Prompt Engineering for HPC Vibe Coding

Argonne NATIONAL LABORATORY

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

David Keyes
Extreme Computing Research Center
david.keyes@kaust.edu.sa

Vibe coding

- Software development method
- Builds applications by human prompting in natural language of an AI agent *with imperatives, rather than writing code line-by-line*
- Agent(s) handle architecture, design, and coding
- Coined 3 February 2025 by Andrej Karpathy
- My title is an exaggeration… until 6 months from now ☺

Andrej Karpathy
OpenAI (2015)
Tesla (2017)
Eureka Labs (2024)
Anthropic (2026)

Vibe coding at DOE's 2026 Salishan Conf

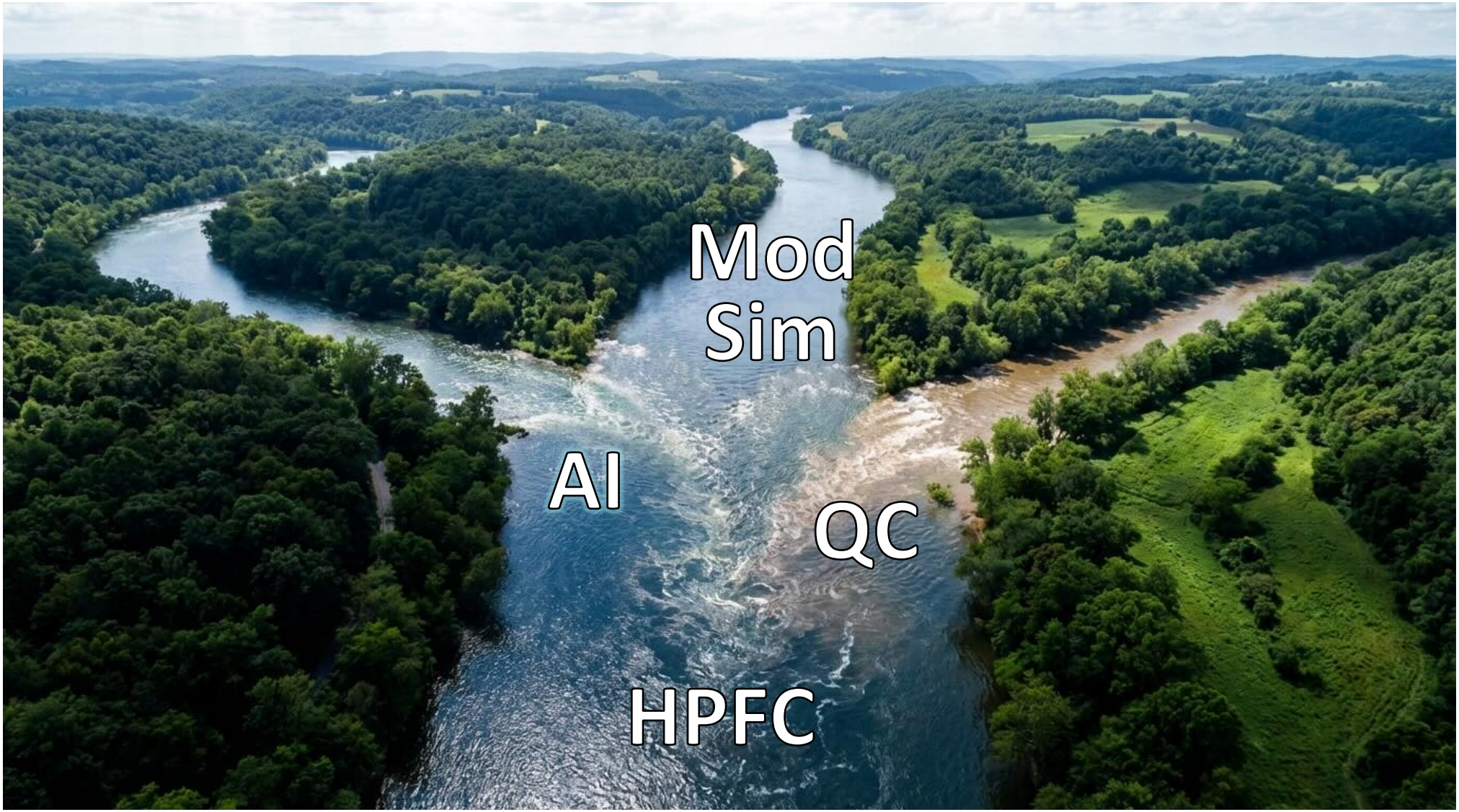
Agentic AI for Multiphysics code development and simulation

Rob Reiben
Project Manager
LLNL

Ignacio's talk today

… Agentic AI is proving valuable in strengthening both operational workflows and early-stage research. **Agentic AI supports code development**, debugging, and performance optimization for large scale multiphysics applications. We highlight interaction modes that range from AI-assisted algorithm design to fully agentic refactors, **including GPU kernel tuning for performance critical regions**…

Imperatives from Mod-Sim for High Performance *Fused* Computing



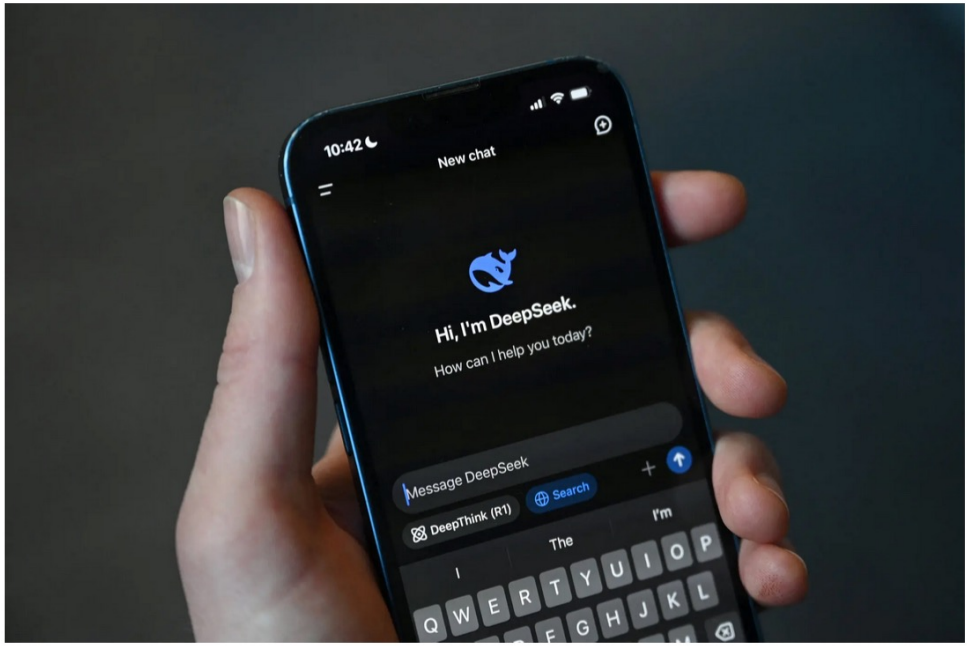
These tributaries need each other

27 January 2025

What to Know About DeepSeek and How It Is Upending A.I.

How did a little-known Chinese start-up cause the markets and U.S. tech giants to quake? Here's what to know.

Listen to this article · 9:13 min Learn more Share full article 306



DeepSeek is a Chinese start-up run by a quantitative stock trading firm called High-Flyer. Greg Baker/Agence France-Presse — Getty Images

- Static/dynamic runtime system
- Overlapped communication with computation w/lookahead
- Low-precision representations
- Fused matrix transpose and computations
- Reduction of storage and communication thru redundant (cheap) computation
- Tile quantization

These are all standard HPC techniques!

A big win was, in addition:

- Mixture of experts (MoE, 1991)

arXiv:2412.19437v1 27 Dec 2024, 53 pp.

These tributaries are finding each other

Platform Architecture for Tight Coupling of High-Performance Computing with Quantum Processors

SHANE A. CALDWELL, MOEIN KHAZRAEE, ELENA AGOSTINI, TOM LASSITER, COREY SIMPSON, OMRI KAHALON, MRUDULA KANURI, JIN-SUNG KIM, SAM STANWYCK, MUYUAN LI, JAN OLLE, CHRISTOPHER CHAMBERLAND, BEN HOWE, BRUNO SCHMITT, JUSTIN G. LIETZ, and ALEX MCCASKEY, NVIDIA Corporation, USA

JUN YE, Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore

ANG LI, Pacific Northwest National Laboratory, USA and University of Washington, USA

ALICIA B. MAGANN, COREY I. OSTROVE, KENNETH RUDINGER, ROBIN BLUME-KOHOUT, and KEVIN YOUNG, Quantum Performance Laboratory, Sandia National Laboratories, USA

NATHAN E. MILLER, Lincoln Laboratory, Massachusetts Institute of Technology, USA

YILUN XU and GANG HUANG, Lawrence Berkeley National Laboratory, USA

IRFAN SIDDIQI, University of California, Berkeley, USA and Lawrence Berkeley National Laboratory, USA

JOHN LANGE, Oak Ridge National Laboratory, USA and University of Pittsburgh, USA

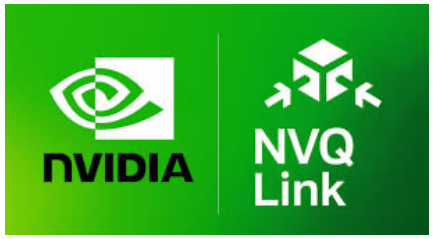
CHRISTOPHER ZIMMER and TRAVIS HUMBLE, Oak Ridge National Laboratory, USA

We propose an architecture, called NVQLink, for connecting high-performance computing (HPC) resources to the control system of a quantum processing unit (QPU) to accelerate workloads necessary to the operation of the QPU. We aim to support every physical modality of QPU and every type of QPU system controller (QSC). The HPC resource is optimized for real-time (latency-bounded) processing on tasks with latency tolerances of tens of microseconds. The network connecting the HPC and QSC is implemented on commercially available Ethernet and can be adopted relatively easily by QPU and QSC builders, and we report a round-trip latency measurement of 3.96 µs (max) with prospects of further optimization. We describe an extension to the CUDA-Q programming model and runtime architecture to support real-time callbacks and data marshaling between the HPC and QSC. By doing so, NVQLink extends heterogeneous, kernel-based programming to the QSC, allowing the programmer to address CPU, GPU, and FPGA subsystems in the QSC, all in the same C++ program, avoiding the use of a performance-limiting HTTP interface. We provide a pattern for QSC builders to integrate with this architecture by making use of multi-level intermediate representation dialects and progressive lowering to encapsulate QSC code.

Additional Key Words and Phrases: Quantum Computing, High-performance computing, System Architecture

Authors' Contact Information: Shane A. Caldwell, scaldwell@nvidia.com; Moein Khazraee; Elena Agostini; Tom Lassiter; Corey Simpson; Omri Kahalon; Mrudula Kanuri; Jin-Sung Kim; Sam Stanwyck; Muyuan Li; Jan Olle; Christopher Chamberland; Ben Howe; Bruno Schmitt; Justin G. Lietz; Alex McCaskey, NVIDIA Corporation, Santa Clara, California, USA; Jun Ye, Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore, Singapore, Singapore; Ang Li, Pacific Northwest National Laboratory, Richland, Washington, USA and University of Washington, Seattle, Washington, USA; Alicia B. Magann; Corey I. Ostrove; Kenneth Rudinger; Robin Blume-Kohout; Kevin Young, Quantum Performance Laboratory, Sandia National Laboratories, Albuquerque, New Mexico, USA; Nathan E. Miller, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, USA; Yilun Xu; Gang Huang, Lawrence Berkeley National Laboratory, Berkeley, California, USA; Irfan Siddiqi, University of California, Berkeley, Berkeley, California, USA and Lawrence Berkeley National Laboratory, Berkeley, California, USA; John Lange, Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA and University of Pittsburgh, Pittsburgh, Pennsylvania, USA; Christopher Zimmer; Travis Humble, Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA.

NVQLink's growing ecosystem



ALICE & BOB · Anyon Technologies · atom computing · DELL Technologies · diraq · equal1

IONQ · IQM · Infleqtion · KEYSIGHT · ORCA COMPUTING · Pasqal

ORQC · QBLOX · BERKELEY LAB QubiC · QUANTINUUM · QUANDELA · qci Quantum Circuits, Inc.

QUANTUM MACHINES · QUANTUM MOTION · QuEL, Inc. · QuEra Computing Inc. · rigetti · seeqc

QICK · sdt · Silicon Quantum Computing · wistron · Zurich Instruments

ATPESC quantum day

Wed, Aug 5

8:30 a.m.

Track 7: Quantum Computing

Welcome, Framing, and Expectations

Laura Schulz, Argonne National Laboratory

oneHPC

	Simulation	Analytics	Learning
QPU	Variational Quantum algs; Trotterization	Quantum Fourier transform	Quantum ML; Data generation
GPU	Structured-data apps ("throughput friendly")		Training
CPU	Unstructured-data apps ("latency tolerant")		Inference

oneCSE

	to Simulation	to Analytics	to Learning
Simulation provides	–	Physics-based "regularization"	Data for training (augmenting real-world data)
Big Data Analytics provides	Steering in high dimensional parameter space; *In situ* processing	–	Cleaned feature vectors for training
Machine Learning provides	Replacement of models with learned functions; Smart data compression	Detection and classification; Imputation of missing data	–


HPFC

Extreme computing universals

Reside high on the memory hierarchy	Memory
Reduce message number and volume Reduce frequency of synchronization Reduce span of synchronization	Networks
Embrace heterogeneity	Processors
Exploit high-order discretizations Exploit hierarchical representations Exploit data sparsity Exploit nested levels of parallelism Exercise the "right to reorder" Employ dynamic schedule and balancing Co-design algorithms with hardware Take resilience into algorithms Process on the fly	Algorithms
Code to specialized backends Avoid over-resolving and over-solving Maximize "science per Joule" Reformulate before computing	Users

HPC vibe coding prompt #1

```
Reside high on the memory hierarchy
```

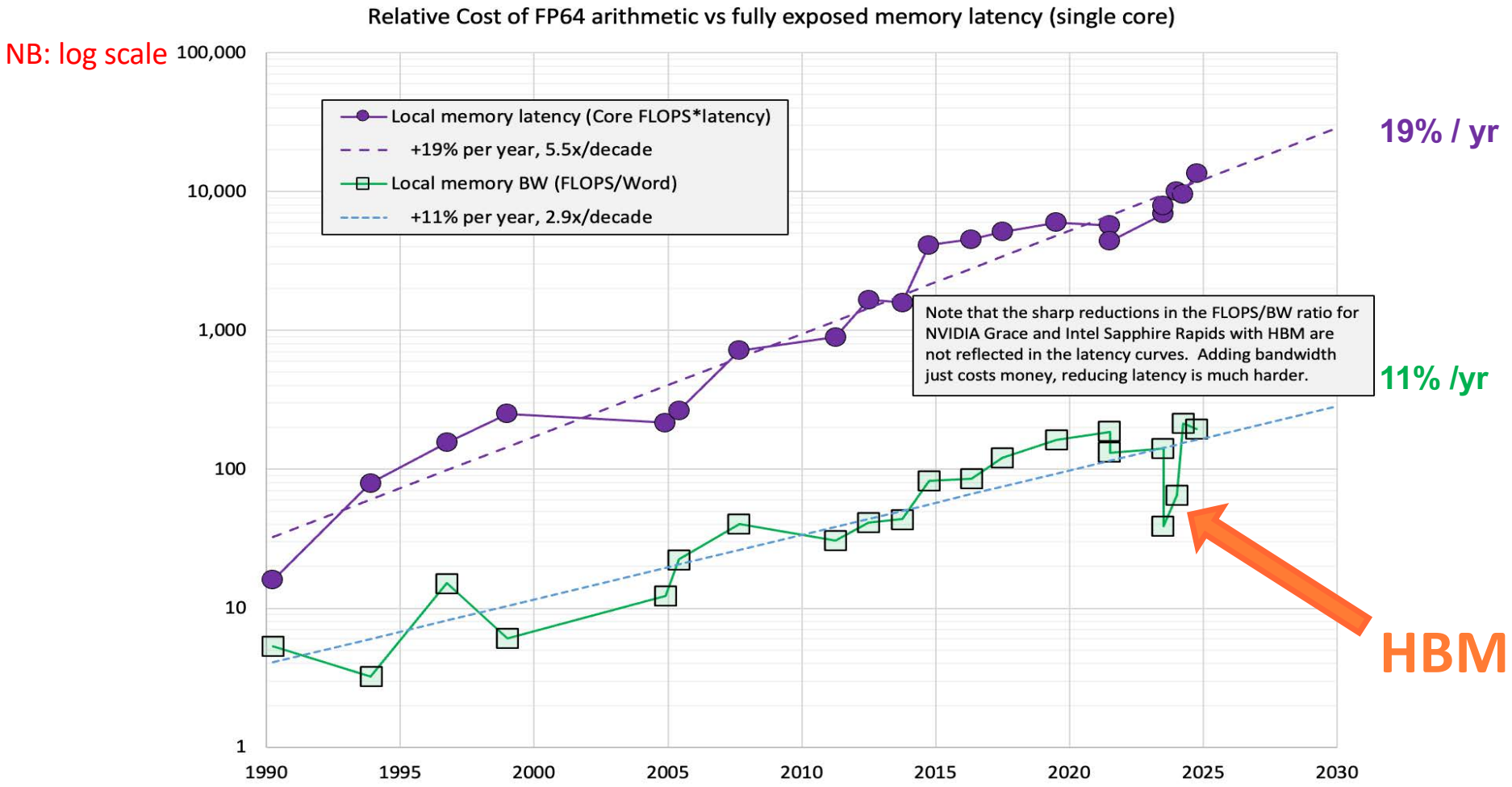
Why – latest hardware trend reports from John McCalpin & Keren Bergman

How – *e.g.*, hierarchical matrices

For each imperative, I'll give an example or two of a "Why" and a "How".
At the end of the talk, I'll invite you to supply a better set of examples, for the next version.

