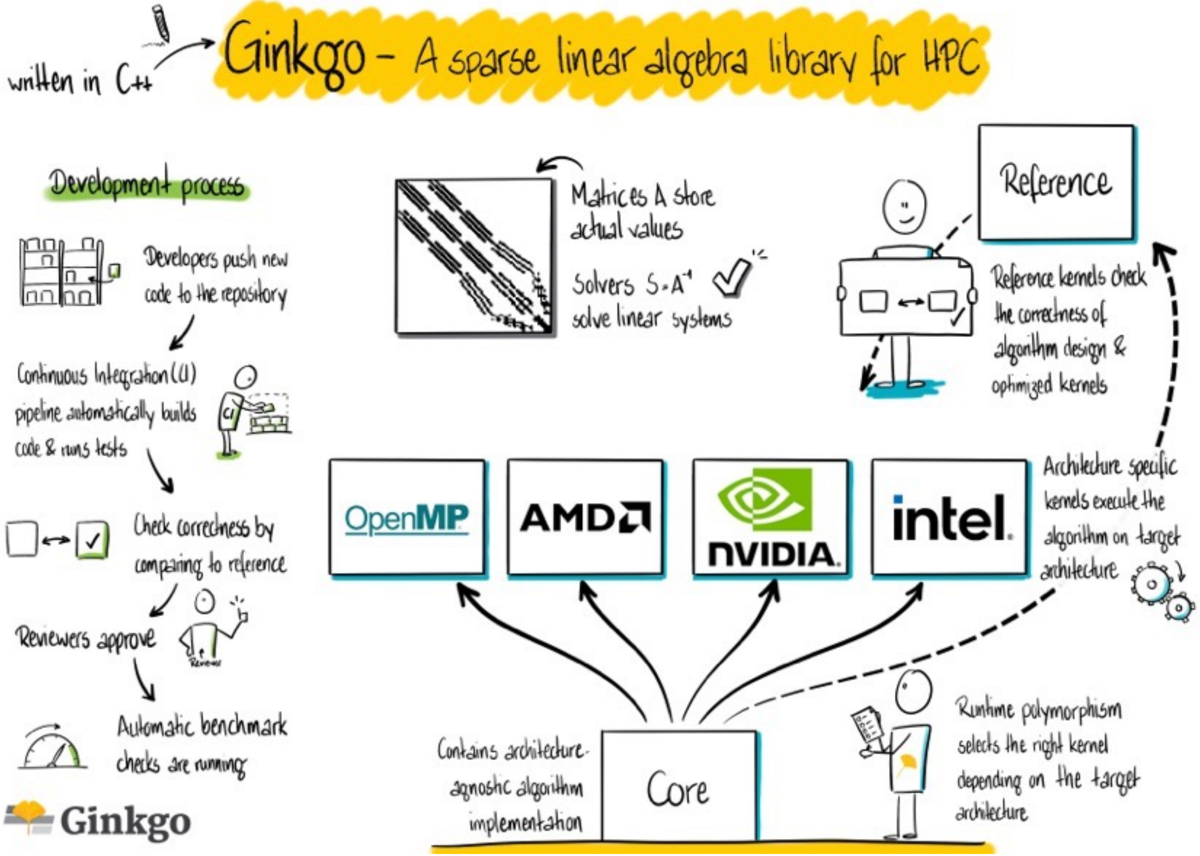
Designing a math toolset for ECP applications



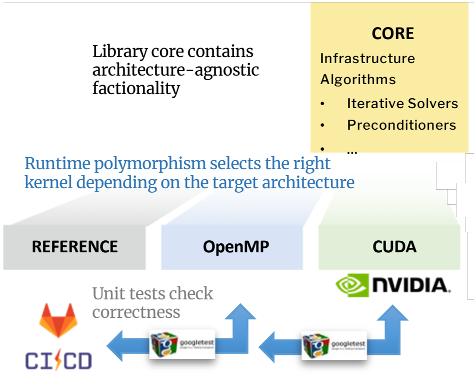


Designing a math toolset for ECP applications

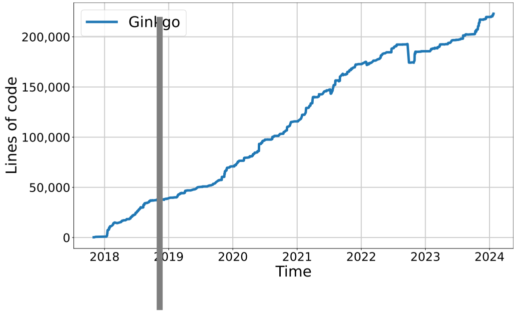




Starting with the CUDA backend

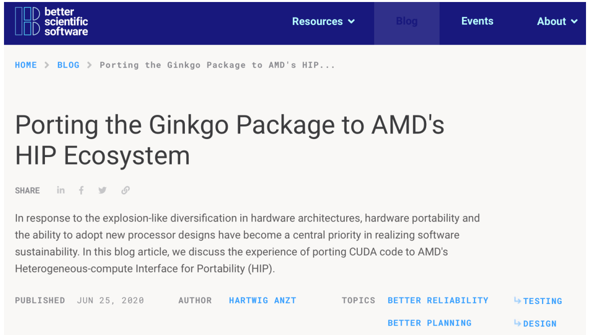


	Functionality	OMP	CUDA
Basic	SpMV	✓	✓
	SpMM	✓	✓
	SpGeMM	✓	✓
Krylov solvers	BiCG	✓	✓
	BiCGSTAB	✓	✓
	CG	✓	✓
	CGS	✓	✓
	GMRES	✓	✓
	IDR	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓
	ILU/IC		✓
	Parallel ILU/IC	✓	✓
	Parallel ILUT/ICT	✓	✓
	Sparse Approximate Inverse	✓	✓



Extending to AMD GPUs

~2 months


better scientific software | Resources | Blog | Events | About

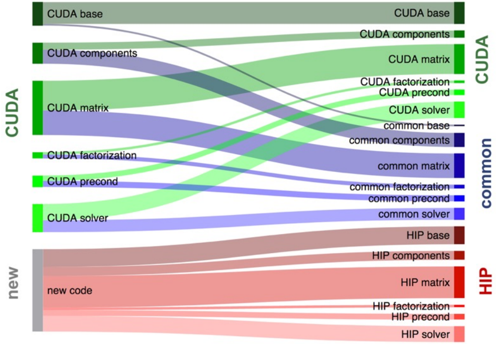
HOME > BLOG > Porting the Ginkgo Package to AMD's HIP...

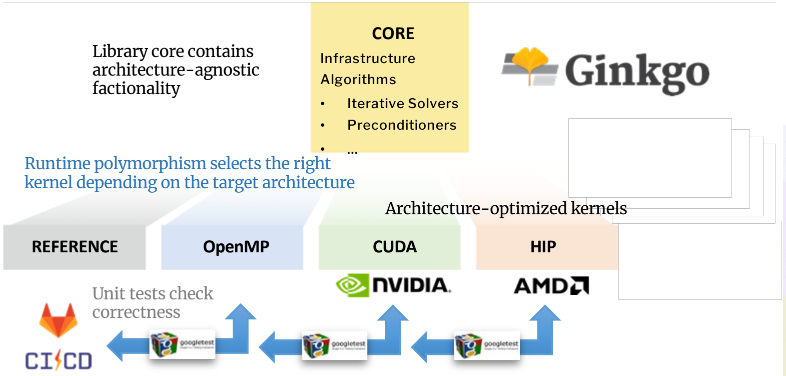
Porting the Ginkgo Package to AMD's HIP Ecosystem

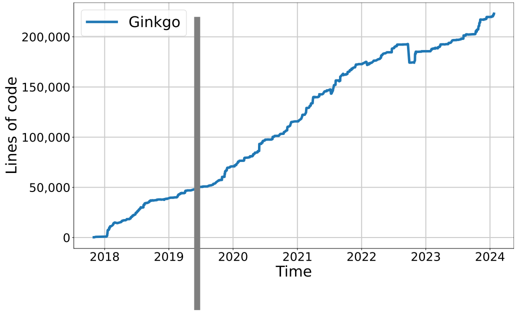
SHARE

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







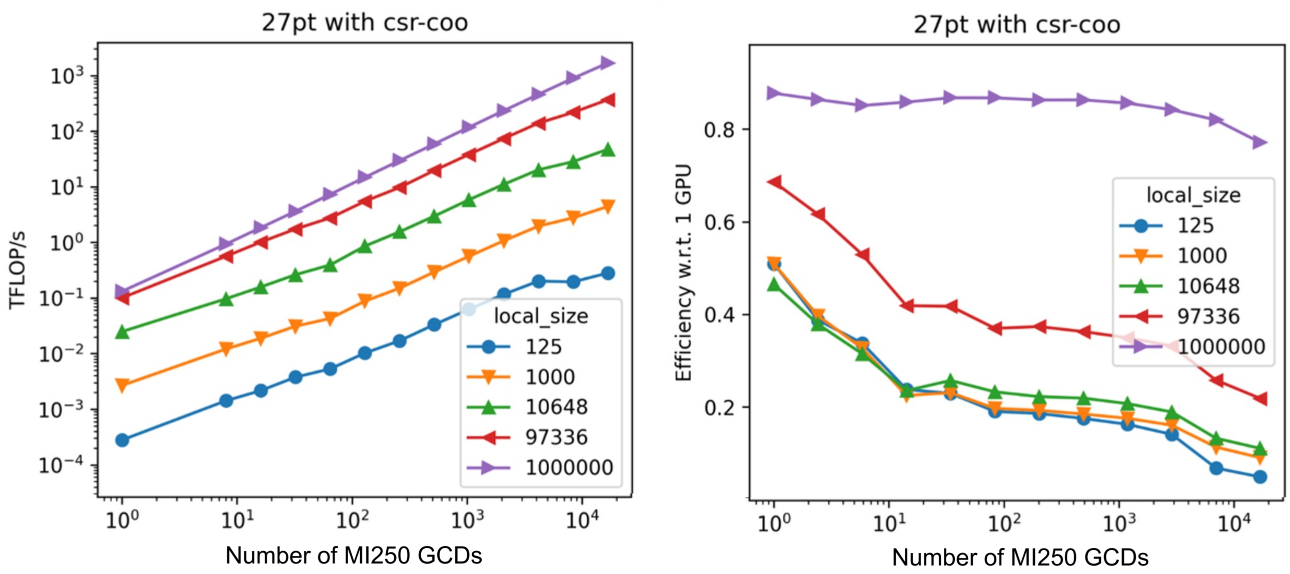
	Functionality	OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓

Weak and strong Scalability

Frontier (#1 TOP500)

SpMV Weak scaling: problem size increases with parallel resources

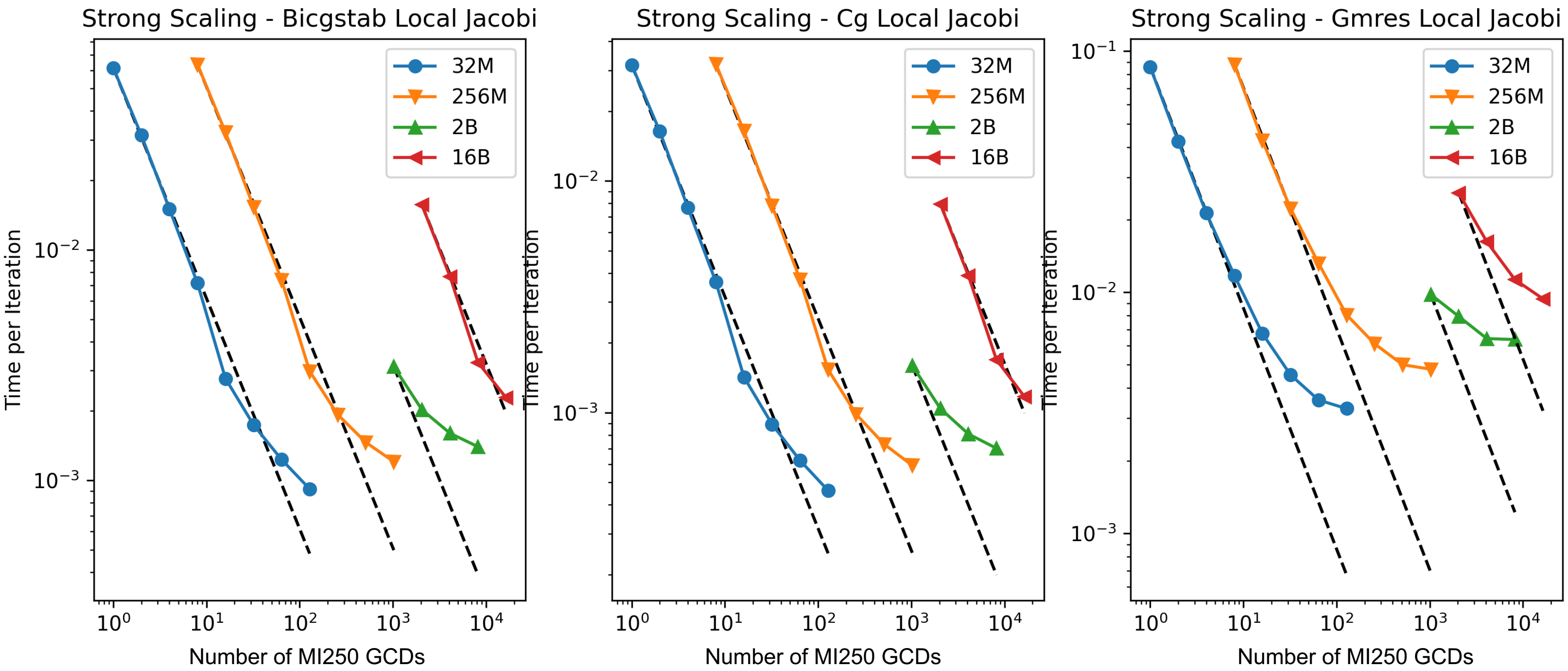
Weak scaling up to 8k AMD MI250 GPUs (16k GCDs)



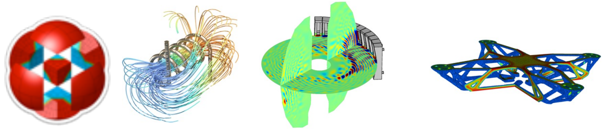
Weak and strong Scalability

Strong scaling: problem size constant, parallel resources increase

Frontier (#1 TOP500)



We “forgot” the customer on the way…



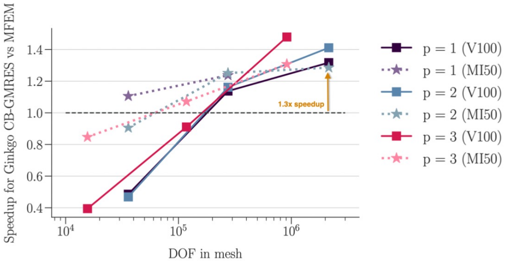
MFEM is a *free*, *lightweight*, *scalable* C++ library for finite element methods.

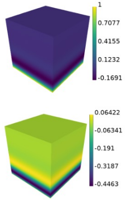
Speeding up MFEM’s “example 22” on GPUs

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:

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

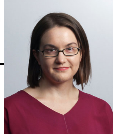
with a = 1, b = 1, ω = 10, c = 20



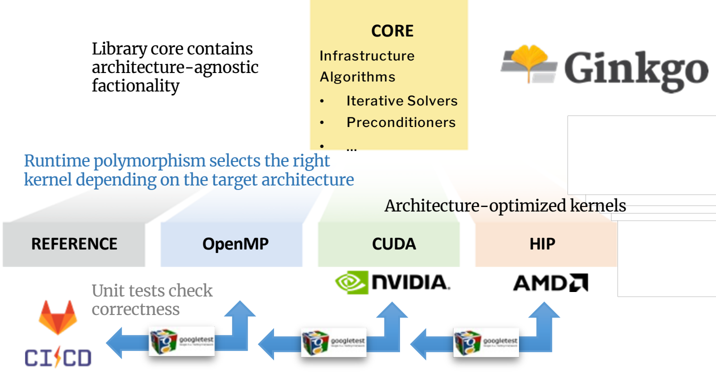


Real part of solution (top), imaginary part of solution

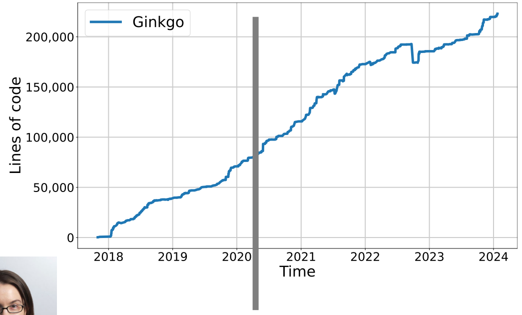
Speedup of Ginkgo’s Compressed Basis-GMRES solver vs MFEM’s GMRES solver for three different orders of basis functions (p), using MFEM matrix-free operators and the Ginkgo-MFEM integration wrappers in MFEM. CUDA 10.1/V100 and ROcm 4.0/MI50.



Natalie Beams

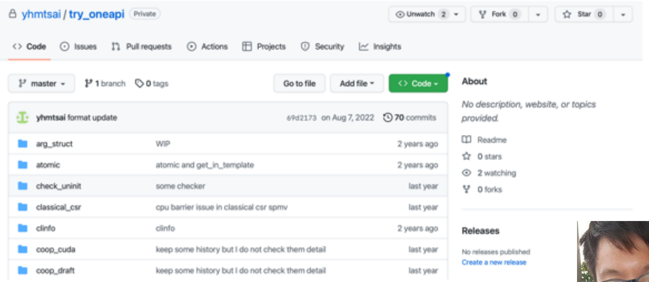


Functionality		OMP	CUDA	HIP
Basic	SpMV	✓	✓	✓
	SpMM	✓	✓	✓
	SpGeMM	✓	✓	✓
Krylov solvers	BiCG	✓	✓	✓
	BiCGSTAB	✓	✓	✓
	CG	✓	✓	✓
	CGS	✓	✓	✓
	GMRES	✓	✓	✓
	IDR	✓	✓	✓
Preconditioners	(Block-)Jacobi	✓	✓	✓
	ILU/IC		✓	✓
	Parallel ILU/IC	✓	✓	✓
	Parallel ILUT/ICT	✓	✓	✓
	Sparse Approximate Inverse	✓	✓	✓
Utilities	On-Device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

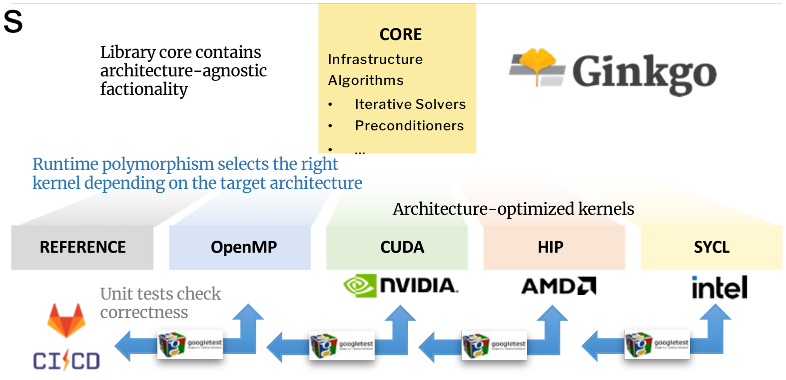


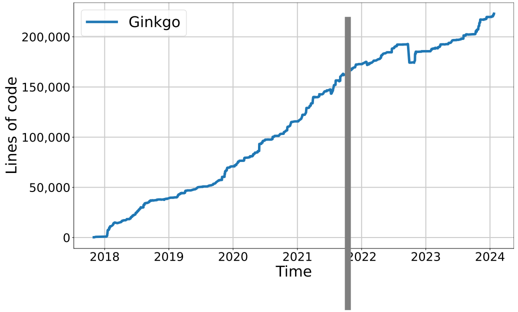
Extending to Intel GPUs

~18 months



Mike Tsai





	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!