[Memory system]

Single core speeds/feeds ratios, 1990-2025

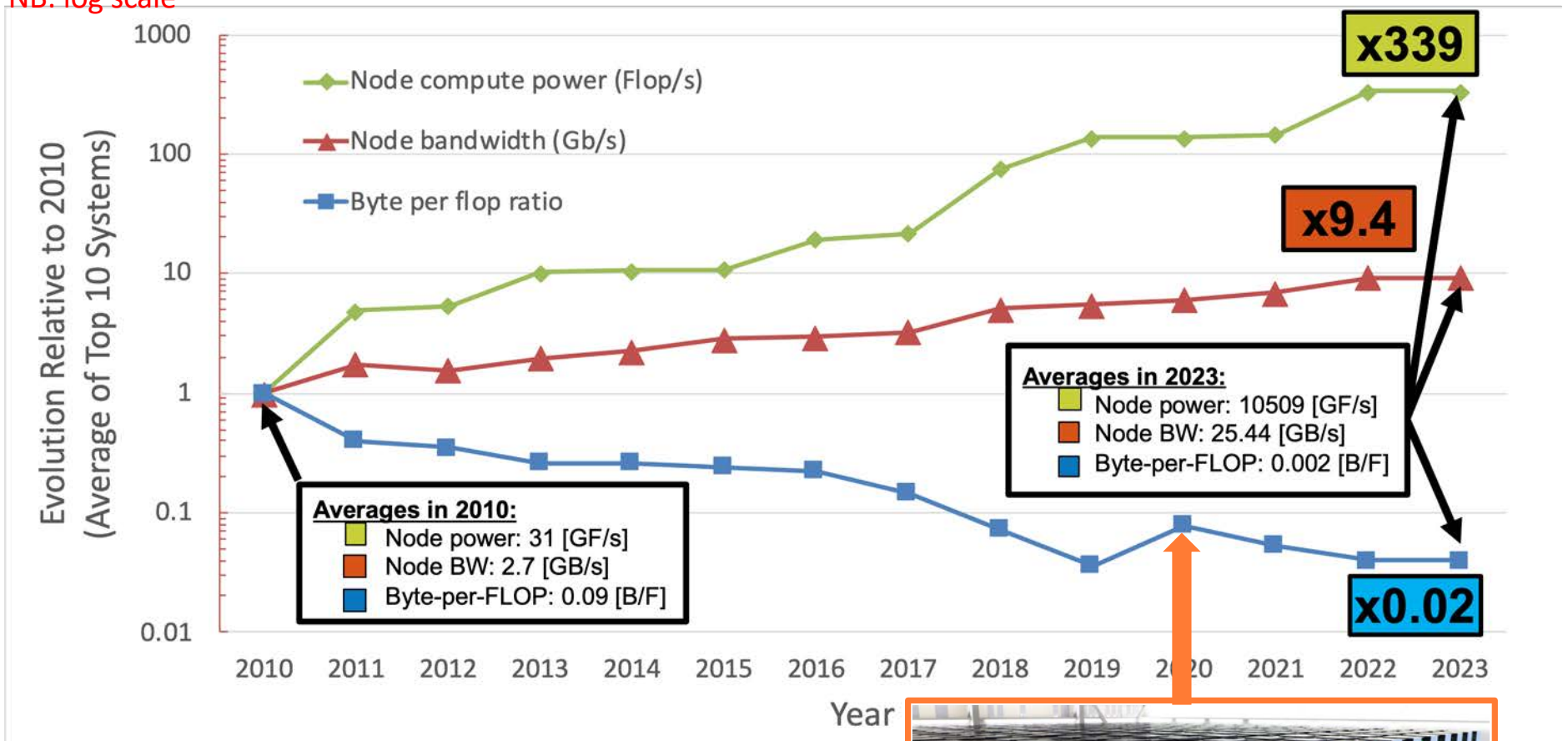


John McCalpin, now at BSC, has been tracking architectural trends through the STREAM benchmark since 1990, when he noticed that code loops that gave 90% peak on Cray gave less than 10% on RISC

- **On-node machine balance for BW and latency**
- **Dimensionless BW imbalance based on GF/s for one core divided by on-node sustainable BW**
- **Dimensionless latency based on GF/s for one core times latency for load from local memory**

HPL Top 10 memory BW trends, 2010-2023

NB: log scale



Today's *HPCWire* named Karen one of the Legends of HPC

Keren Bergman's lab at Columbia has been tracking architectural trends in memory and networking interconnects for two decades. This slide is updated thru 2023. Trends continue, but absolute performances are missing Chinese exascale entries.

Shrink footprint to reside high in the hierarchy

For a square dense matrix of $O(N)$:

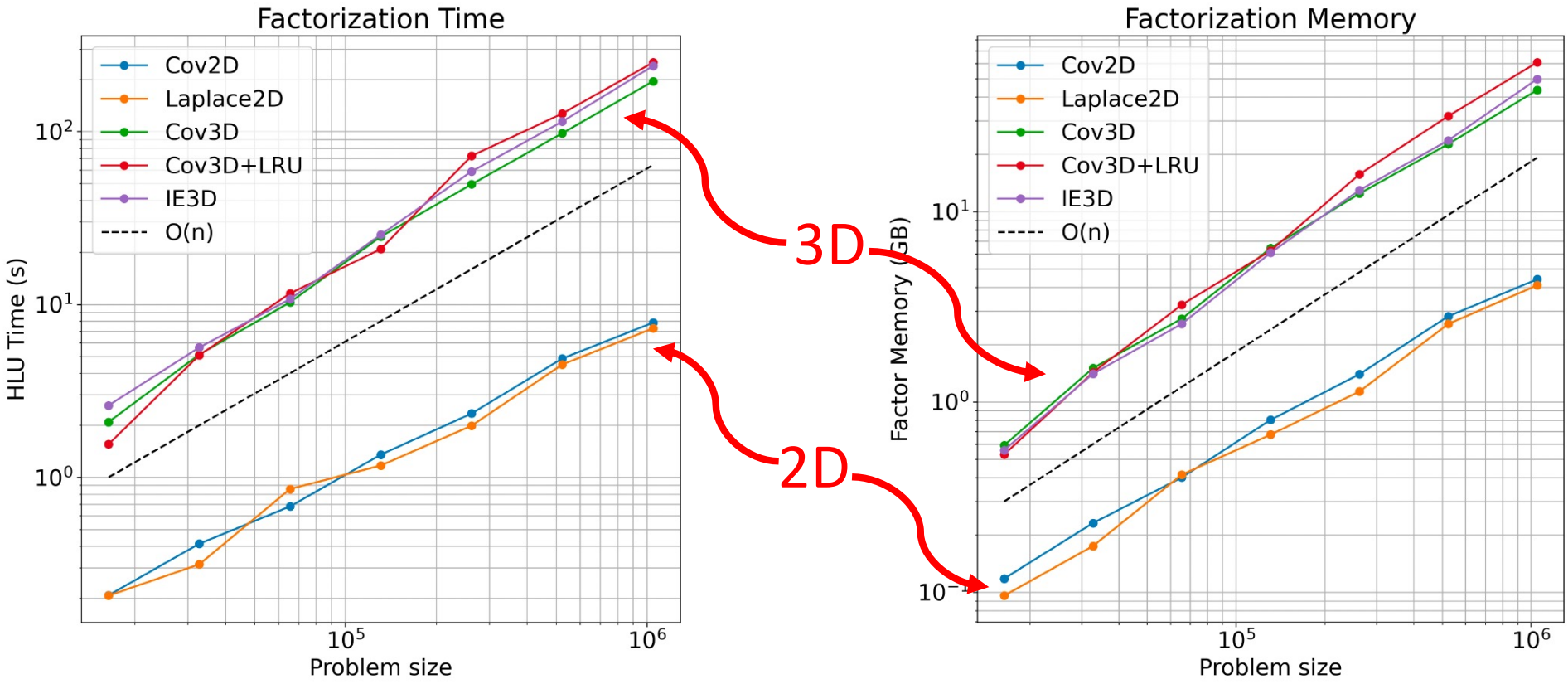
- "Straight" LU or LDL^{T}
  - Operations $O(N^3)$
  - Storage $O(N^2)$

Yang Liu's talk yesterday

- BLock low-rank (Amestoy, Buttari, L'Excellent & Mary, *SISC*, 2016)*
  - Operations $O(k^{0.5} N^2)$
  - Storage $O(k^{0.5} N^{1.5})$
  - for uniform blocks with size chosen optimally for max rank k of any compressed block, bounded number of uncompressed blocks per row
- Hierarchically low-rank (Grasedyck & Hackbusch, *Computing*, 2003)
  - Operations $O(k^2 N \log^2 N)$
  - Storage $O(k N)$ ← meet hierarchy with hierachy
  - for strong admissibility, where k is max rank of any compressed block

* First reported $O(k^{0.5} N^{2.5})$, then later $O(k^{0.5} N^2)$ for variant that reorders updates and recompression

Linear memory (& time) scaling with $\mathcal{H}^2$



(a) Log-log plot of the factorization time for various problem sizes showing linear growth for factorization.

(b) Log-log plot of the memory consumed by the factors of various problems showing linear growth in memory.

FIG. 13. The factorization algorithm exhibits linear complexity in time and memory.

c/o W. Boukaram (KAUST) *et al.*, IMA JNA special issue in memory of Nick Higham (2026)

HPC vibe coding prompt #2

```
Reduce message number and volume
```

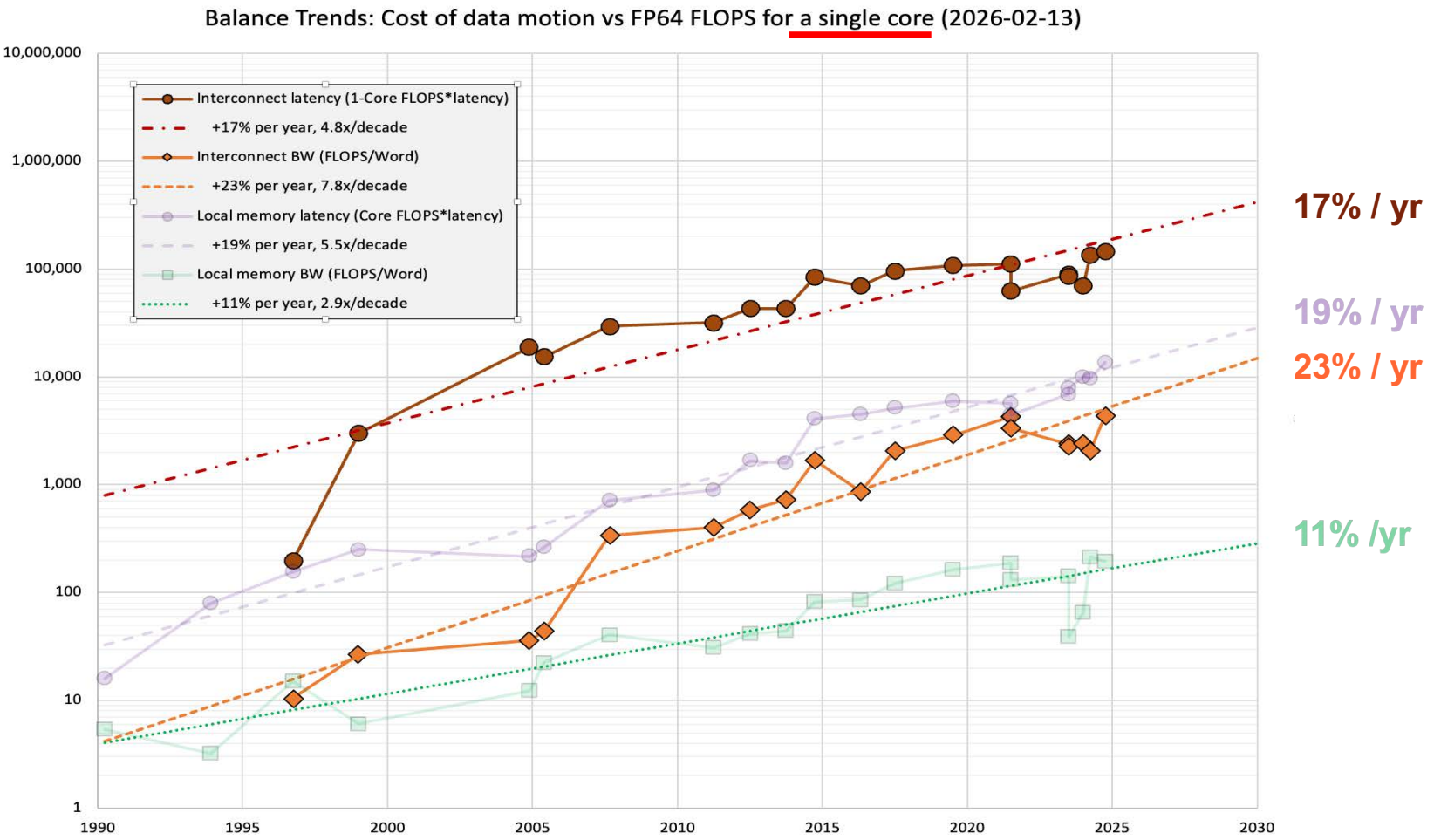
Why – more trends (off-node) from John McCalpin

How – *e.g.*, 2.5D matrix algorithms

[Network]

Off-node speeds/feeds ratios, 1990-2025

NB: log scale



Another chart from John McCalpin superposing *network* BW and latency over *on-node* BW and latency (previous chart, faded)

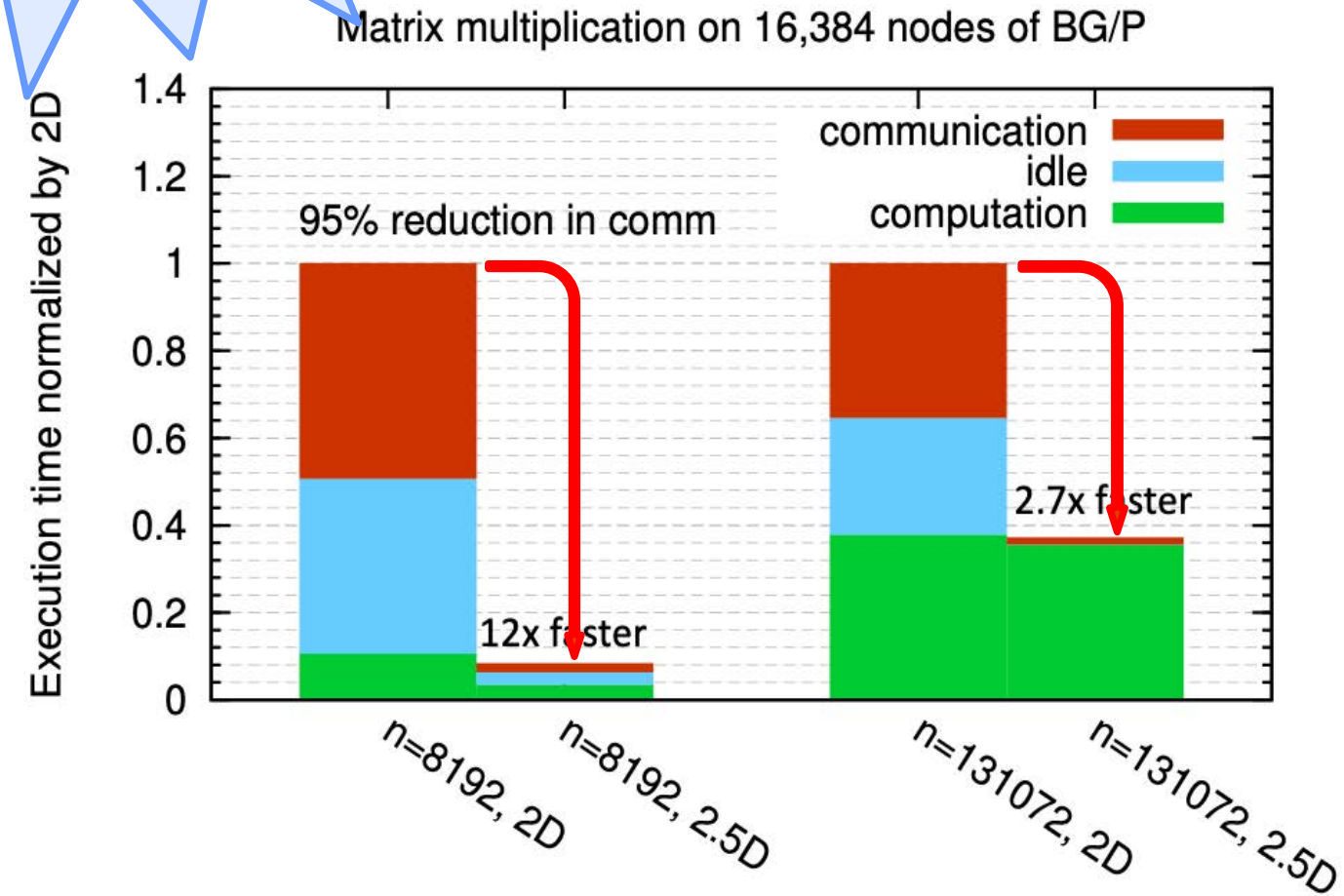
- **Off-node machine balance for BW and latency**
- **Dimensionless network imbalance based on GF/s for 1 core divided by interconnect sustainable BW**
- **Dimensionless network latency based on GF/s for 1 core times latency for get from remote memory**

Use extra memory to reduce communication

Jim Demmel's talk Monday

Communication volume can be reduced *at a cost of greater memory per node* in so-called "2.5D matrix algorithms" for LU, Cholesky, QR, Gram-Schmidt, and eigensolvers.

Many implementations attain the theoretical lower bounds.



c/o E. Solomonik & J. Demmel (Berkeley), EuroPar 2011 (Distinguished Paper Prize)

HPC vibe coding prompt #3

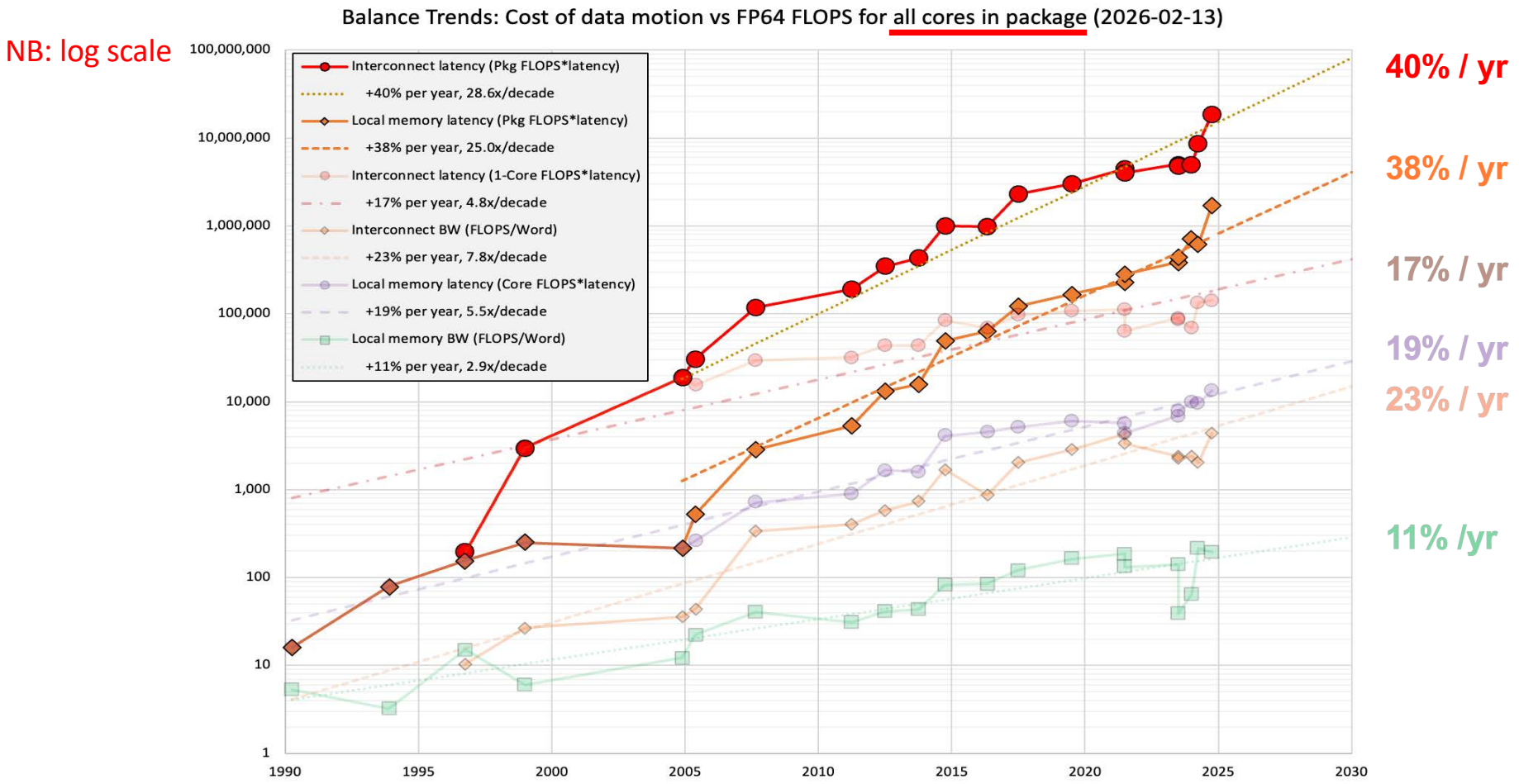
```
Reduce frequency of synchronization
```

Why – McCalpin off-node trends previously cited

How – *e.g.*, bundled reductions in conjugate gradients

[Network]

Off-node speeds/feeds ratios, 1990-2025



Another chart from John McCalpin superposing network BW and latency over on-node *for all cores in a package,* superposed over previous trends (faded)

- **On-node and off-node machine balance for latency based on *package flop/s***
- **Dimensionless latency based on GF/s for *all cores* time latency for load from local memory**
- **Dimensionless latency based on GF/s for *all cores* times latency for get from remote memory**

Perform extra local flops to synchronize less

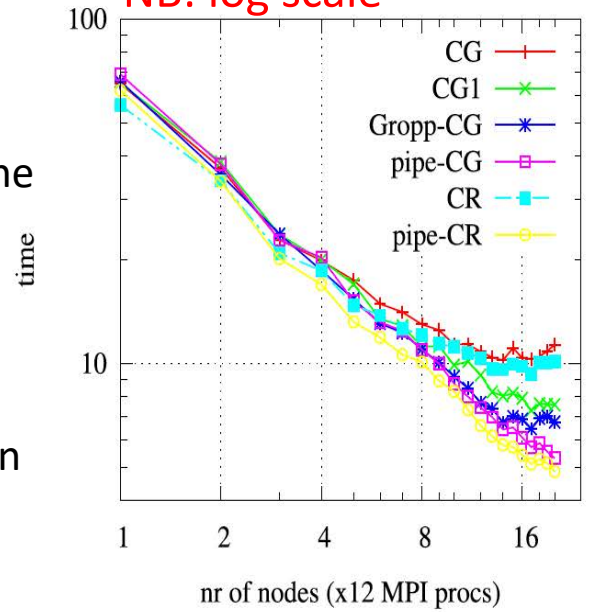
Table 1
Overview of the different preconditioned CG and CR variations. Column *flops* lists the number of flops ($\times N$) for AXPYS and dot-products. The *time* column has the time spent in global all-reduce communication (G), in the matrix–vector product (SPMV) and the preconditioner (PC). Column *#global synchronizations* has the number of global communication phases per iteration. The *memory* column counts the number of vectors that need to be kept in memory (excluding x and b).

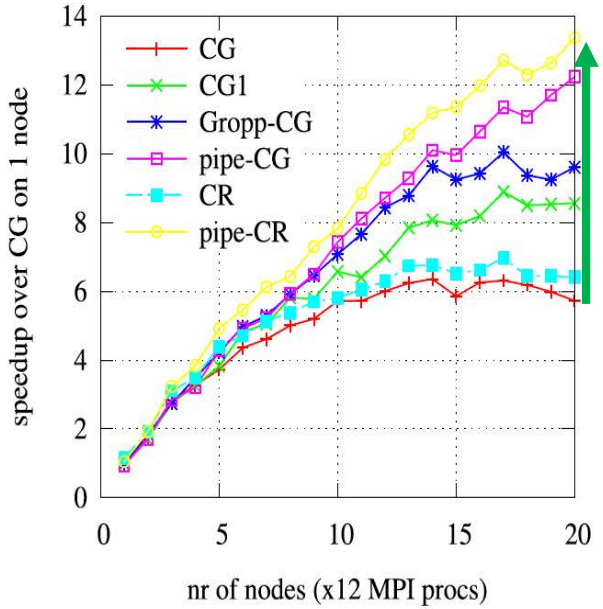
	Flops	Time (excl, AXPYS, DOTS)	#Glob syncs	Memory
CG	10	2G + SpMV + PC	2	4
Chron/Gear-CG	12	G + SpMV + PC	1	5
CR	12	2G + SpMV + PC	2	5
Pipe-CG	20	max(G, SpMV + PC)	1	9
Pipe-CR	16	max(G, SpMV) + PC	1	7
Gropp-CG	14	max(G, SpMV) + max(G, PC)	2	6

Frequency of inner products in preconditioned conjugate gradients can be reduced at the cost of greater arithmetic per synchronization and greater storage

Savings up to 2.5x on 3D hydrostatic ice sheet flow app

NB: log scale





Speedup of 2.25x, despite 2x flops 2x mem

Fig. 3. Left: Time to solution for the 3D hydrostatic ice sheet flow simulation using $100 \times 100 \times 50$ Q1 finite elements. Right: Speedup as function of number of nodes over standard CG solver on a single node.

c/o P. Ghysels (LBNL) & W. Vanroose, Par. Comp. (2014)

HPC vibe coding prompt #4

```
Reduce span of synchronization
```

Why – McCalpin off-node trends previously cited
How – *e.g.*, linear / nonlinear Schwarz preconditioning for PDEs

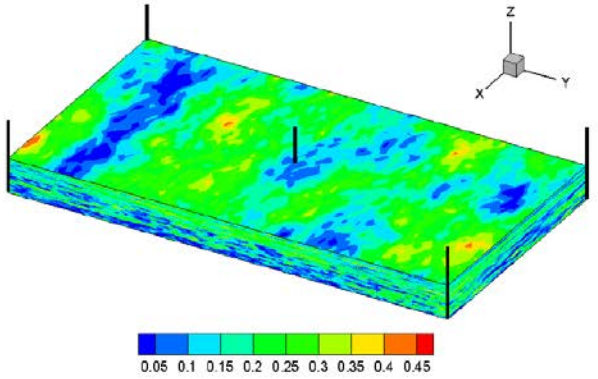
[Network]

Perform extra inner solves to reduce outer

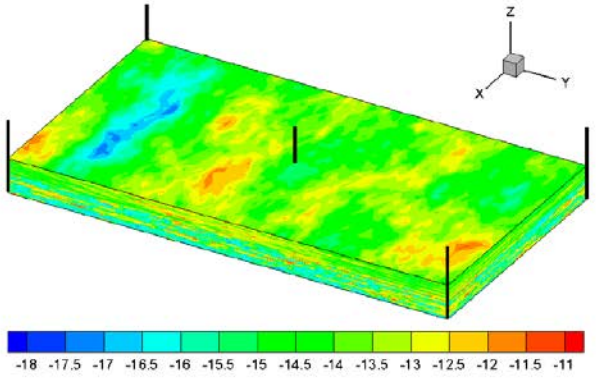
Number of global Newton iterations in a Newton-Krylov solver, each step with global preconditioned GMRES iterations, can be reduced by nonlinear preconditioning – solving Newton-Krylov inner problems on subsets of the equations

[left] Convergence for global Newton solves at various implicit timesteps

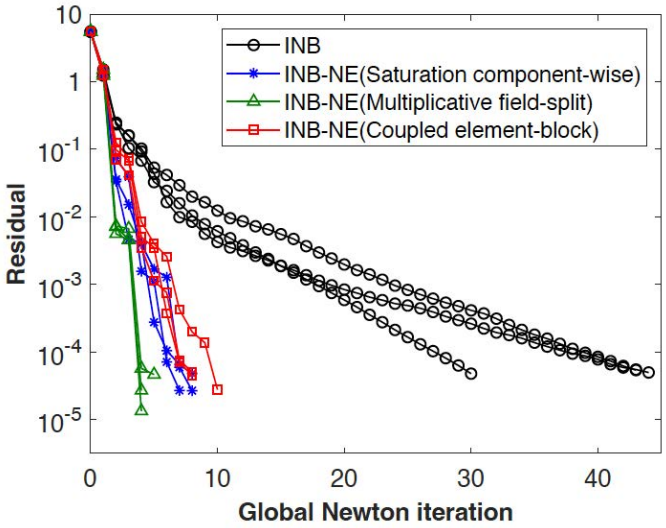
[right] Overall savings up 2.5x on 3D SPE10 multiphase reservoir flow in strong scaling



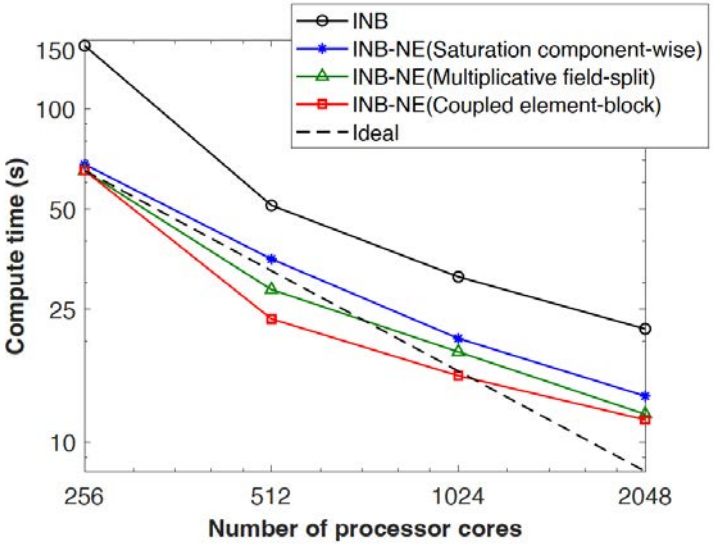
(a) Porosity



(b) Permeability ($\log_{10} K_{yy}$)



improved Newton convergence



improved strong-scaled runtime

c/o L. Luo (KAUST) *et al.*, SISC (2021)

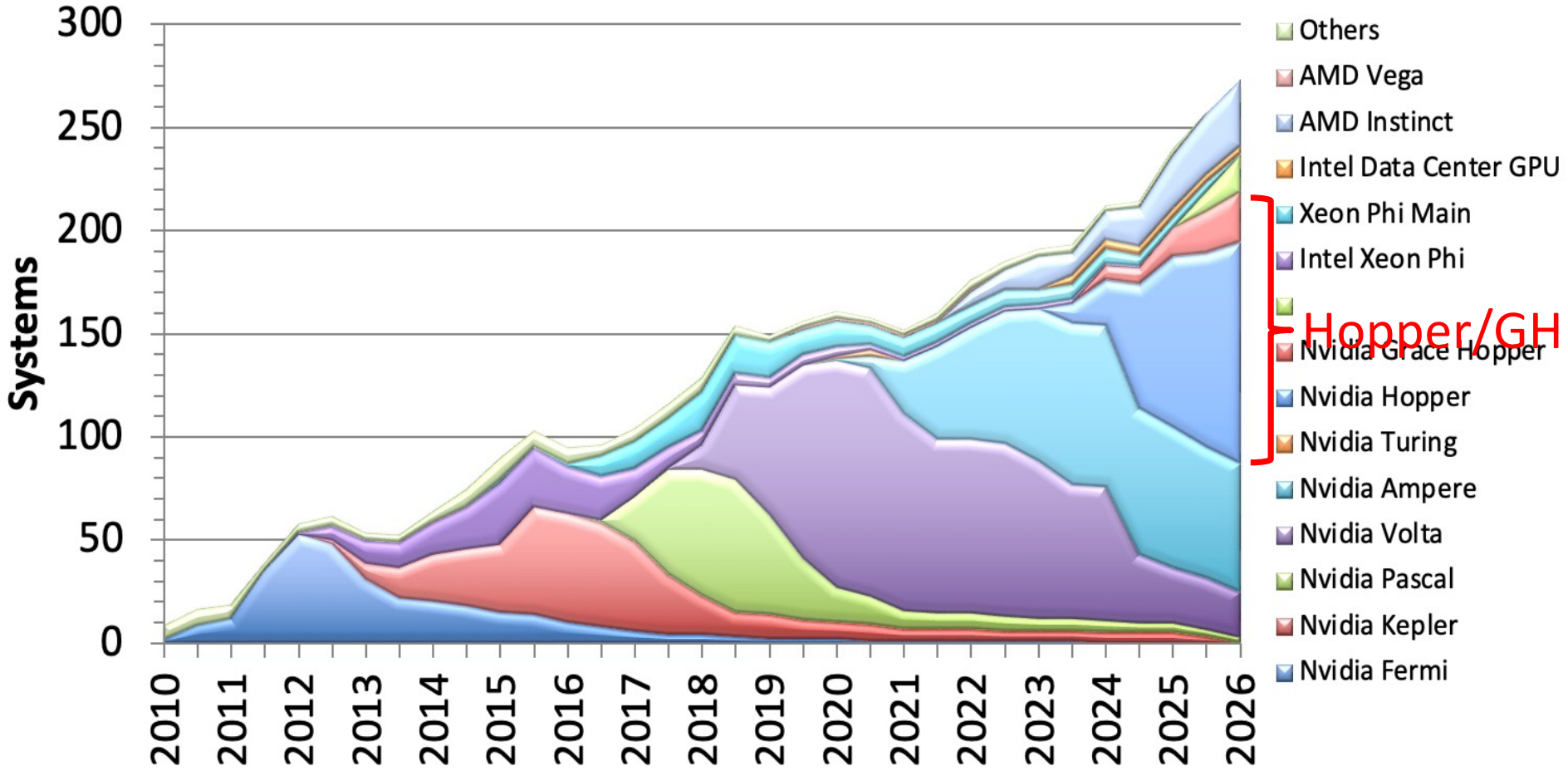
HPC vibe coding prompt #5

```
Embrace heterogeneity
```