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

New issue

Closed mmeterel opened this issue 22 days ago · 3 comments

mmeterel commented 22 days ago · edited — Contributor

Summary

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

Version

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

```
 1 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
 3 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.RealDoublePrecision/Column_Major_TITAN_RTX (Failed)
 7 - BLAS/RT/Nrm2TestSuite/Nrm2Tests.ComplexDoublePrecision/Column_Major_TITAN_RTX (Failed)
17 - BLAS/RT/IamaxTestSuite/IamaxTests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
19 - BLAS/RT/IamaxTestSuite/IamaxTests.RealDoublePrecision/Column_Major_TITAN_RTX (Failed)
23 - BLAS/RT/IamaxTestSuite/IamaxTests.ComplexDoublePrecision/Column_Major_TITAN_RTX (Failed)
27 - BLAS/RT/DotuTestSuite/DotuTests.ComplexDoublePrecision/Column_Major_TITAN_RTX (Failed)
35 - BLAS/RT/DotcTestSuite/DotcTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
67 - BLAS/RT/AsumTestSuite/AsumTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
81 - BLAS/RT/ScalTestSuite/ScalTests.RealSinglePrecision/Column_Major_TITAN_RTX (Failed)
85 - BLAS/RT/ScalTestSuite/ScalTests.ComplexSinglePrecision/Column_Major_TITAN_RTX (Failed)
```

Assignees: mmeterel
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: No branches or pull requests
2 participants

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

tid % subgroup size >= 4 gives wrong division

(double) 1/a gives wrong result when the tid % subgroup si
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 0.93162935
1.0000000506
CPU has more worse result

It is connected to optimizations (not reproducible with O0).
fp-speculation=off do not improve results.
Ticket number: XDEPS-4031 ()

Devcloud node issue

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

ginkgohub/oneapi:cuda11.6
DIGEST: sha256:0bc4c10d79a75b183ac1deafcda753365c6e1a94edc3046a9a0eb8ba2d7b9d94

OS/ARCH	COMPRESSED SIZE	LAST PUSHED
linux/amd64	6.63 GB	22 days ago by yhmtsai

IMAGE LAYERS

```
1  ADD file ... in /
2  CMD ["bash"]
3  ENV NVARCH=x86_64
```

Fix cuda/hip backend location #219

New issue

Merged mkrainiuk merged 2 commits into oneapi-src:develop from yhmtsai:fix_cuda_backend_location 20 days ago

Conversation 8 · Commits 2 · Checks 0 · Files changed 16 · +76 −0

yhmtsai commented on Aug 1 · edited — Contributor

Description

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

All Submissions

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

Bug fixes

- Have you added relevant regression tests? A: It is a compiling issue.
- Have you included information on how to reproduce the issue (either in a GitHub issue or in this PR)?

Reproduce:
compile the latest intel llvm and this repo, it will not be able to compile due to missing headers.

yhmtsai added 2 commits 2 months ago

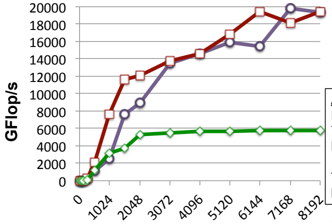
- `use the correct sycl path after intel/llvm#6407` 8edef4a
- `fix the missing sycl/sycl.hpp` 03a2f6f

mmeterel commented on Aug 1 — Contributor

@yhmtsai Thanks for the PR. Is the description `from sycl/CL to CL` correct? My understanding is all header files moved

Reviewers: mkrainiuk, mmeterel
Assignees: No one assigned
Labels: None yet
Projects: None yet
Milestone: No milestone
Development: Successfully merging this pull request may close these issues. None yet
3 participants

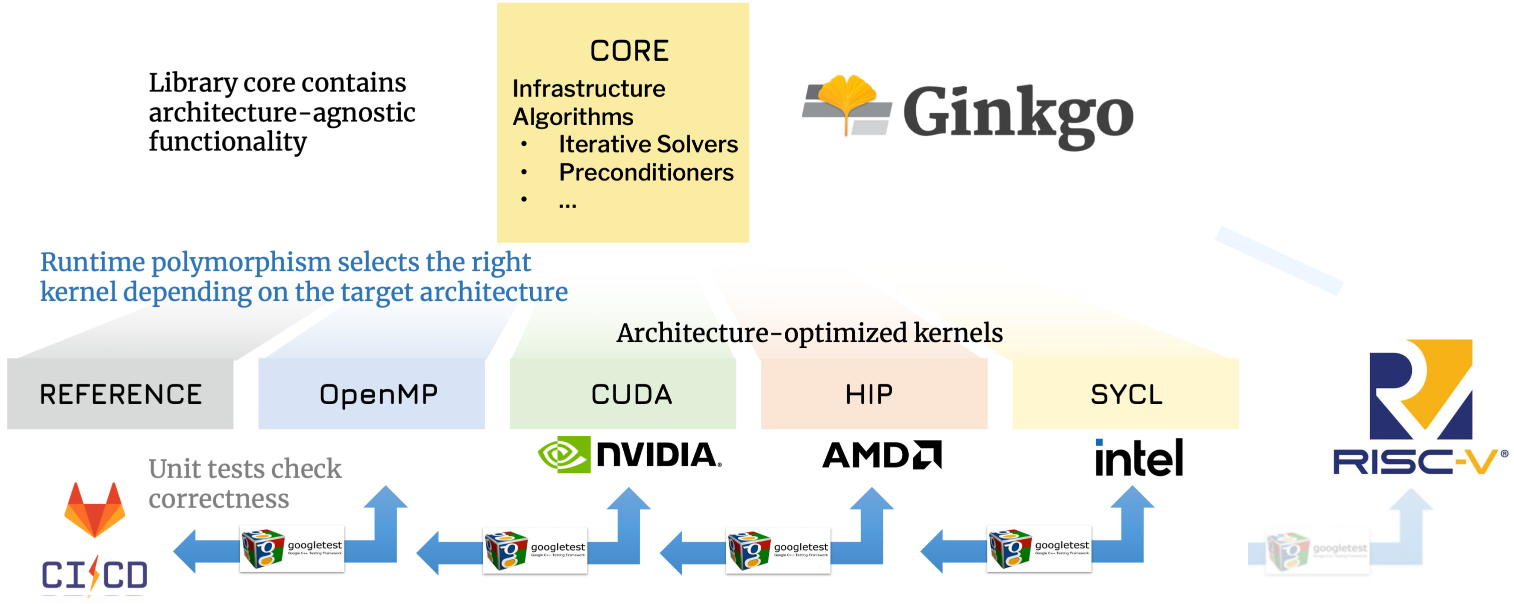
Performance of DPC++ MAGMA SGEMM on Intel GPUs



Arcticus at ALCF
Intel Xe-HP GPU (Arctic Sound 2x)
7,680 x2 Cores @900 MHz
FP32 peak 13,820 x2 GFlop/s

Transitional system to Aurora Exascale supercomputer at ANL

Portability and Extendibility as Central Design Principle

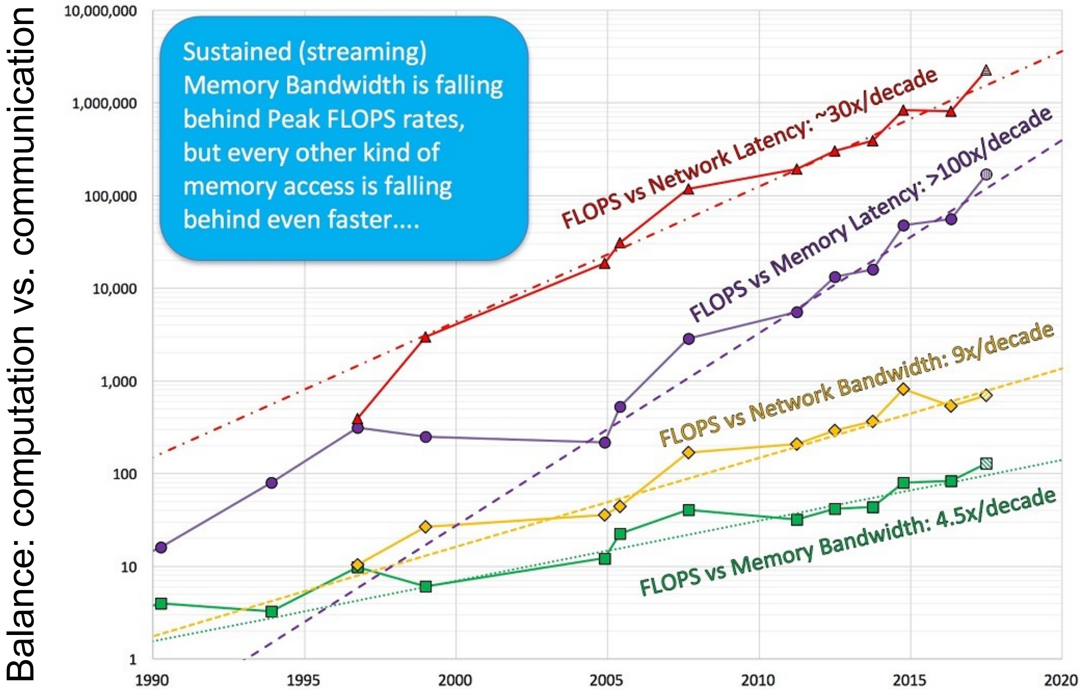


This software design gives portability, performance, and sustainability.