Why – it's everywhere, even where we didn't ask for it (*e.g.*, multiple generations of GPUs in one system)
How – *e.g.*, Variational Quantum Eigensolver for molecular structure

[Processing]

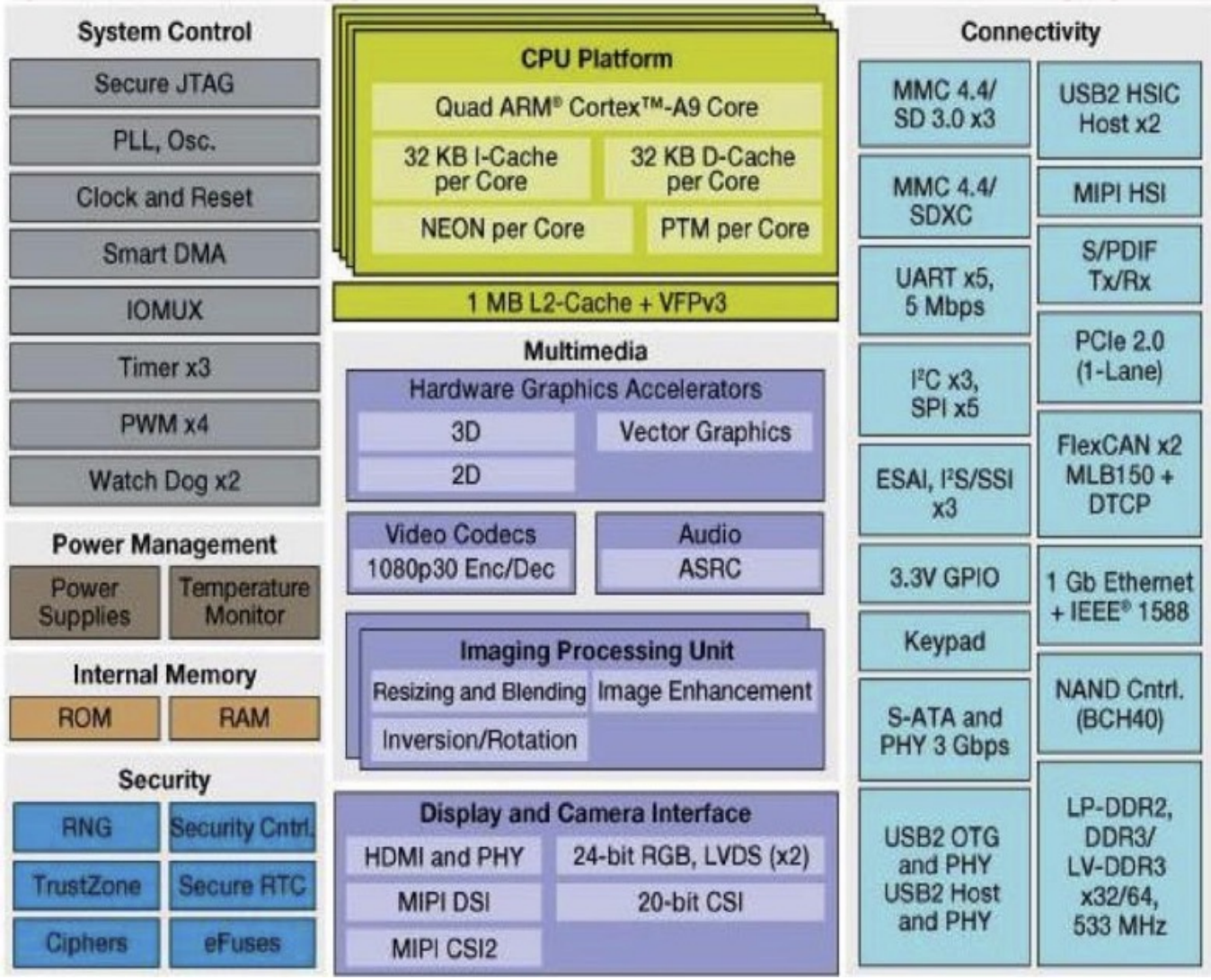
Heterogeneity has taken over (top of) Top500



- About 50% of the Top500 (June 2026) systems exploit accelerators
- Only 2 of Top20 lacks accelerators, only 4 of Top50

c/o Erich Strohmeier (LBNL, June 2026)

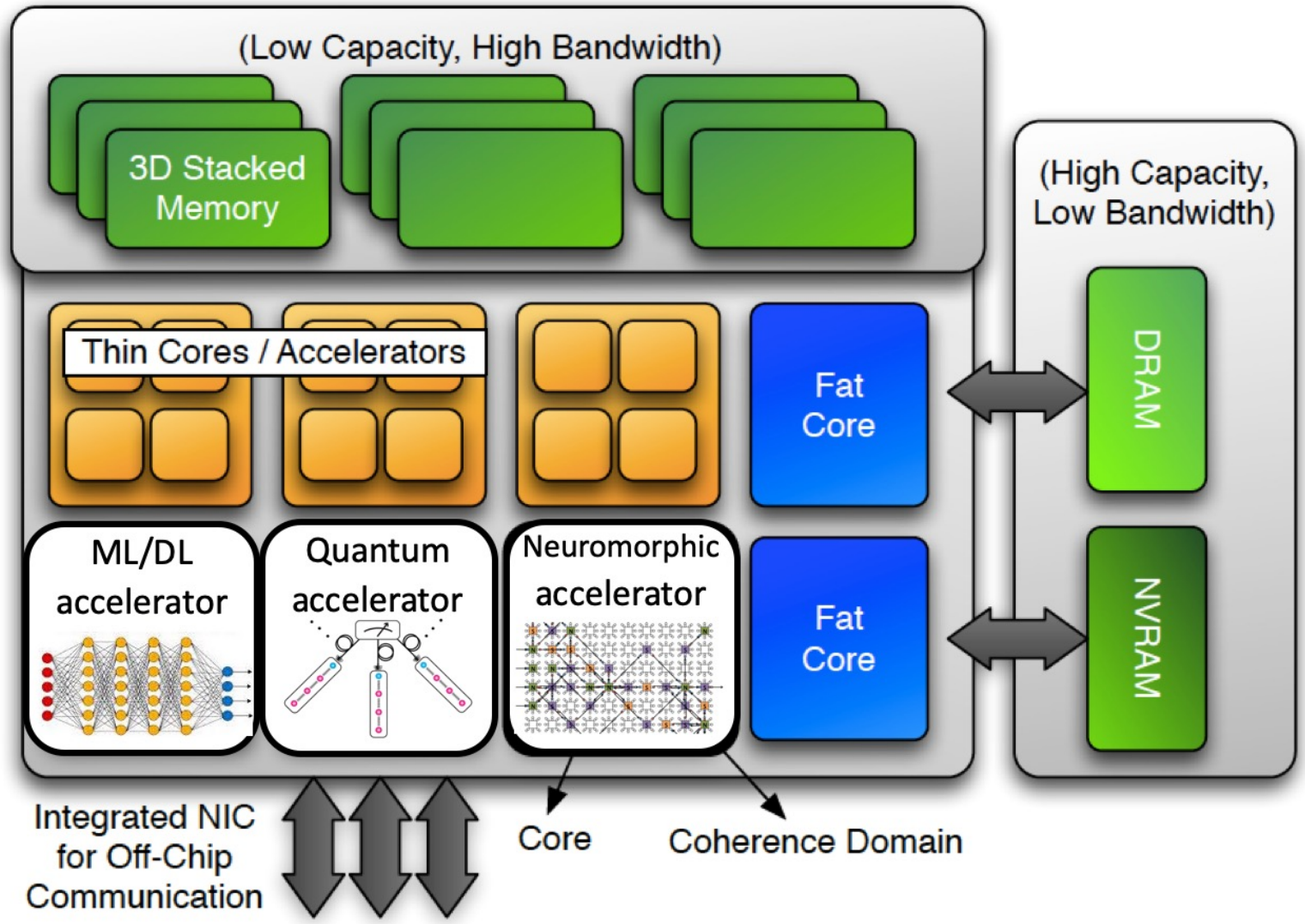
Heterogeneity in your smart phones



Typical smart phone has 40+ special processors

c/o John Shalf (LBNL, 2021)

Extrapolating heterogeneity to HPC



modified, after J. Ang (Sandia) *et al.* (2014)

“Quantum First” philosophy

Computing workflows will cascade:

QPUs will be used everywhere they have advantage

GPUs will be used for everything else with high arithmetic intensity and structured memory accesses

CPUs will be used for everything else with low arithmetic intensity and unstructured memory accesses

Efficiently assigning the resources of a QPU-GPU-CPU hybrid supercomputer to multiple jobs will be a rich programming challenge (see best paper at ISC’25 and plenary at ISC’26, Hamburg)

Keynote at ISC'26 (June 2026, Hamburg)

MARTIN SCHULZ
Professor in Computer Engineering, TUM & Member of the Board of Directors at LRZ, Germany

HPC: A HETEROGENEOUS FUTURE

High performance computing (HPC) has long been driven by relentless demand for more computational power, enabling breakthroughs across science and engineering. For decades, Moore's Law and Dennard Scaling provided a predictable path for performance growth but these trends have slowed dramatically, exposing fundamental challenges that go beyond transistor density. Issues such as data movement, memory bandwidth, and energy efficiency have become dominant bottlenecks, making incremental improvements in raw computational power insufficient. GPUs, while powerful, are not a fundamental departure from traditional architectures. As a result the need for transformative innovation has never been greater.

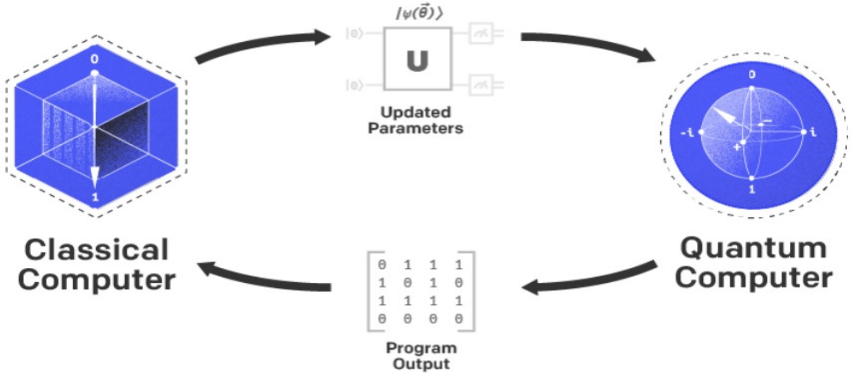
READ FULL ABSTRACT

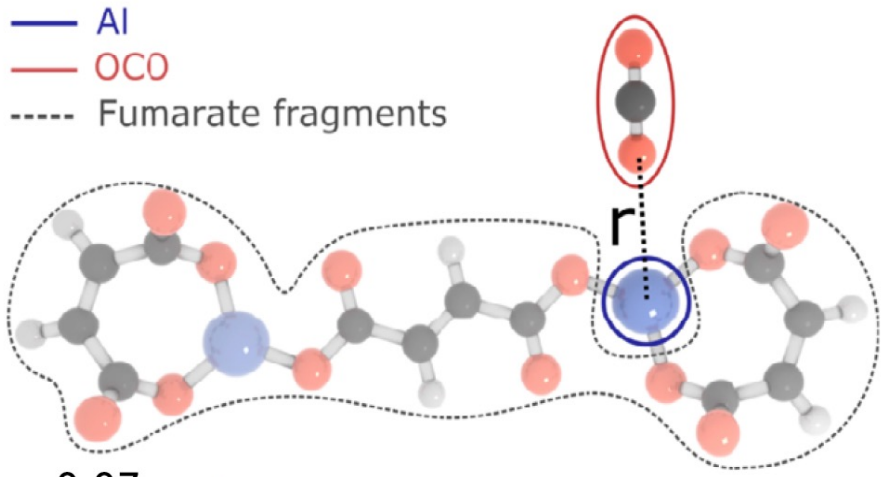
The post-Moore era calls for a paradigm shift that pushes algorithmic advances, but also novel hardware and software approaches. Emerging technologies – quantum computing, neuromorphic architectures, photonics, and others – offer promising acceleration capabilities, but each introduces its own ecosystem, challenges and complexity. To harness these innovations effectively, HPC must evolve toward integrated, heterogeneous environments built on top of unified software stacks that ensure programmability, portability, and accessibility for domain scientists.

This keynote will showcase examples of emerging technologies and their integration into the HPC ecosystem. A concrete case will highlight quantum acceleration through an HPC-QC software stack and hybrid workflows. Building on this, the talk will outline strategies for creating a unified, heterogeneous environment that turns diversity in architectures—from quantum to neuromorphic – into a strength rather than a barrier.

- Update from the best paper of ISC'25

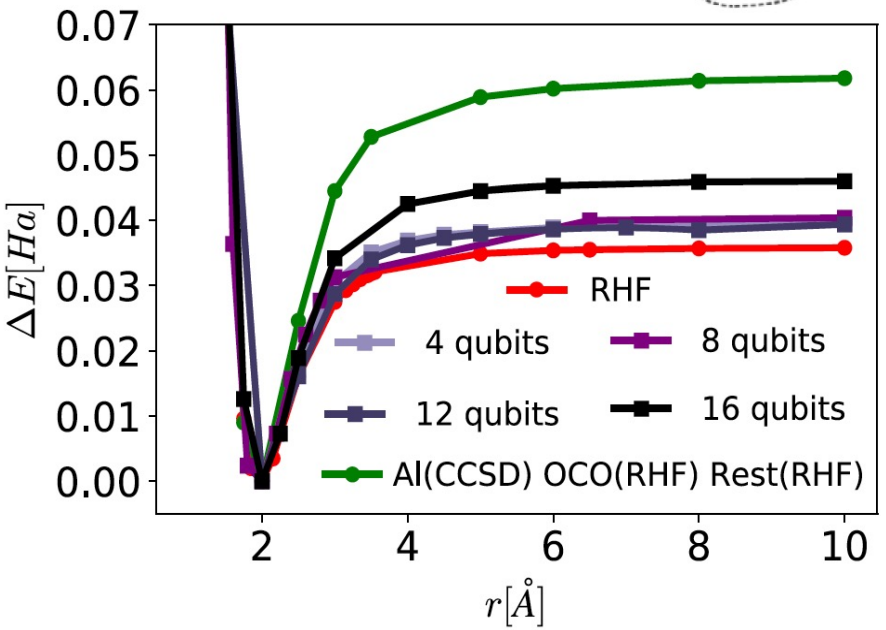
QPU-HPC integration





A variational quantum eigensolver (VQE) alternates between an optimizer run on a classical computer and a quantum circuit that evaluates the energy of a proposed configuration, using each resource to its advantage.

For a building block of a metal-organic framework (MOF) designed to capture CO2, convergence to the accepted curve (green) in the number of qubits is shown at right.



QUANTINUUM TotalEnergies

c/o G. Greene-Diniz (Quantinuum) *et al.*, EPJ Quantum Tech (2022)

HPC vibe coding prompt #6

```
Exploit high-order discretizations
```

Why – better arithmetic intensity and smaller memory for the comparable accuracy

How – replace many low-order elements with fewer high-order elements

[Algorithms]

Use high-order discretizations for fewer DOFs

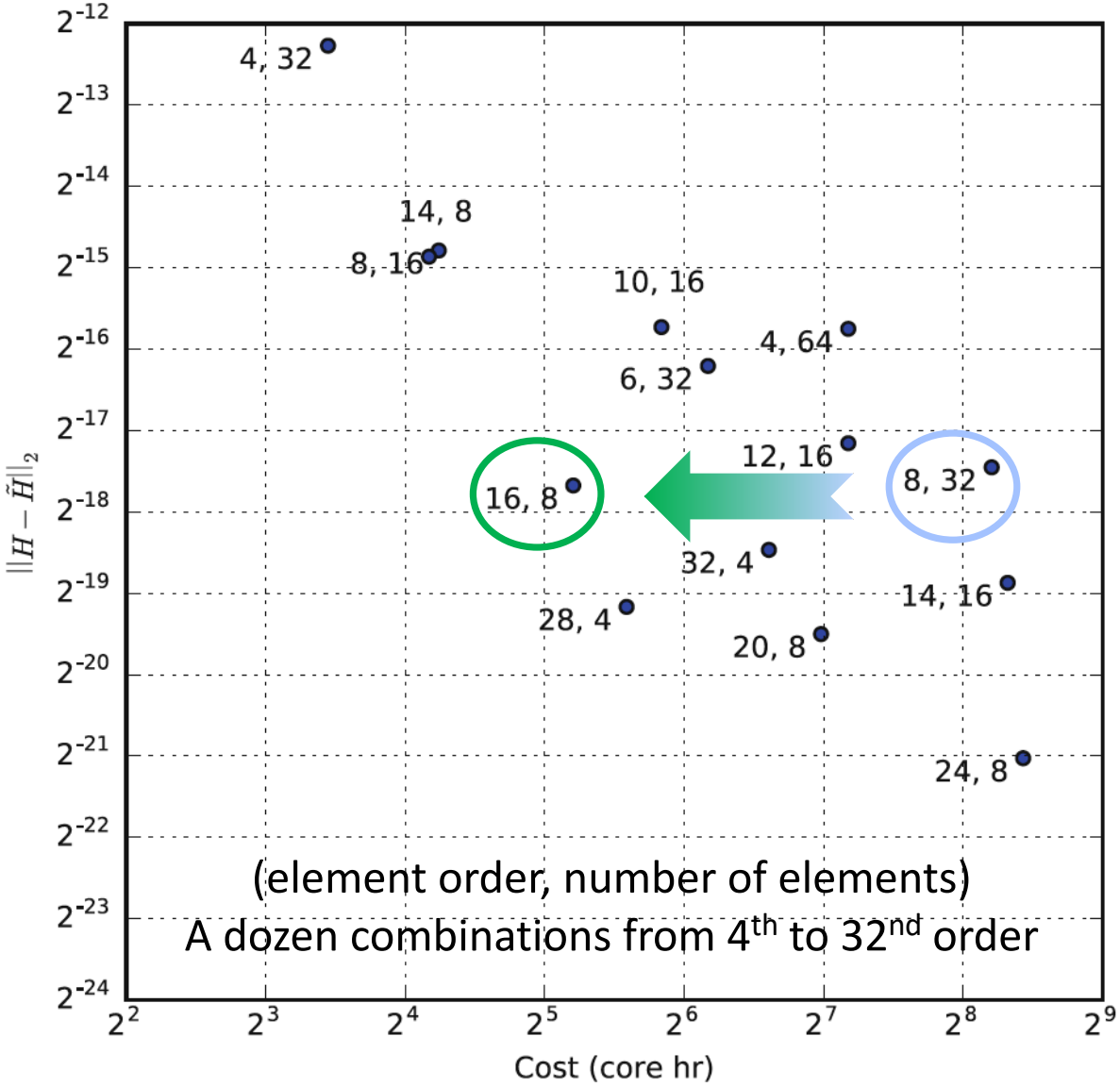
Raising discretization order offers a double win:

- more flops per degree of freedom
- many fewer degrees of freedom

Incompressible Navier-Stokes re-discretized from, e.g., ***32 spectral elements of order 8*** to ***8 spectral elements of order 16*** per dimension

<u>Same error in key functional:</u>
- approx 4e-6

<u>Savings in execution time:</u>
- factor of 8



c/o M. Hutchinson (U Chicago) *et al.*, ISC (2016)

HPC vibe coding prompt #7

```
Exploit hierarchical representations
```

Why – Moore's Law adjusts only the constants; algorithms improve exponents

How – to weak scale to extremes, one must use algorithms with optimal asymptotic complexity, $O(N \log^p N)$

These are typically recursively hierarchical. Through the decades…

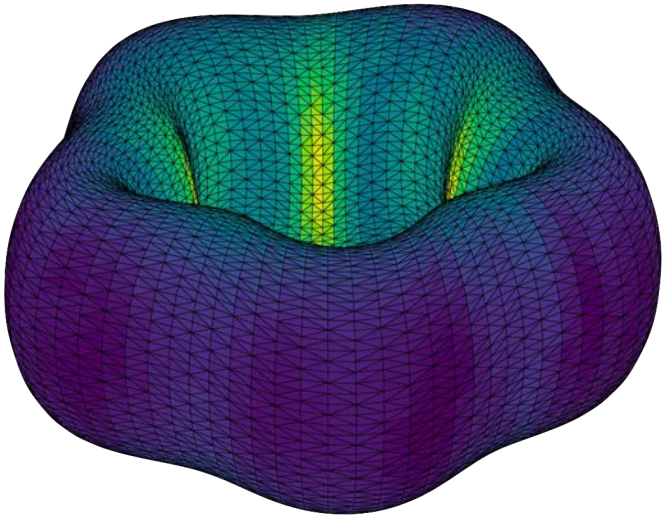
- Fast Fourier Transform (1960's): $N^2 \rightarrow N \log N$
- Multigrid (1970's): $N^{4/3} \log N \rightarrow N$
- Fast Multipole (1980's): $N^2 \rightarrow N$
- Sparse Grids in $d > 1$ dimensions (1990's): $N^d \rightarrow N (\log N)^{d-1}$
- $\mathcal{H}$ matrices (2000's): $N^3 \rightarrow k^2 N (\log N)^2$
- Multilevel Monte Carlo (2000's): $N^{3/2} \rightarrow N (\log N)^2$
- Randomized matrix algorithms (2010's): $N^3 \rightarrow N^2 \log k$
- Tensor trains in $d > 1$ dimensions (2010's): $N^d \rightarrow d\, k^2 N$
- ??? (2020's): **??? → ???**

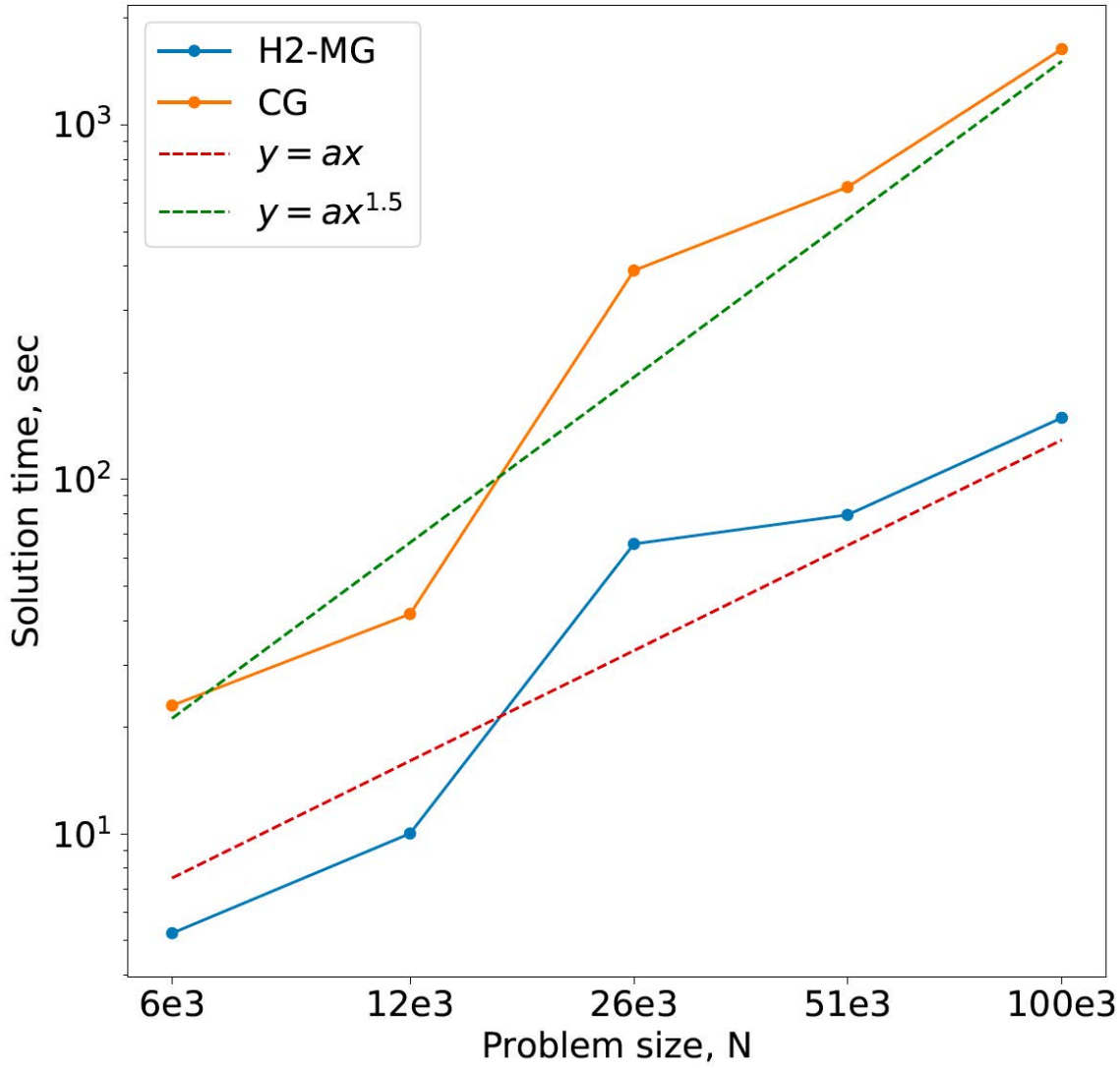
[Algorithms]

Combination: $\mathcal{H}^2$-Multigrid

Combining two optimal methods cannot improve the exponent, but can still improve the constant ☺

Boundary element problem for 3D electrostatic problem with single layer potential





c/o D. Shushnikova (KAUST) *et al.*, SIAM SISC (2026) – come to SIAM CSE'27 for GPU version

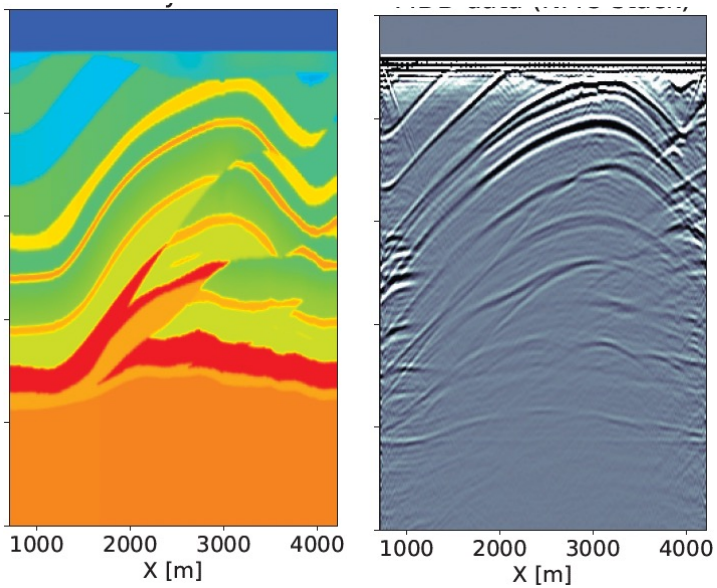
HPC vibe coding prompt #8

```
Exploit data sparsity
```

Why – see: “live high on the memory hierarchy”
How – *e.g.*, tile low rank matrix multiplication

[Algorithms]

Meet “curse of dimensionality” with “blessing of low rank”



Society for Exploration Geophysicists (SEG) ”overthrust” model for velocity of save propagation under the sea and inference using multidimensional deconvolution in the frequency domain, with 26K sources and 16K receivers

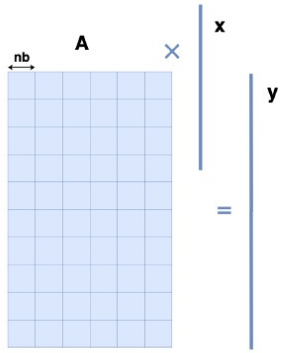


Fig. 2: Original dense MVM.

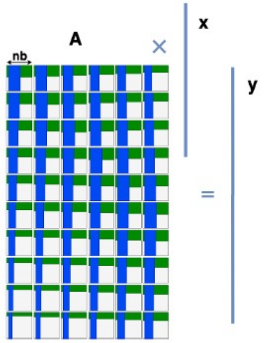


Fig. 3: Rank-compressed operator.

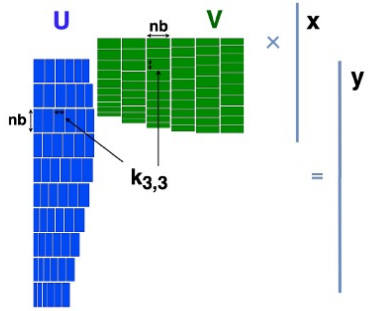


Fig. 4: Stacked bases U and V.

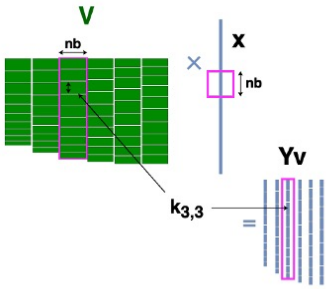


Fig. 5: V-batch stage of MVM.

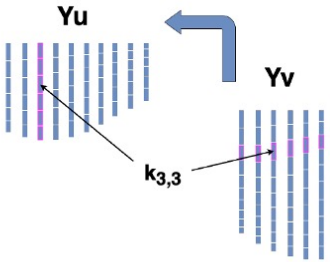


Fig. 6: Shuffle from V to U bases.

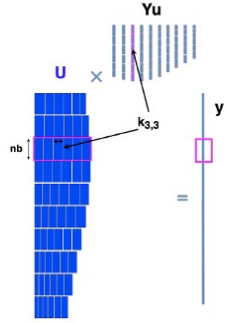


Fig. 7: U-batch of MVM.

Frequency domain source to receiver map in a seismic processing application, with tile low-rank decomposition to save memory for the all-SRAM Cerebras CS-2 wafer scale engine

c/o H. Ltaief (KAUST) *et al.*, SC’23 (Gordon Bell Finalist)

Cerebras CS-2 Wafer-Scale Engine

Cerebras went public on 14 May 2026 and their shares nearly doubled in the first day of trading.

HOW

Meet “curse of dimensionality” with “blessing of low rank”

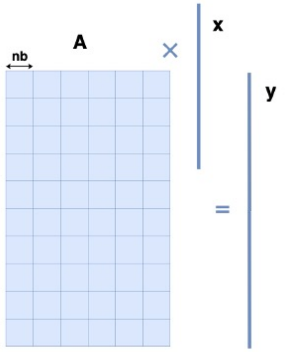


Fig. 2: Original dense MVM.

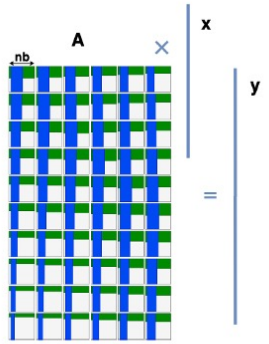


Fig. 3: Rank-compressed operator.

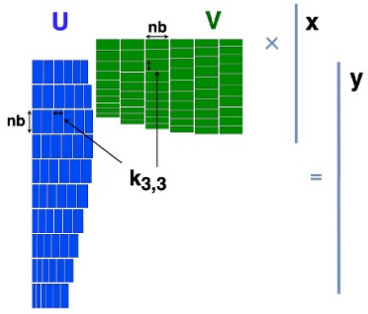


Fig. 4: Stacked bases U and V.

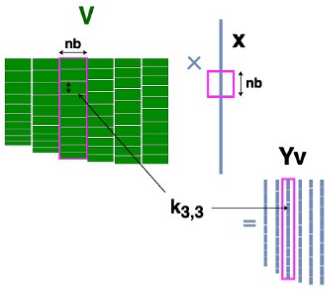


Fig. 5: V-batch stage of MVM.

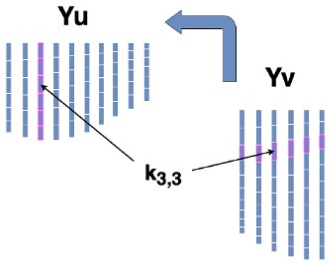


Fig. 6: Shuffle from V to U bases.