Hardware Trends

NVIDIA

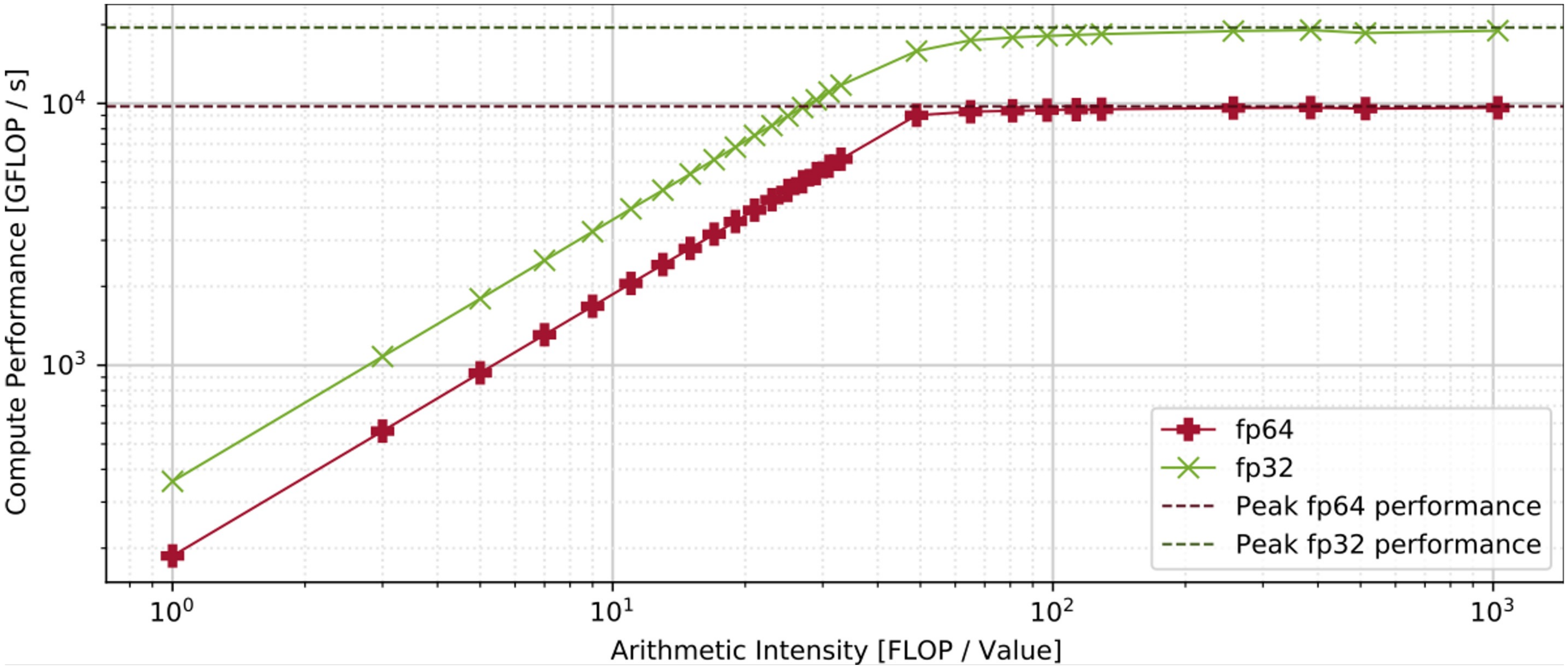
Form Factor	H100 SXM
FP64	34 teraFLOPS
FP64 Tensor Core	67 teraFLOPS
FP32	67 teraFLOPS
TF32 Tensor Core	989 teraFLOPS[2]
BFLOAT16 Tensor Core	1,979 teraFLOPS[2]
FP16 Tensor Core	1,979 teraFLOPS[2]
FP8 Tensor Core	3,958 teraFLOPS[2]
INT8 Tensor Core	3,958 TOPS[2]
GPU memory	80GB
GPU memory bandwidth	3.35TB/s



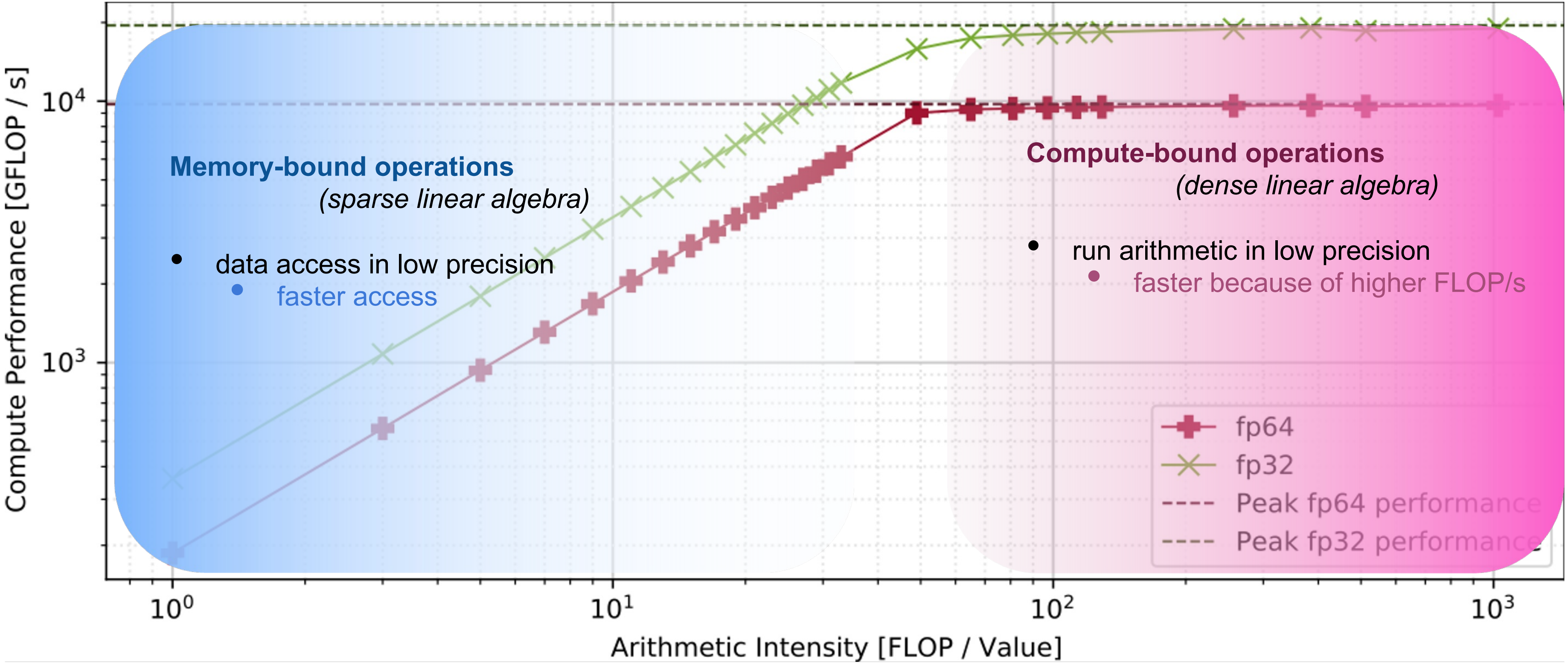
- (Dense) Matrix Performance > Vector Performance
- Low Precision Perf > High Precision Performance

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

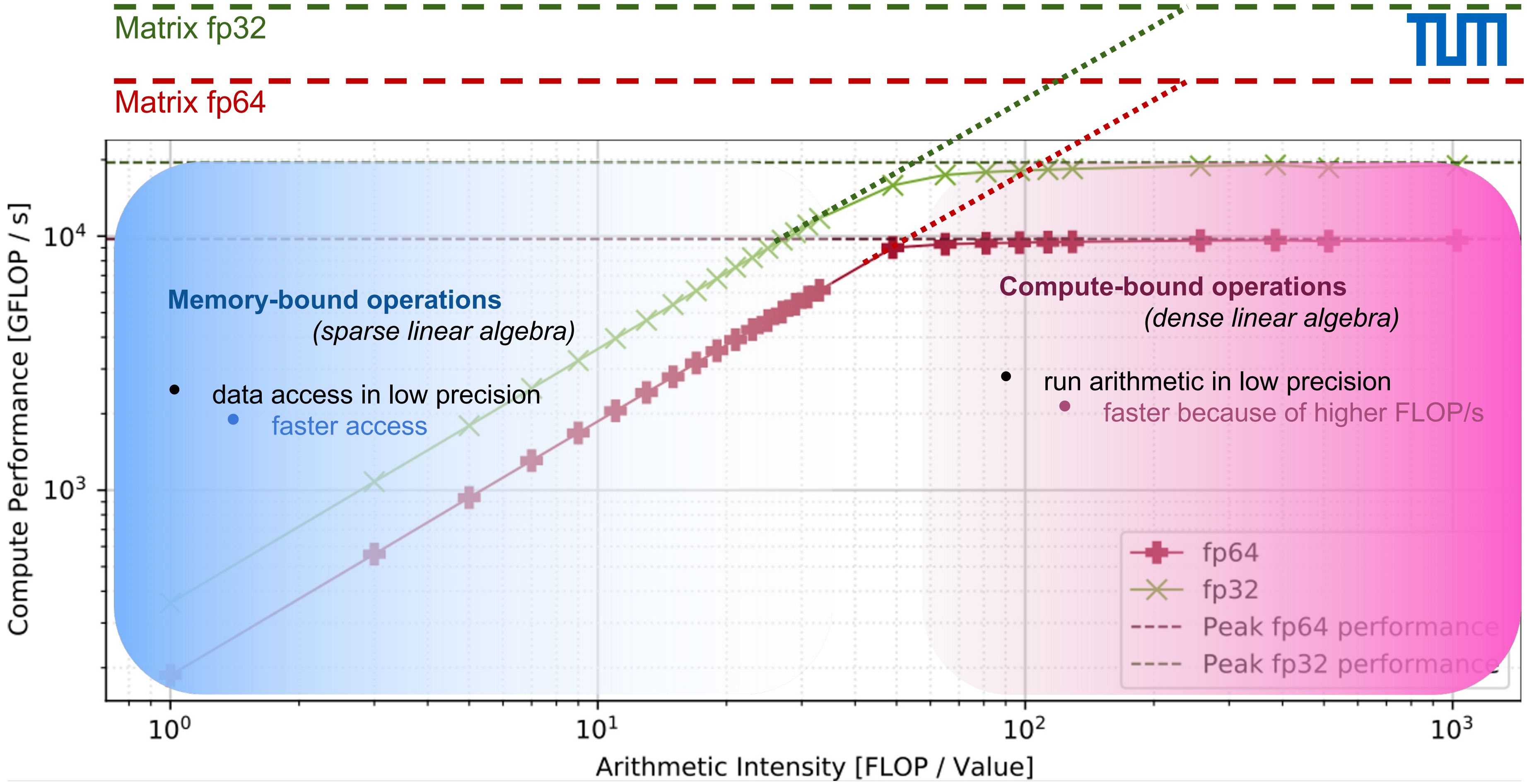
NVIDIA A100



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



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]
Arithmetic Intensity [FLOP / Value]
10^4
10^3
10^0
10^1
10^2
10^3



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**

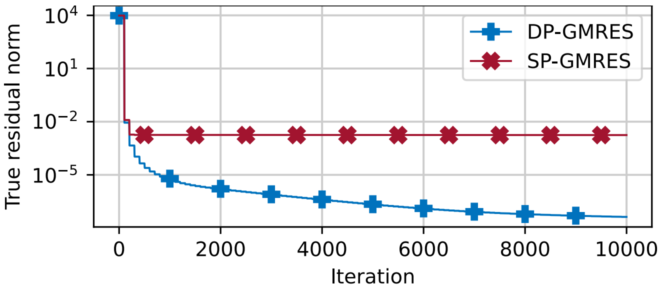
```
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms
```

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.