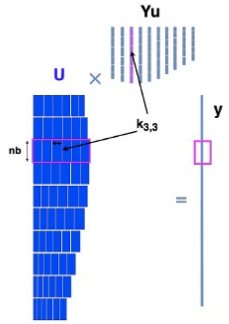
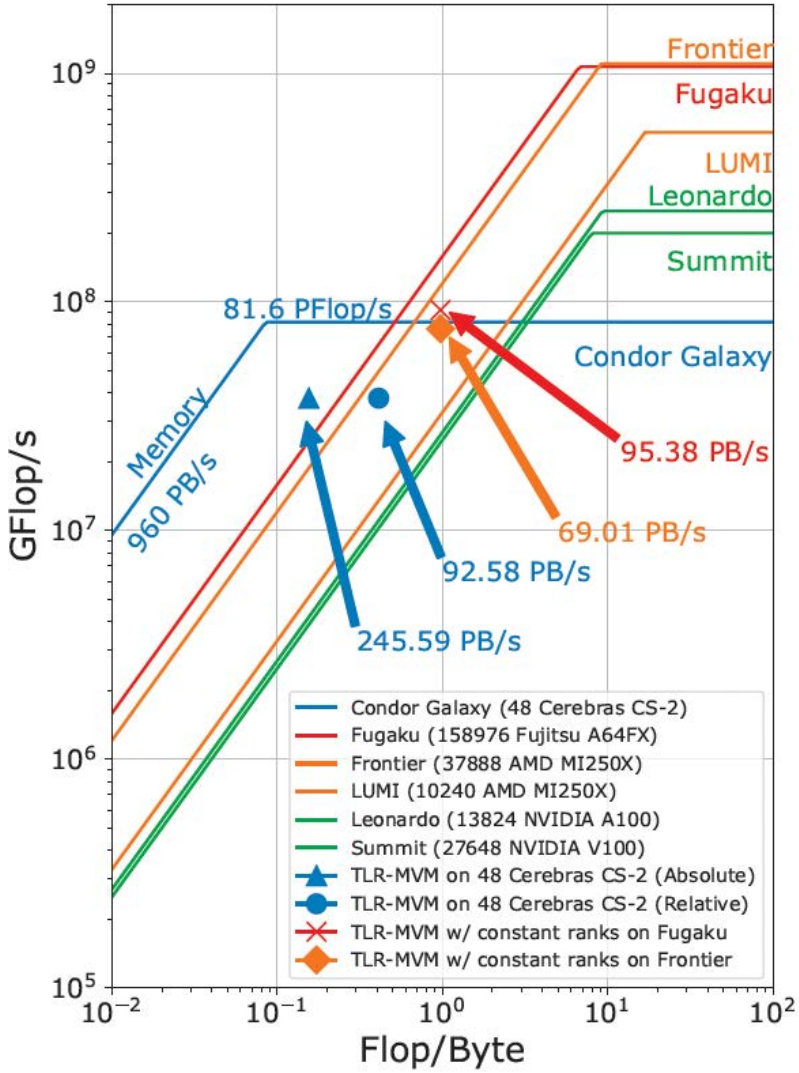


Fig. 7: U-batch of MVM.



Frequency domain source to receiver map in a seismic processing application, with tile low-rank decomposition to save memory for the all-SRAM Cerebras CS-2 wafer scale engine

c/o H. Ltaief (KAUST) *et al.*, SC’23 (Gordon Bell Finalist)

HPC vibe coding prompt #9

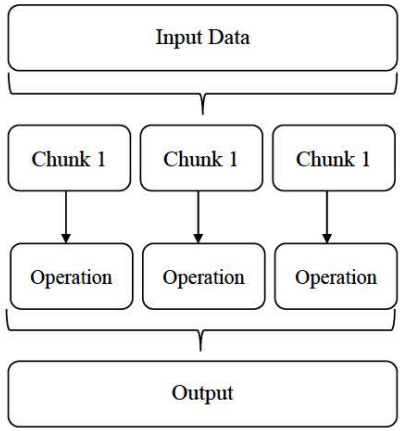
```
Exploit nested levels of parallelism
```

Why – exploit all available concurrency

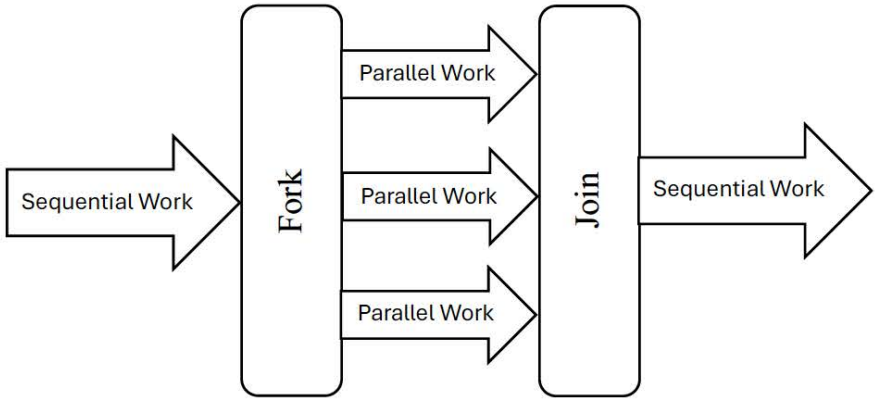
How – *e.g.*, wrap application already saturated in parallelism with swarm optimization (multiple independent instances)

[Algorithms]

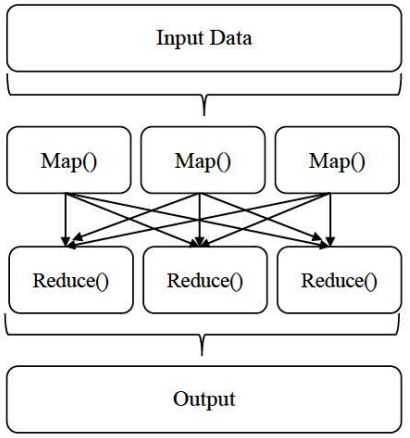
Parallel programming paradigms



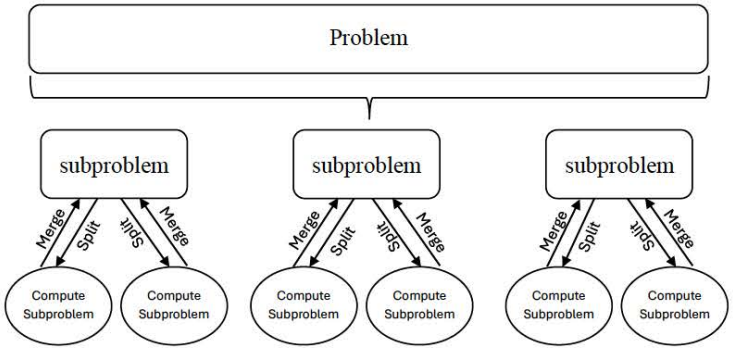
(a) Data Parallelism



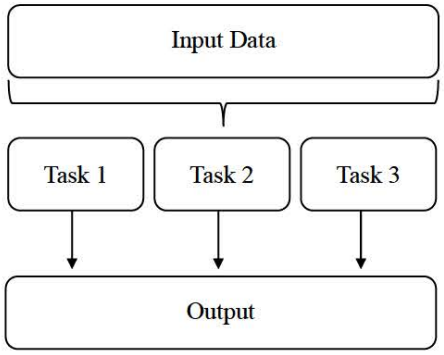
(b) Fork/Join



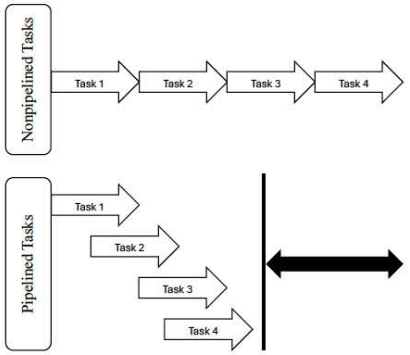
(c) Map/Reduce



(d) Divide and Conquer



(e) Task Parallelism



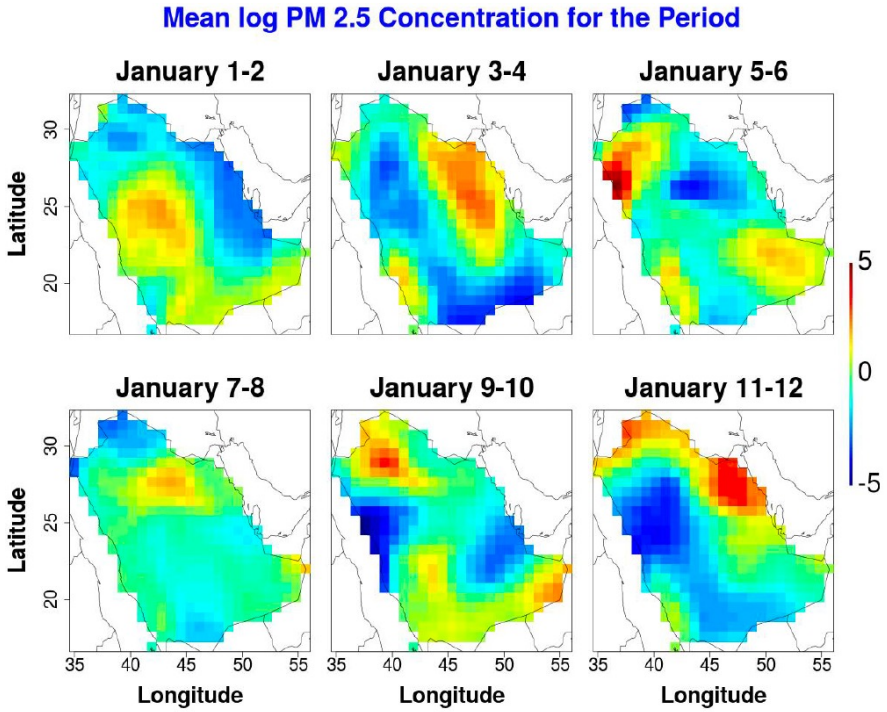
(f) Pipeline

c/o S. Abdulah (KAUST) *et al.*, Computational Statistics (2025)

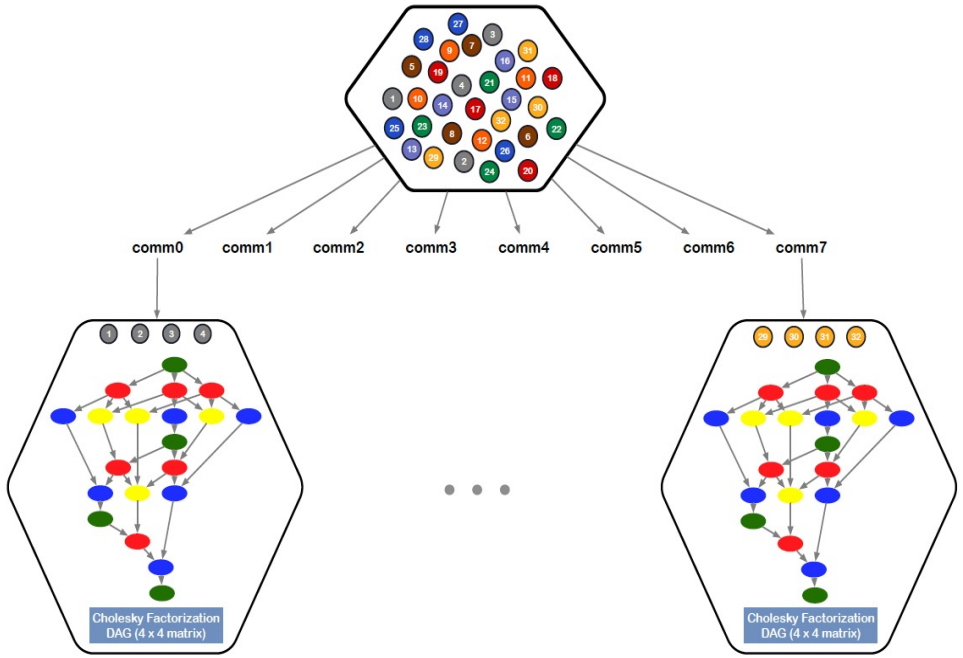
Wrap swarm around DAG-based tile alg

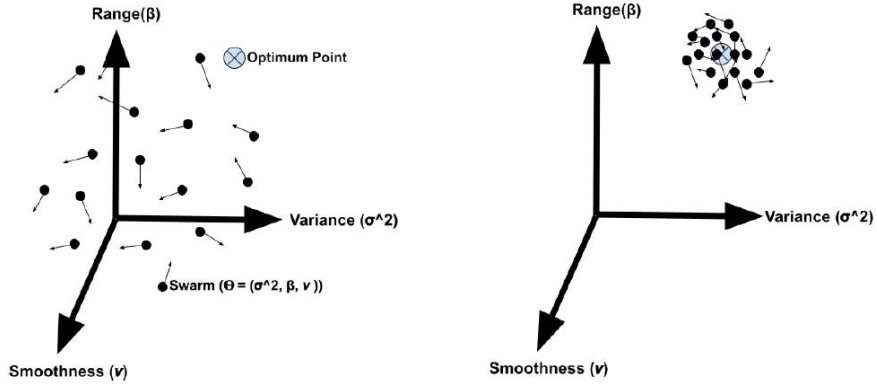
Maximum likelihood estimation:
maximize scalar objective as function of parameters θ, with observation vector Z and covariance matrix Σ :

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{Z}^T\Sigma^{-1}(\boldsymbol{\theta})\mathbf{Z} - \frac{1}{2}\log|\Sigma(\boldsymbol{\theta})|$$



c/o M. Salvaña (KAUST) *et al.*, PASC'22





(a) Iteration 0 (b) Iteration N

HPC vibe coding prompt #10

```
Exercise the "right to reorder"
```

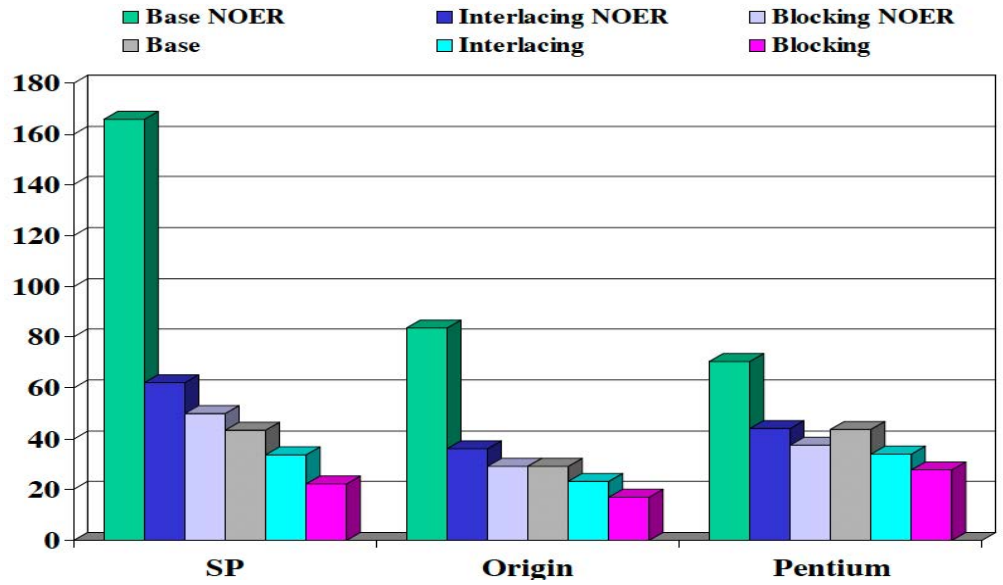
Why – spatial and temporal cache locality

How – *e.g.*, coloring, blocking, space-filling curves

Exploit the "right to re-order"

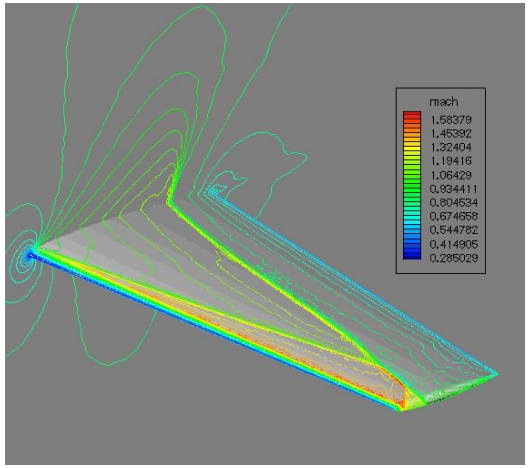
- Reorder mesh for temporal locality
- Interlace degrees of freedom at a point for spatial locality
- Block for loop unrolling

Saves 5x on node for unstructured CFD



Execution times for Euler flow over M6 wing for a fixed-size grid of 22,677 vertices (90,708 DOFs incompressible; 113,385 DOFs compressible)[a]

Enhancements			Results			
			Incompressible		Compressible	
Field interlacing	Structural blocking	Edge reordering	Time/step (s)	Ratio	Time/step (s)	Ratio
			83.6	–	140.0	–
×			36.1	2.31	57.5	2.44
×	×		29.0	2.88	43.1	3.25
		×	29.2	2.86	59.1	2.37
×		×	23.4	3.57	35.7	3.92
×	×	×	16.9	4.96	24.5	5.71



c/o K. Anderson (NASA) et al., SC'99 (Gordon Bell Prize)

HPC vibe coding prompt #11

```
Employ dynamic schedule and balancing
```

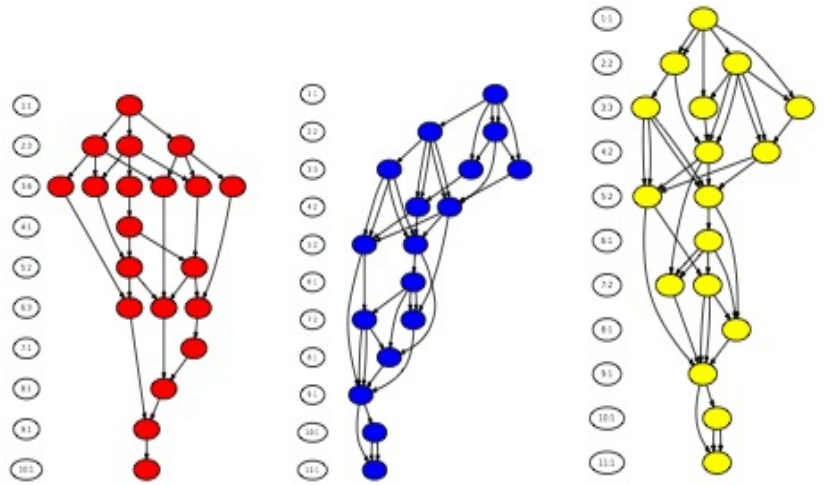
Why – increase concurrency opportunities; bulk synchronous processing leads to idleness while adaptive algorithms increase nonuniformity
How – dynamic runtime systems