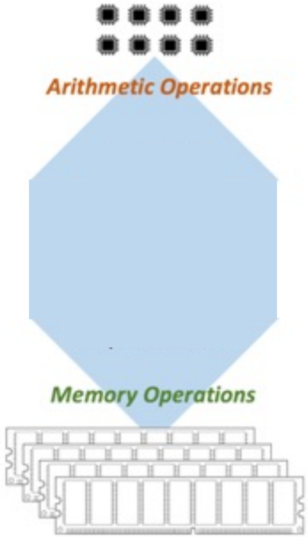
ECP Focus Effort Mixed Precision

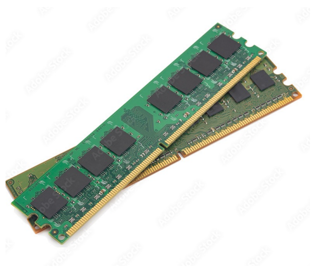
- Traditionally, we use a strong coupling between the

  precision formats used for arithmetic operations

  the precision format handling data in main memory.

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





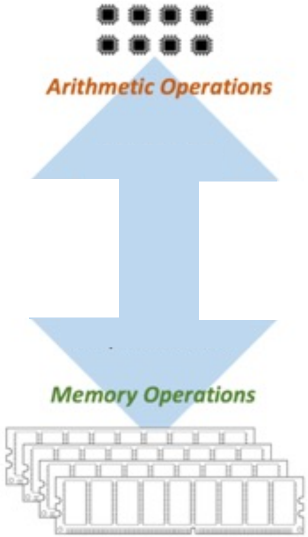
ECP Focus Effort Mixed Precision

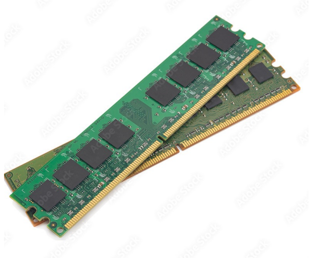
- Traditionally, we use a strong coupling between the

  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*





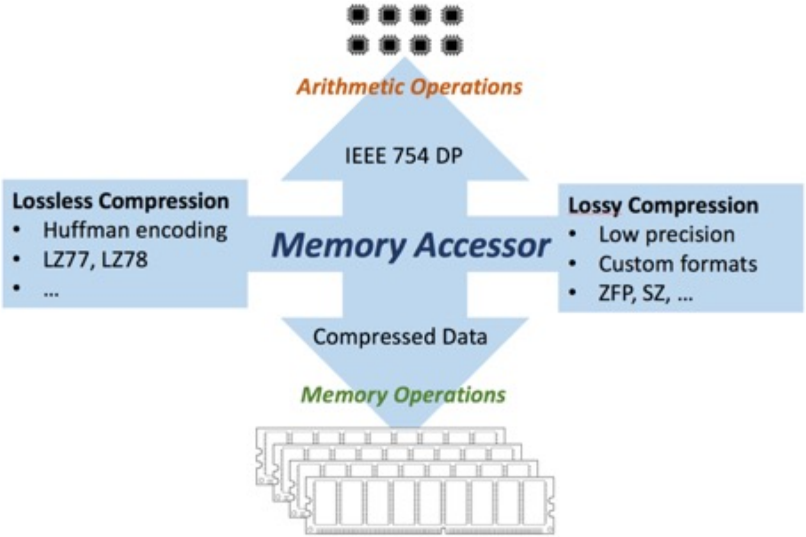
ECP Focus Effort Mixed Precision

- Traditionally, we use a strong coupling between the

  precision formats used for arithmetic operations

  the precision format handling data in main memory.

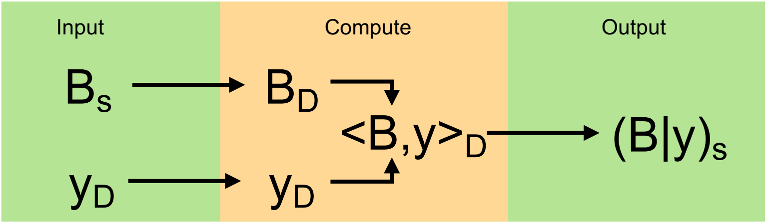
- *We should compute in fp64*
- *Data should be compressed for main memory access (low precision/compression)*
- *Compression / Conversion needs to happen on-the-fly*

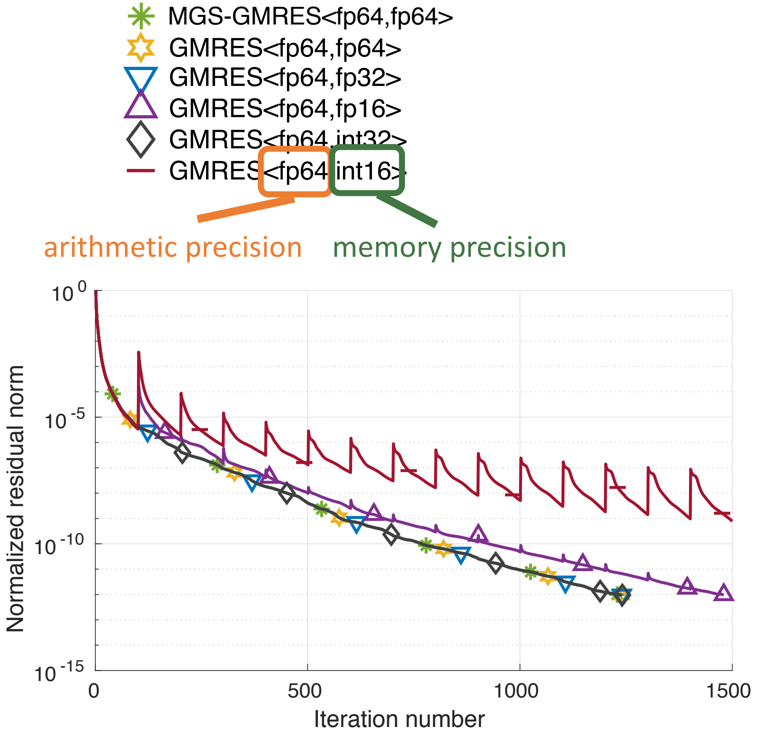


ECP Focus Effort Mixed Precision

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**

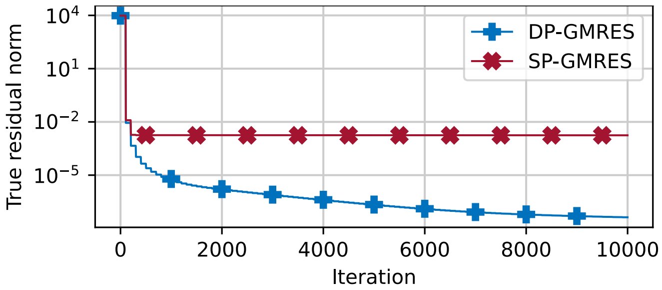
```
Double precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 9.6639e-09
GMRES iteration count: 23271
GMRES execution time: 43801 ms
```

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}

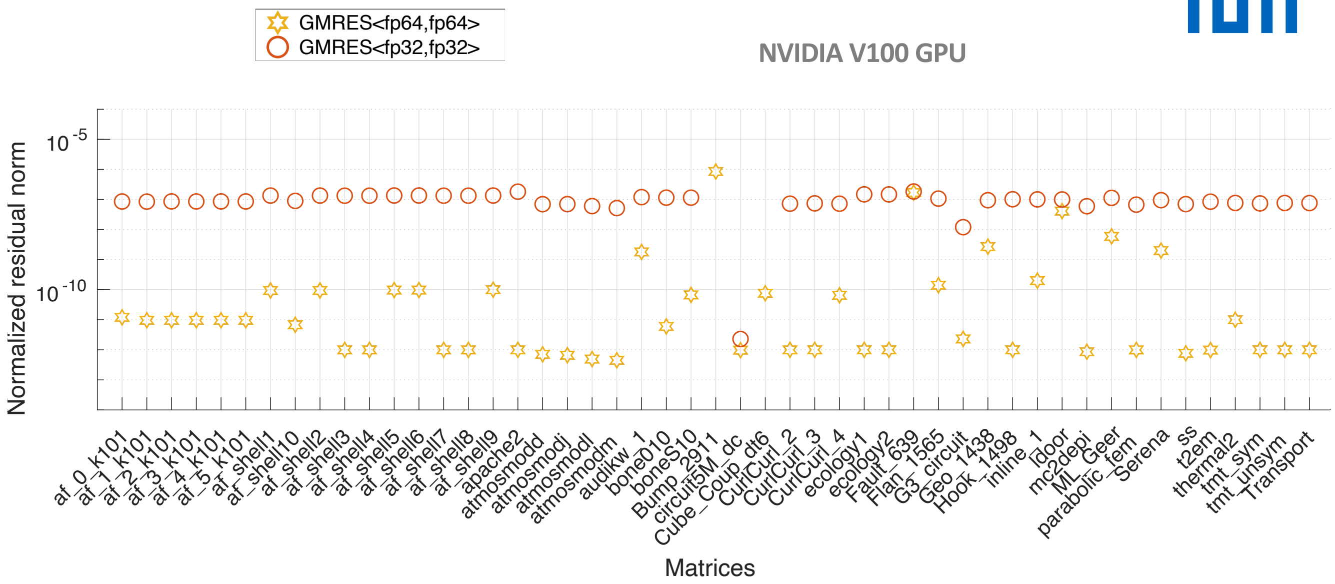
```
Single precision GMRES
Initial residual norm
sqrt(r^t r): 9670.36
Final residual norm
sqrt(r^T r): 0.00175464
GMRES iteration count: 25000
GMRES execution time: 27376 ms
```

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}

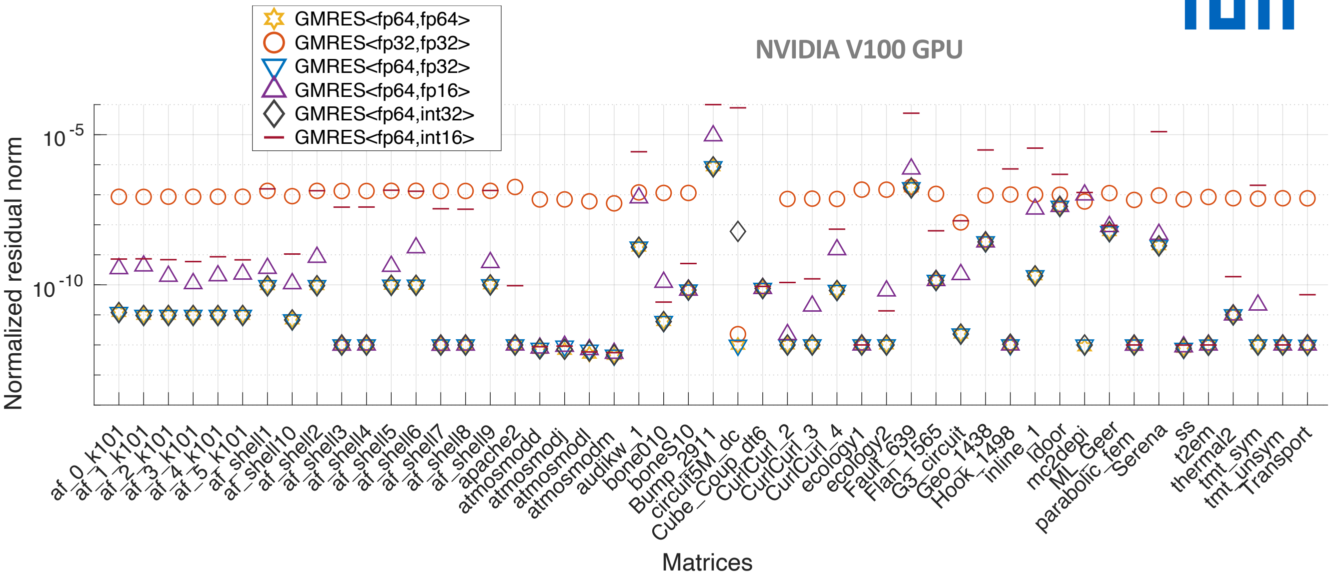
Accuracy of DP GMRES
Performance similar to SP GMRES



GMRES<fp64,fp64>
GMRES<fp32,fp32>
NVIDIA V100 GPU
Normalized residual norm
10^{-5}
10^{-10}
Matrices
af_0_k101
af_1_k101
af_2_k101
af_3_k101
af_4_k101
af_5_k101
af_shell1
af_shell10
af_shell2
af_shell3
af_shell4
af_shell5
af_shell6
af_shell7
af_shell8
af_shell9
apache2
atmosmodd
atmosmodj
atmosmodl
atmosmodm
audikw_1
bone010
boneS10
Bump_2911
circuit5M_dc
Cube_Coup_dt6
CurlCurl_2
CurlCurl_3
CurlCurl_4
ecology1
ecology2
Fault_639
Flan_1565
G3_circuit
Geo_1438
Hook_1498
inline_1
ldoor
mc2depi
ML_Geer
parabolic_fem
Serena
ss
t2em
thermal2
tmt_sym
tmt_unsym
Transport

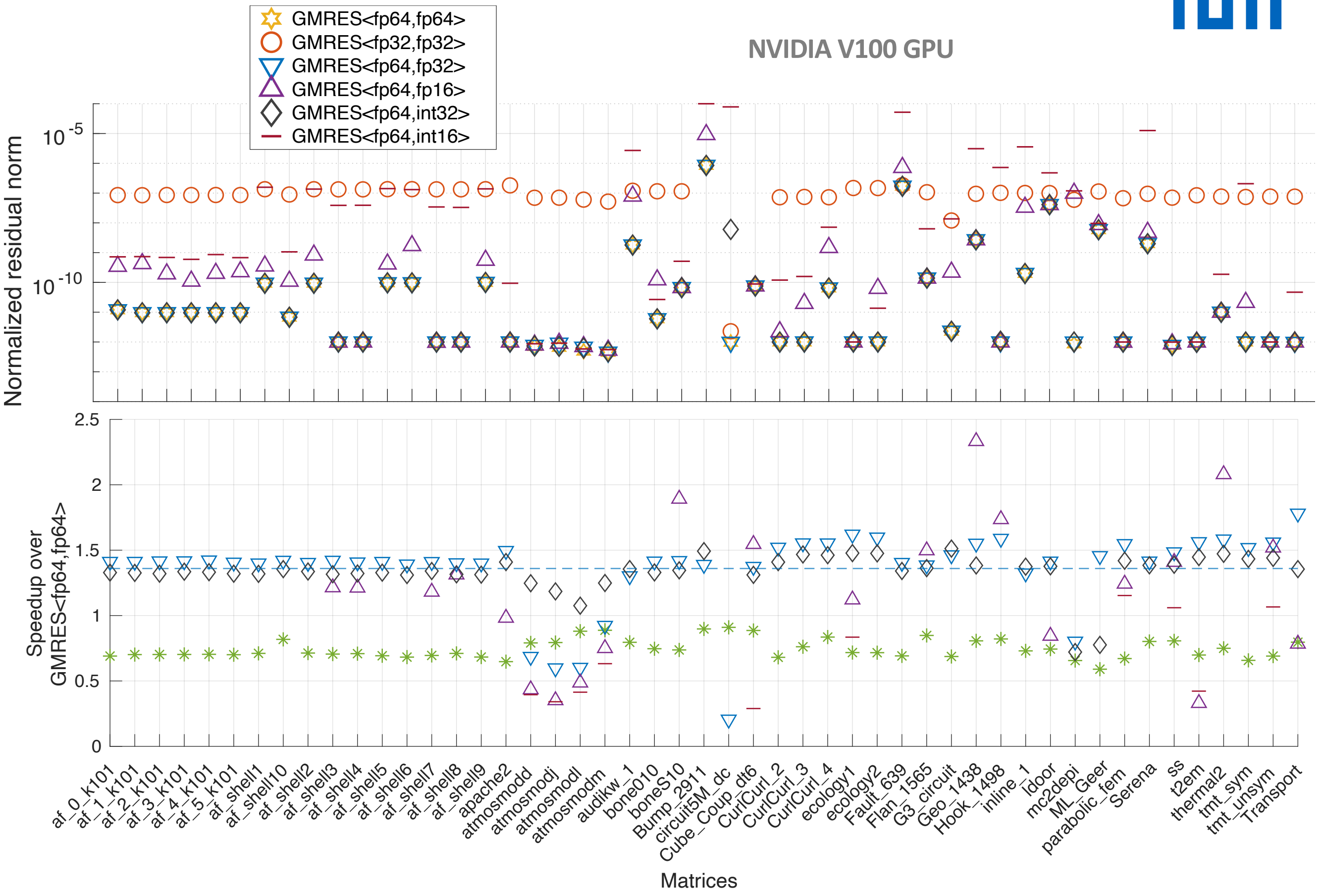
NVIDIA V100 GPU

- 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



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

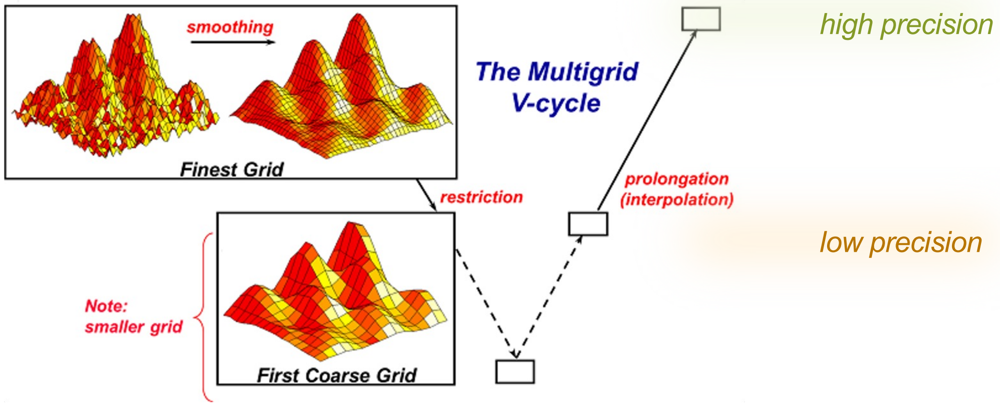
ECP Focus Effort Mixed Precision

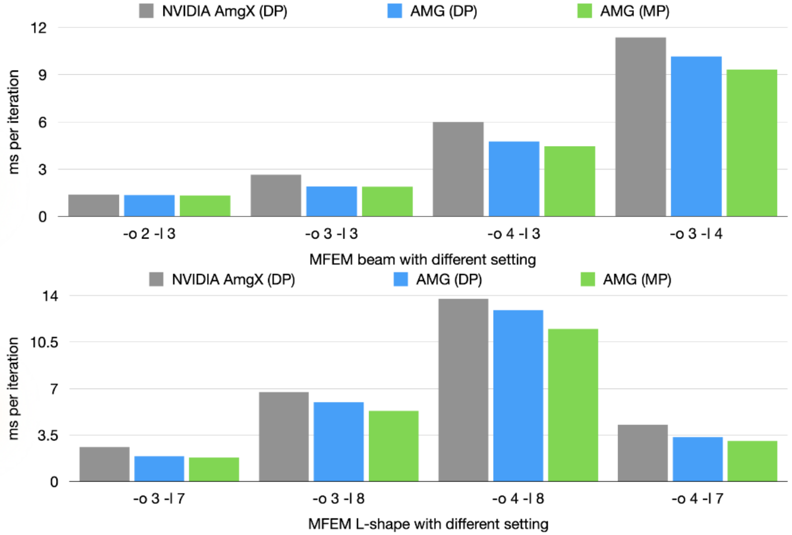
- **Preconditioning iterative solvers**
  - Idea: Approximate inverse of system matrix to make the system "easier to solve": $P^{-1} \approx A^{-1}$

    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



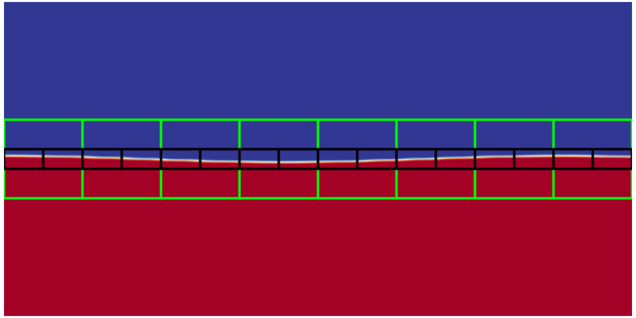


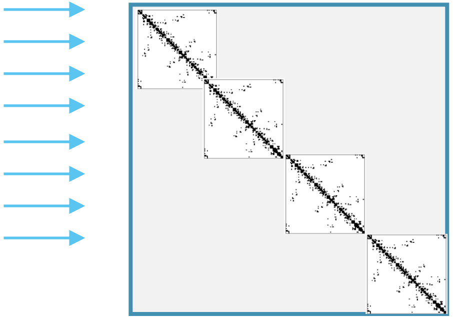
Batched Focus Effort

Batched iterative solvers for SUNDIALS / PeleLM

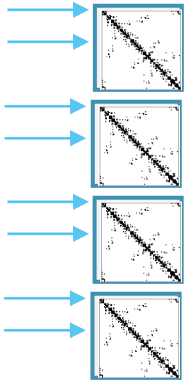
PeleLM is a parallel, adaptive mesh refinement (AMR) code that solves the reacting Navier-Stokes equations in the low Mach number regime. The core libraries for managing the subcycling AMR grids and communication are found in the [AMReX source code.]

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

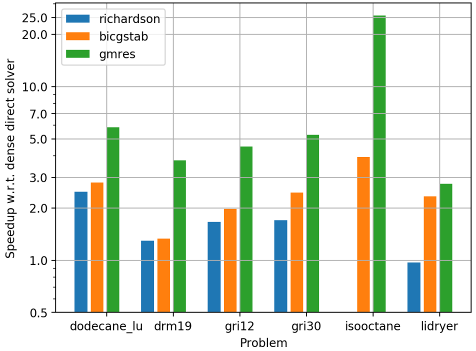




Carol Woodward   Cody Balos



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%)

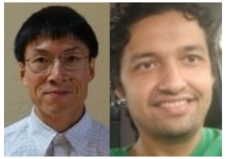


Batched Sparse Iterative Solvers for Computational Chemistry Simulations on GPUs

Publisher: IEEE Cite This PDF

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

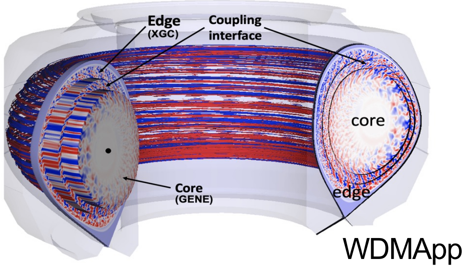
Batched Functionality for the Collision Operator

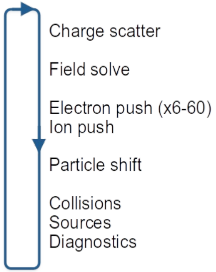


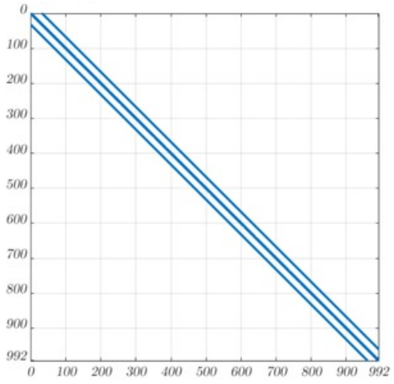
Paul Lin   Dhruva Kulkarni

__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.

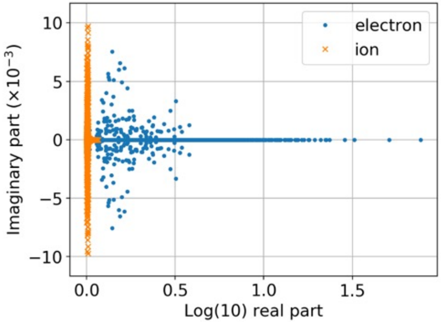
Source: https://xgc.pppl.gov/html/general_info.html







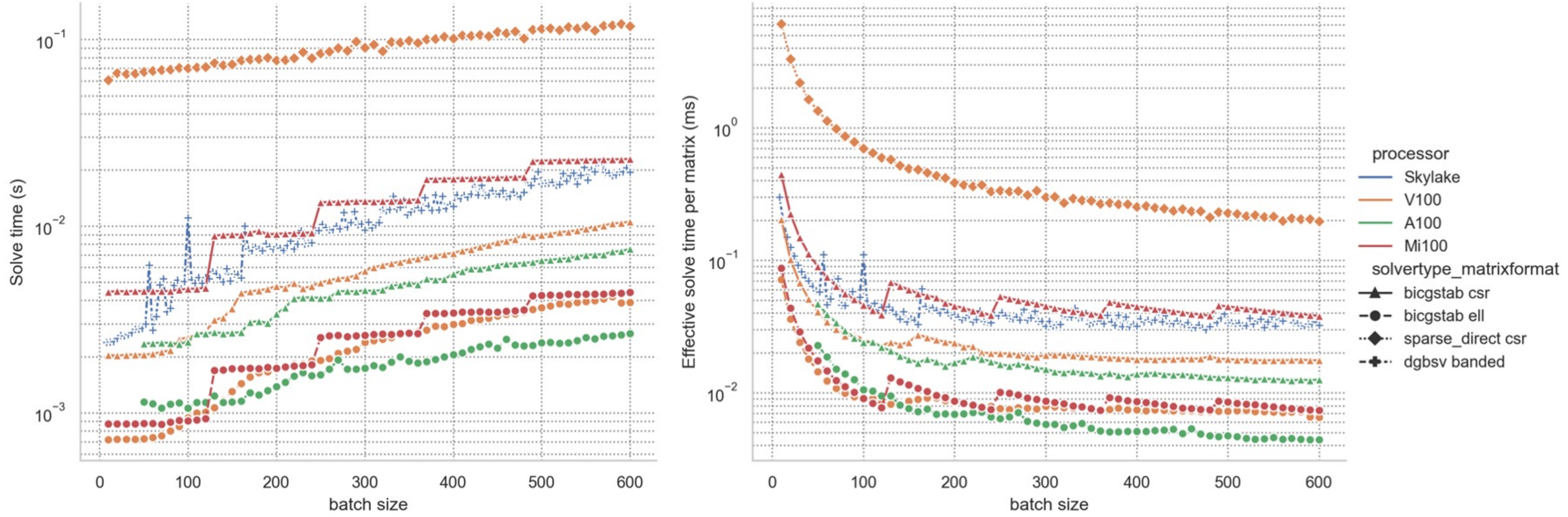
- Two species
- Ions easy to solve
- Electrons hard to solve
- Banded matrix structure
- Non-symmetric, need BiCGSTAB
- n = ~1,000
- nz = ~9,000



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

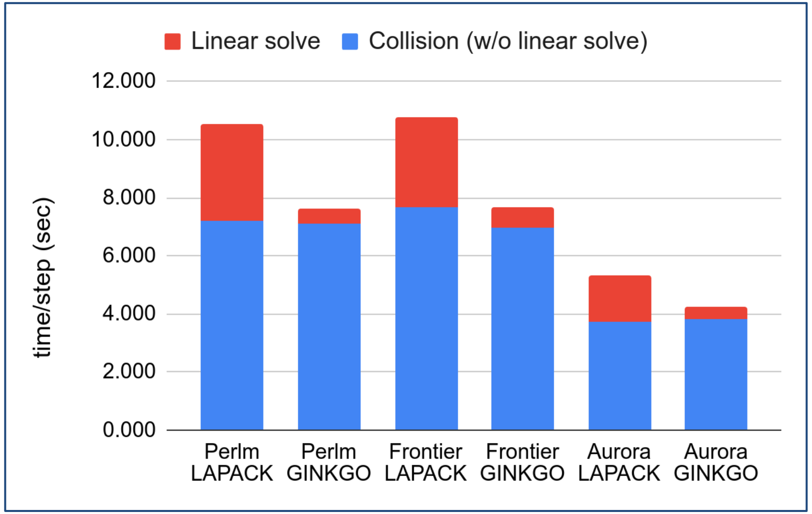
- Outer nonlinear solver: Picard method with inner linear solves
- Linear solve: discretize velocity space with approx 35x35 velocity grid
- direct solve on CPU using LAPACK banded solver **dgbsv**
- After GPU porting of XGC, this is the remaining CPU intensive kernel for collision operator

Batched Functionality for the Collision Operator



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

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:

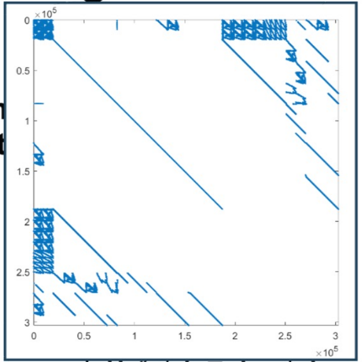
$H_{tsk}(x_t, y_{tsk}) = 0$ flow definitions, power balance

$Q_{tsk}(x_t, y_{tsk}) \leq 0$ 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…

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

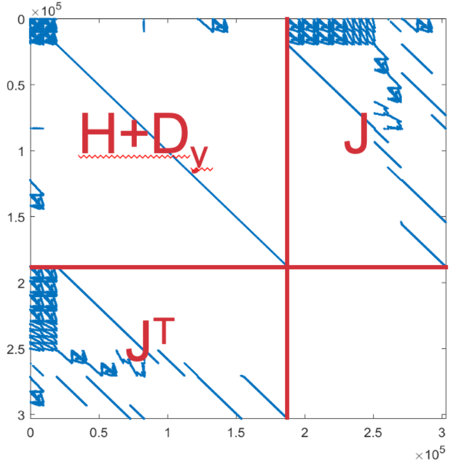


- The characteristic block-arrow coupling structure can be exploited to decompose the optimization problem, nevertheless there is no solver that can tackle this on a GPU-based architecture.

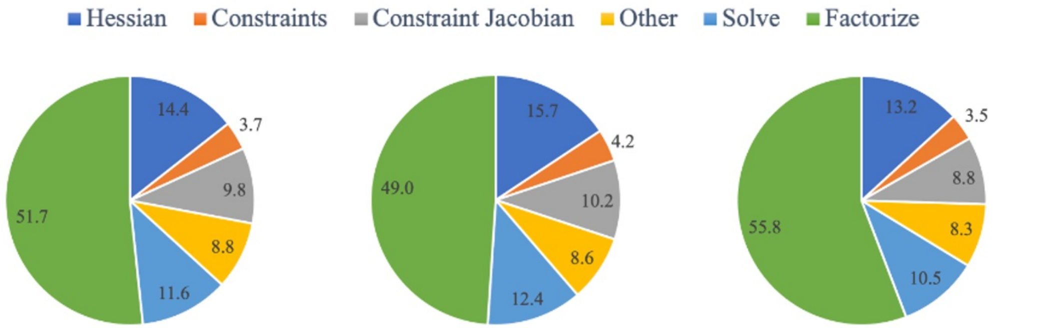
EXASGD

© Slaven Peles

Sparse Direct Solvers

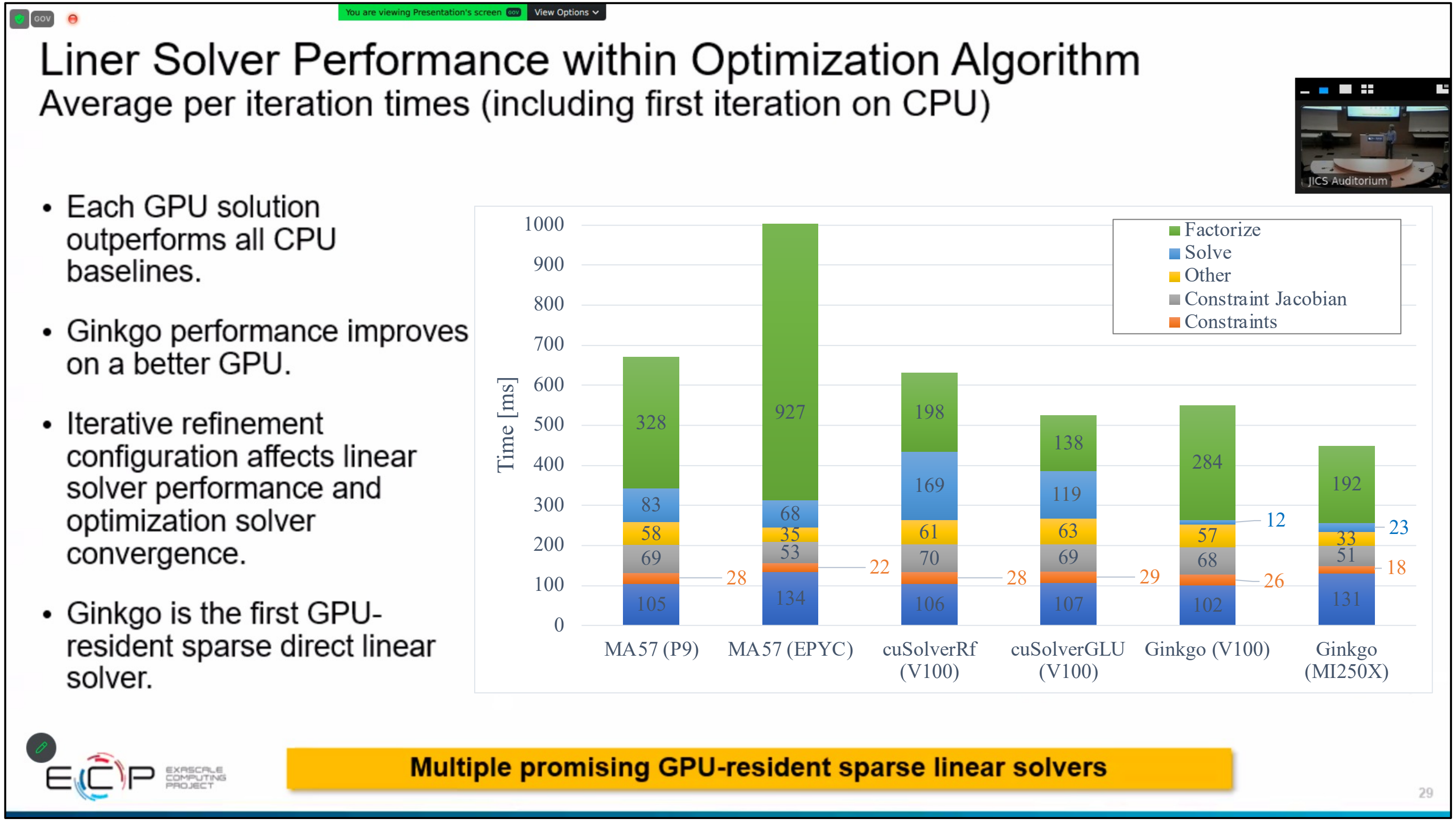


Grid	Buses	Generators	Lines	$N(K_k)$	$\mathrm{nnz}(K_k)$
Northeastern US	25 K	4.8 K	32.3 K	108 K	1.19 M
Eastern US	70 K	10.4 K	88.2 K	296 K	3.20 M
Western and Eastern US	82 K	13.4 K	104.1 K	340 K	3.73 M



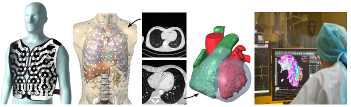
(a) Northeast U.S. grid (b) Eastern U.S. grid (c) Eastern and Western U.S. grids

Sparse Direct Solvers



After 6 years of development

- EuroHPC Project MICROCARD uses Ginkgo

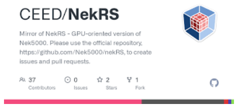


https://www.microcard.eu

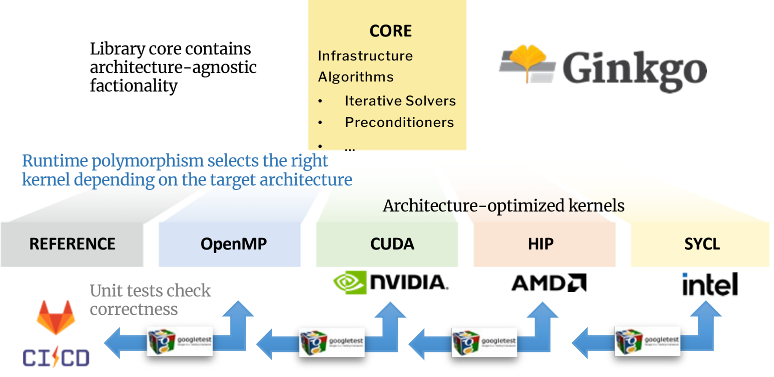
- DoE SciDAC-5 : Development of High-Fidelity Simulation Capabilities for ELM-free Design Optimization

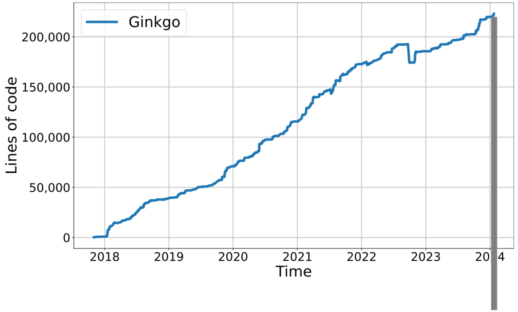
- BMBF PDExa and ExaSIM projects use Ginkgo

The Open Source CFD Toolbox



- Companies are evaluating Ginkgo





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

MPI Support   Single-GPU Support

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.

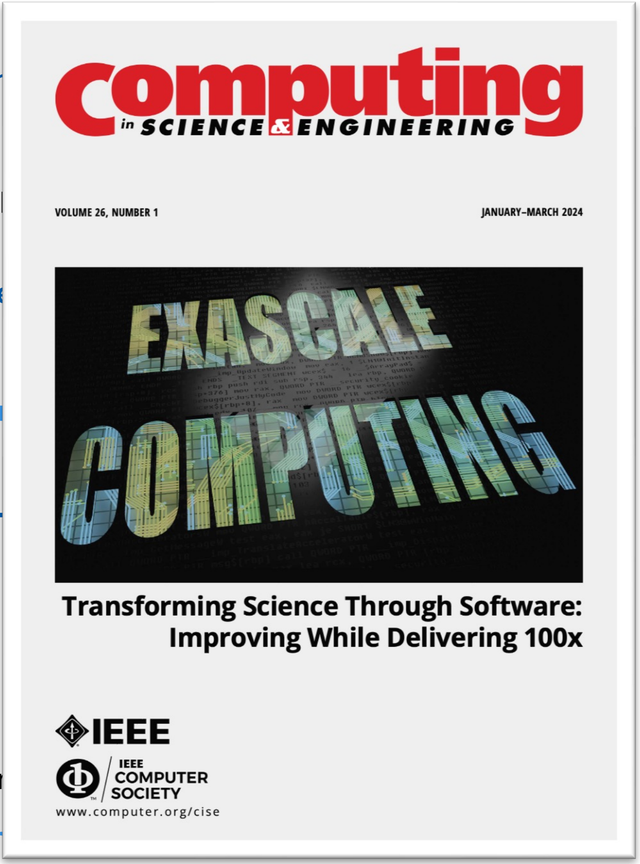
- **Workforce recruitment and workforce retention are the key to success in software development.**
  - Money does not write software. RSEs do. **We need to create attractive career plans.**
  - We need to make research software development attractive to students. **Academic recognition. Industry career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

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

- **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 over the years

- **ECP earmarking roughly half the budget to Software & App development is a gam**
  - **Central component for the success of ECP.**
  - This concept needs to – and does become - the blueprint for other nations, com
- **Workforce recruitment and workforce retention are the key to success in software**
  - Money does not write software. RSEs do. **We need to create attractive career**
  - We need to make research software development attractive to students. **Acade**
- **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 a
  - **Listen to the application needs. Value input and acknowledge collaborator**



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 acces (low precision/compression)*
- *Compression / Conversion needs to happen on-the-fly*