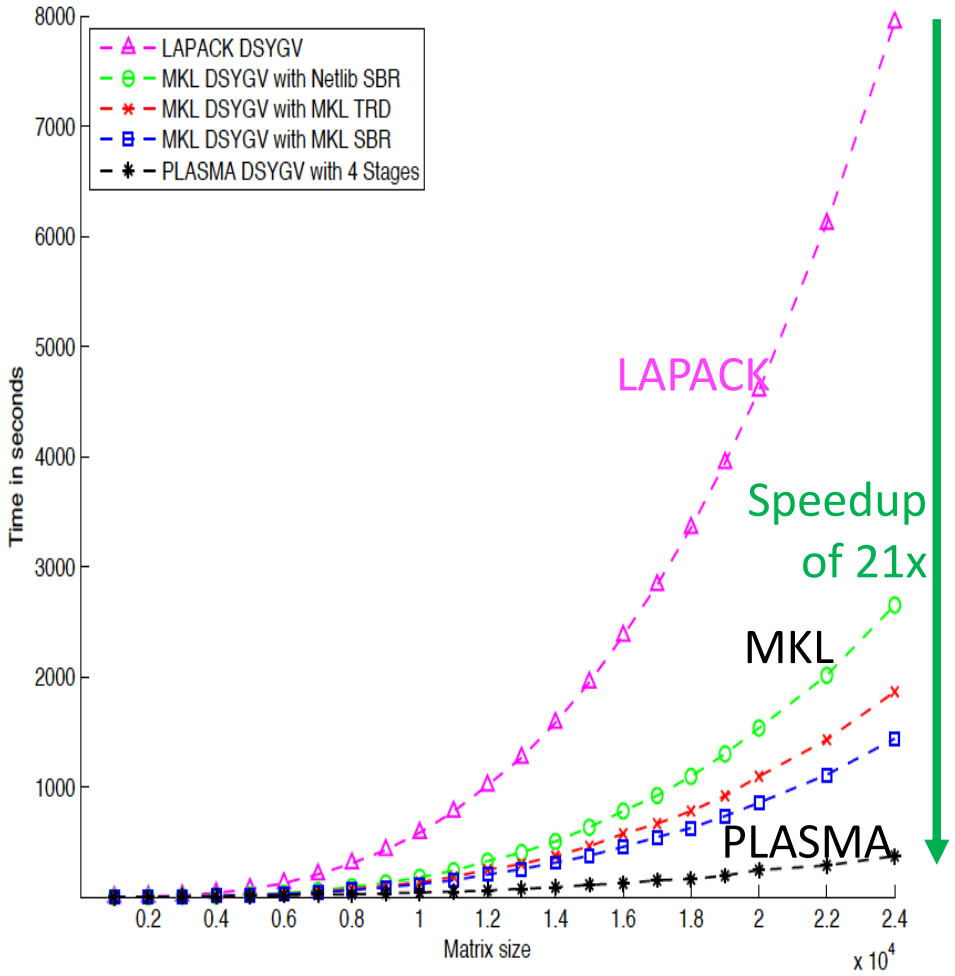
[Algorithms]

Employ dynamic scheduling

$$Ax = \lambda Bx$$

Operation	Explanation	LAPACK routine name
1 $B = L \times L^T$	Cholesky factorization	POTRF
2 $C = L^{-1} \times A \times L^{-T}$	application of triangular factors	SYGST or HEGST
3 $T = Q^T \times C \times Q$	tridiagonal reduction	SYEVD or HEEVD
4 $Tx = \lambda x$	QR iteration	STERF

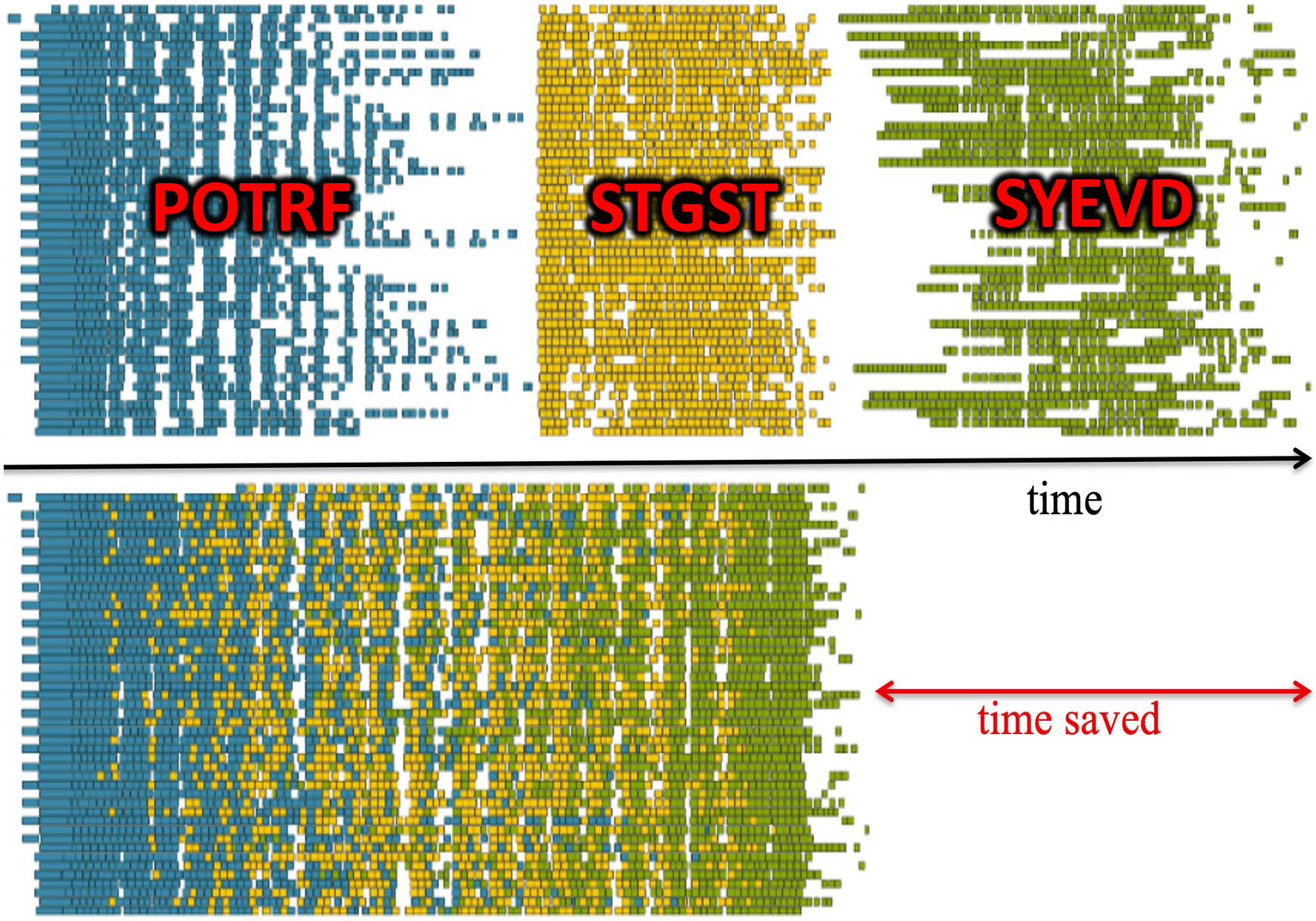




- Remove artifactual synchronizations in the form of subroutine boundaries
- Remove artifactual orderings in the form of pre-scheduled loops

c/o H. Ltaief (KAUST) *et al.*, Par. Comp. (2011)

Employ dynamic scheduling and balancing



HPC vibe coding prompt #12

```
Co-design algorithms with hardware
```

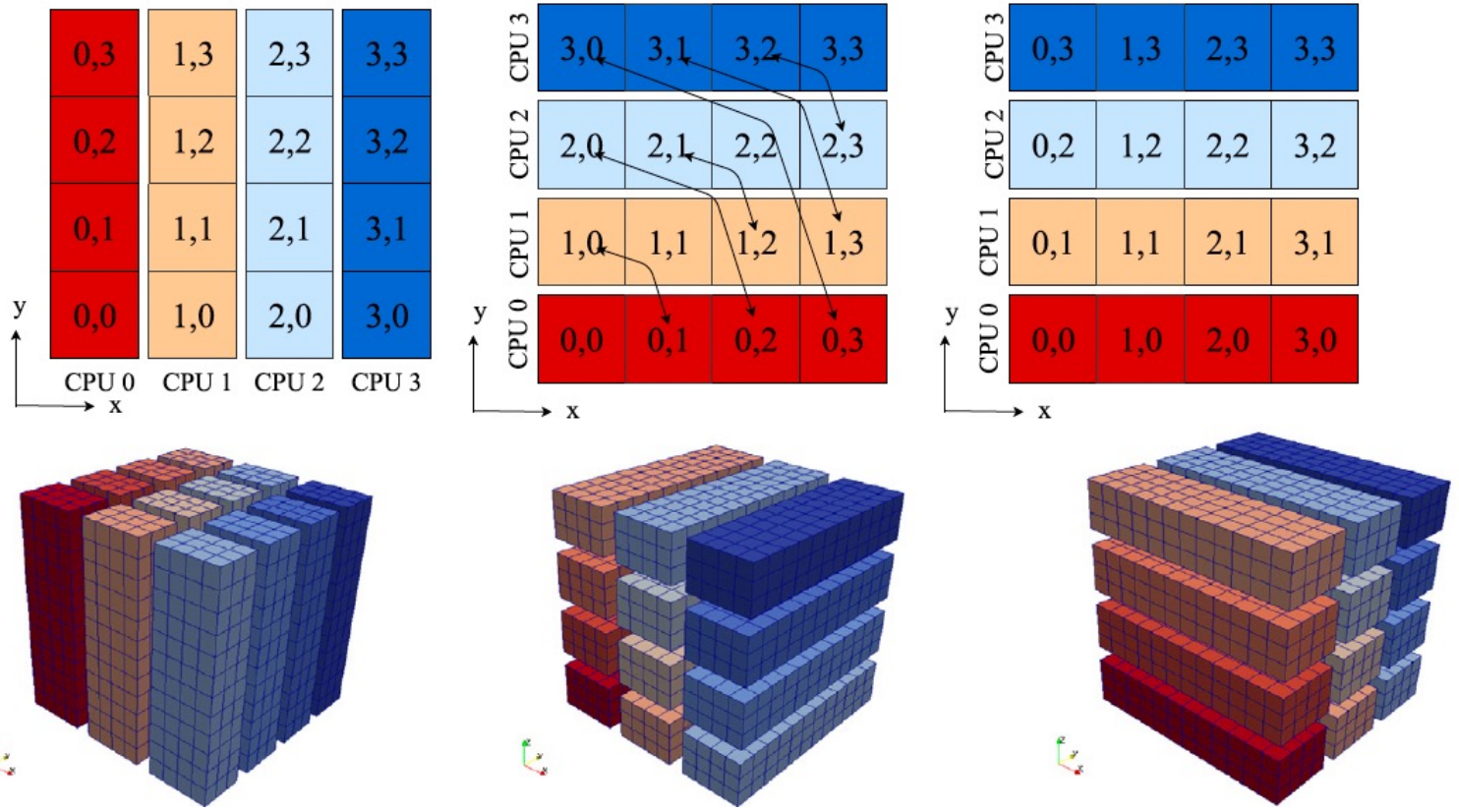
Why – potential performance is left on the table

How – *e.g.*, transpose in the network

[Algorithms]

Co-design hardware for algorithms (!)

- Multi-dimensional FFT (up to 6D in quantum chemistry apps)
- Use MPI_ALLTOALL_W to do the transpose in the network
- User code simplifies and performance in software is neutral today
- Once vendors fully implement *features already in MPI* with smarts *already in some networks*, it will be a large win



c/o L. Dalcin (KAUST) *et al.*, JPDC (2019)

HPC vibe coding prompt #13

```
Take resilience into algorithms
```

Why – fully reliable hardware is costly; fully reliable protocols that operate at the level of the OS (e.g., checkpointing) are time-consuming

How – *e.g.*, variance reduction to discriminate against faulty or malicious returns in federated learning

[Algorithms]

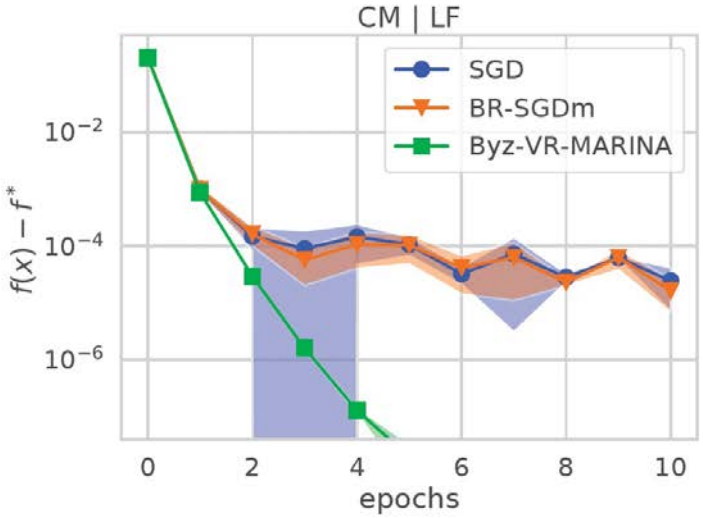
Robustness under “Byzantine” attacks

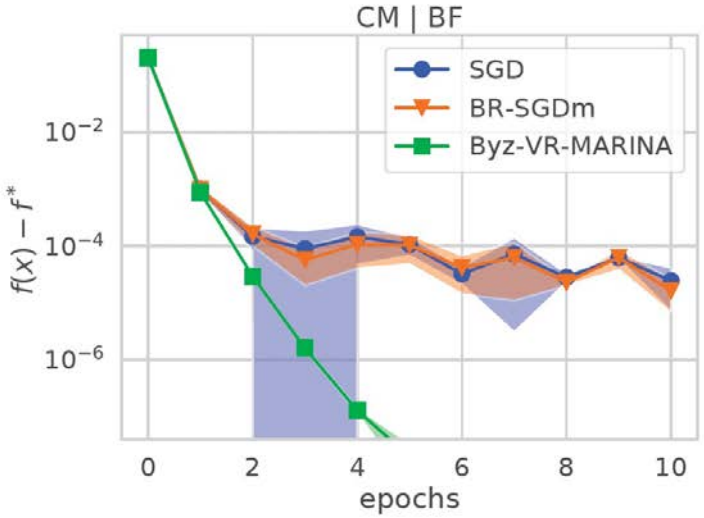
Label flipping attack

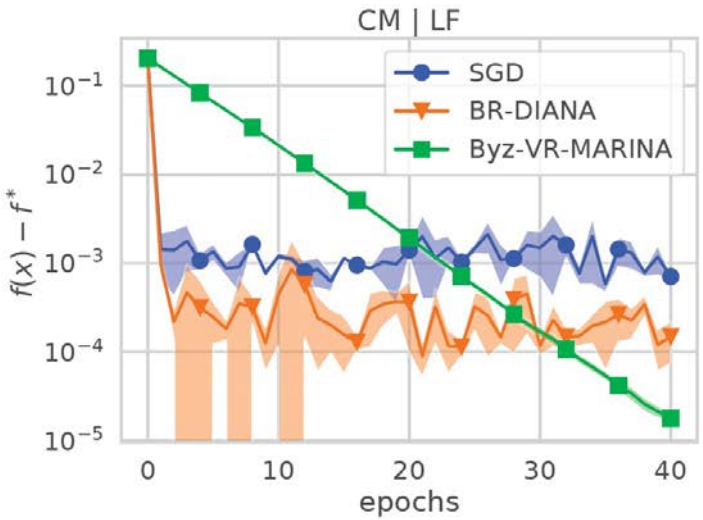
Bit flipping attack

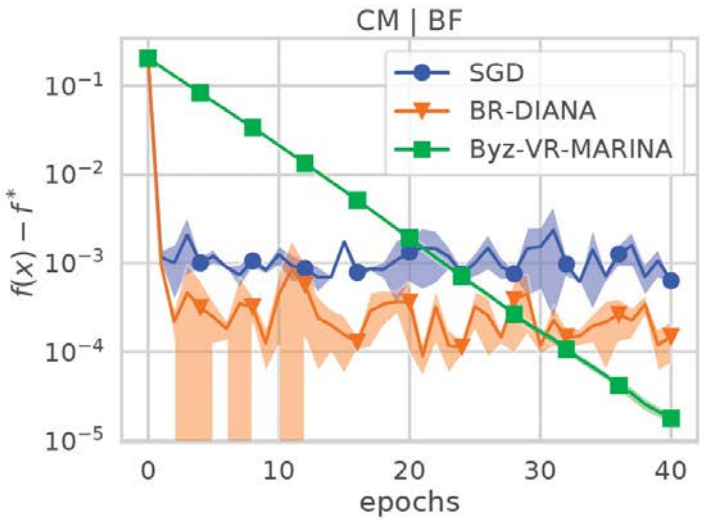
No compression

With compression









Three algorithms:

- **SGD** – stochastic gradient descent
- **BR-SGDm** – SGD with Polyak momentum
- **Byz-VR-MARINA** – with variance reduction

c/o E. Gorbunov (KAUST) *et al.*, ICLR’23

Byz-VR-MARINA progresses past the attacks

HPC vibe coding prompt #14

```
Process on the fly
```

Why – avoid stores and reloads

How – *e.g.*, JIT fused loops

[Algorithms]

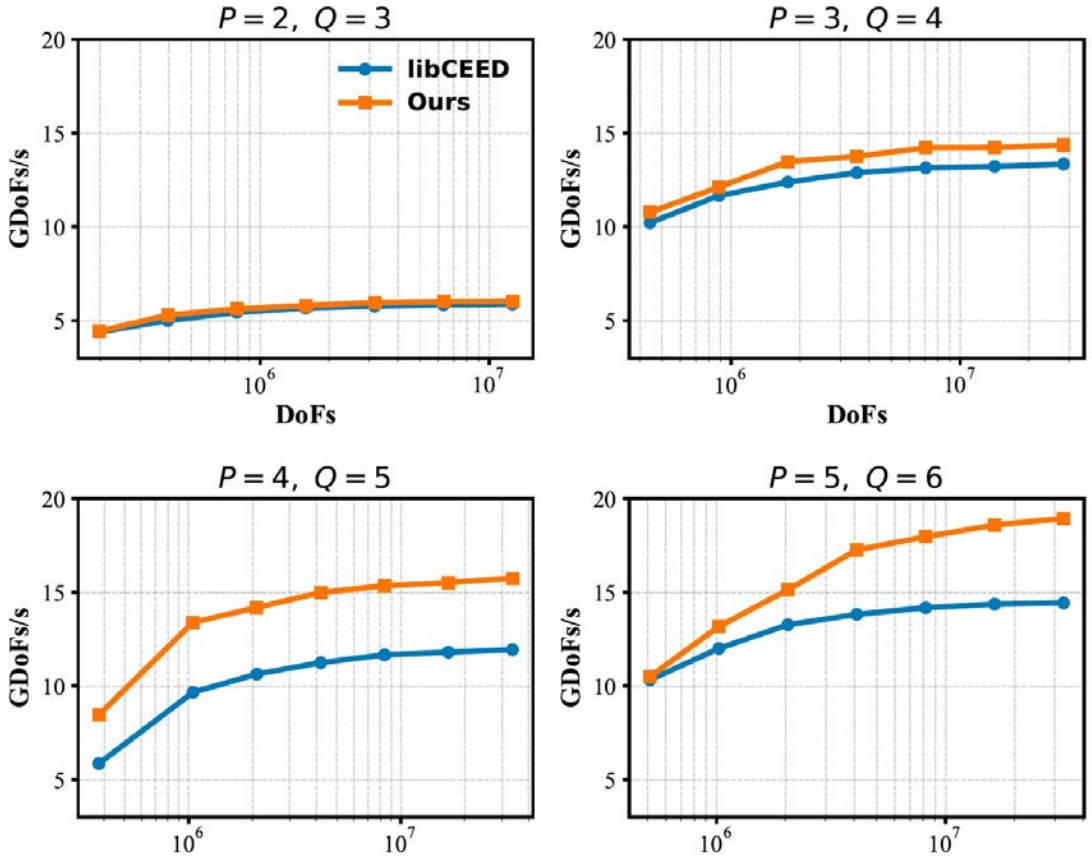
Matrix-free application of finite element pipelines in LibCEED and MFEM (LLNL)

$$\mathbf{y} = \mathbf{A}\mathbf{u} = R^T B^T Q(BR\mathbf{u})$$

- R : restriction to element from global DOFs
- B : mapping to template element quadrature points
- Q : pointwise operator evaluation at quadrature points

P (2 … 5) : polynomial order

Q (3 … 6) : quadrature order



c/o Ren (KAUST) *et al.,* ISC'26

HPC vibe coding prompt #15

```
Code to specialized backends
```

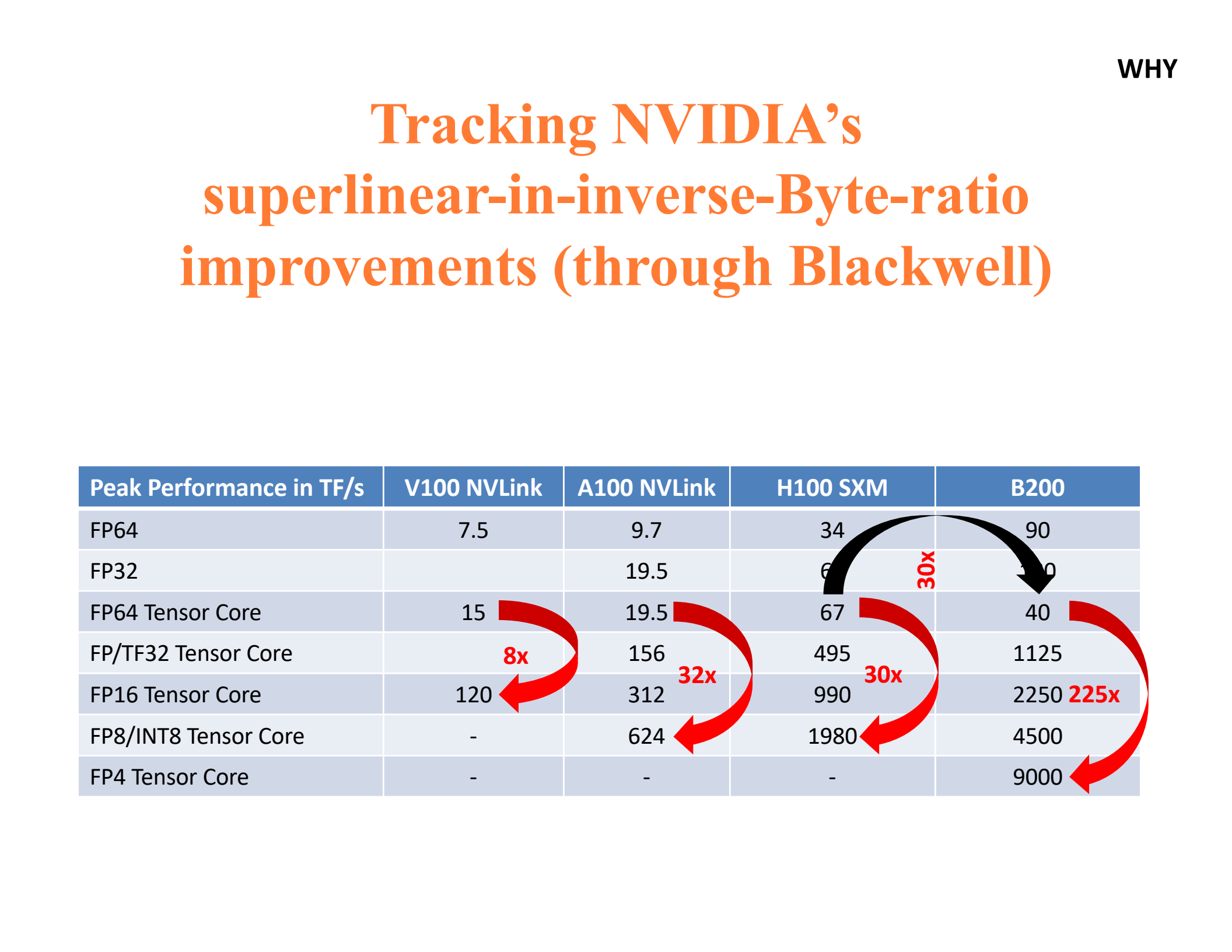
Why – derive maximum performance from the vendor

How – create uniform API for the user

[User]

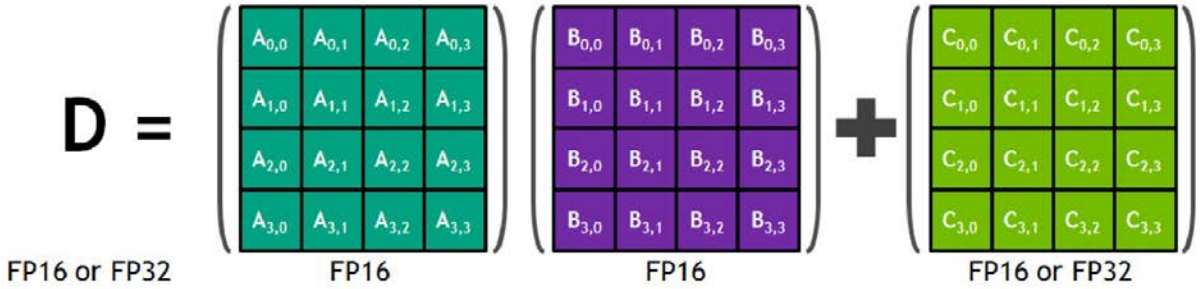
Tracking NVIDIA's superlinear-in-inverse-Byte-ratio improvements (through Blackwell)

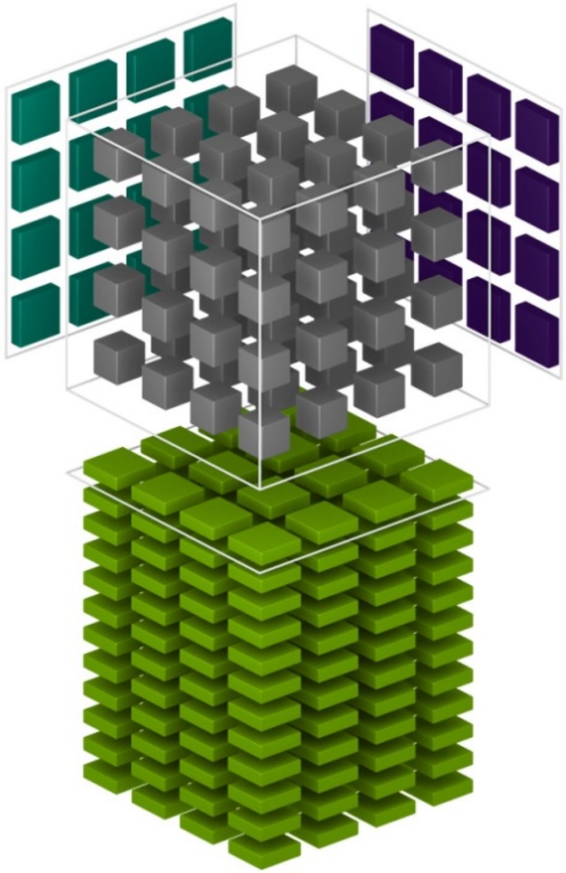
Peak Performance in TF/s	V100 NVLink	A100 NVLink	H100 SXM	B200
FP64	7.5	9.7	34	90
FP32		19.5	6	0
FP64 Tensor Core	15	19.5	67	40
FP/TF32 Tensor Core		156	495	1125
FP16 Tensor Core	120	312	990	2250
FP8/INT8 Tensor Core	-	624	1980	4500
FP4 Tensor Core	-	-	-	9000



Rely on SIMD/SIMT tasks

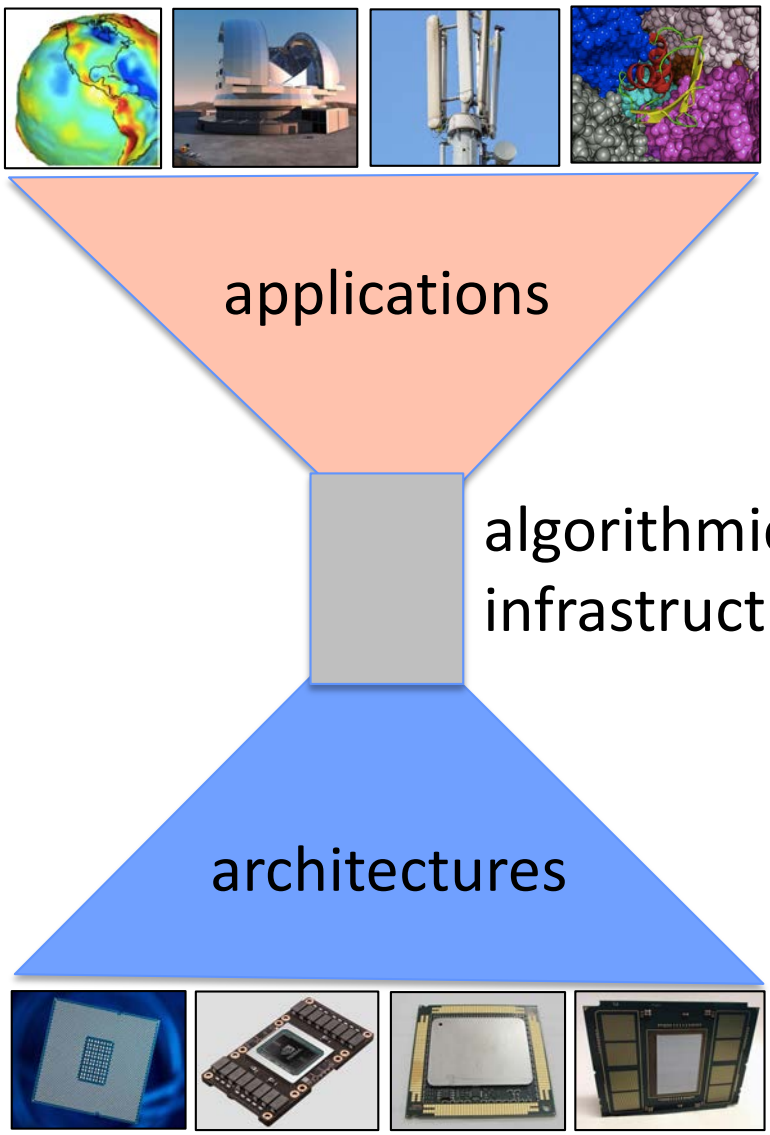
- Specialized operations are now hard-wired, e.g.,
  - traditional vector triadic operations
  - matrix-matrix operations used in DL





- Such instructions cannot be ignored
  - 4x4 matrix-matrix multiply-add does 64 FMADD instructions in one clock cycle
  - varieties of scales and precisions abound
  - more than an order of magnitude efficiency at stake

Employ APIs to specialized back-ends



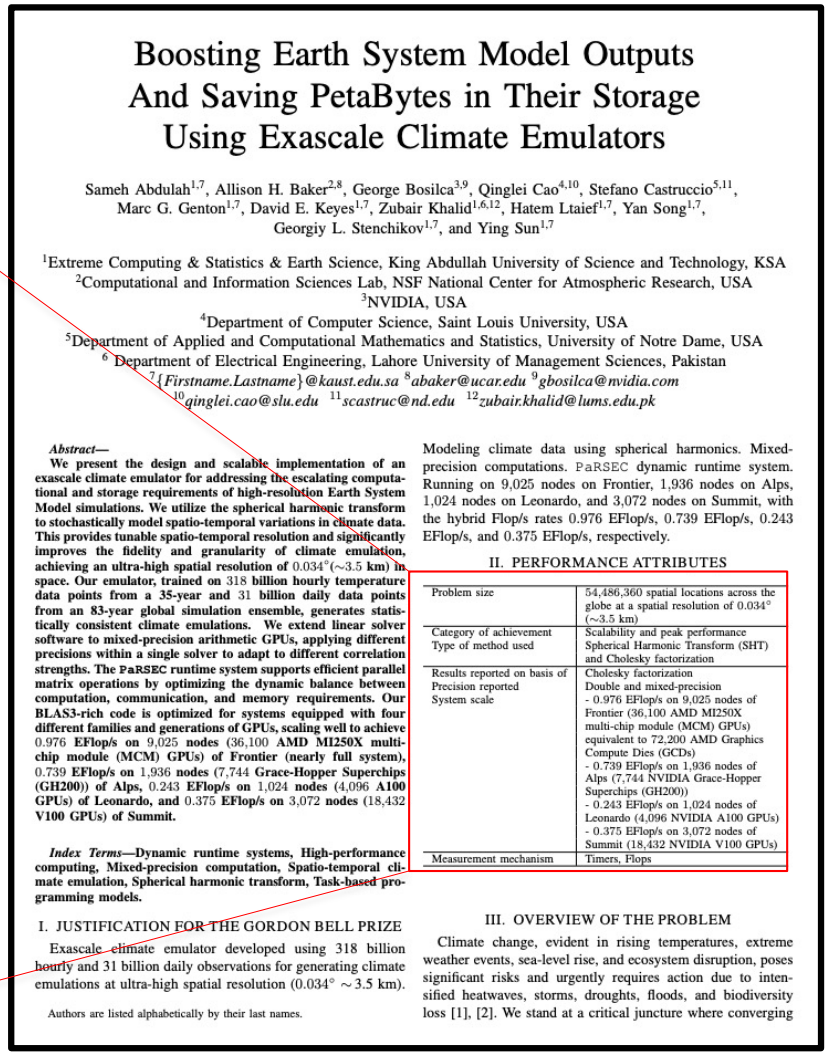
- Tiling and recursive subdivision create large numbers of small problems that can be marshaled for batched operations on GPUs
  - amortize call overheads
  - tune block size to memory capacities and warp thread counts
- Non-temporal stores, coalesced memory accesses, double-buffering, etc. reduce sensitivity to memory
- Code is complex
- Code is architecture-specific at the bottom
- Need to hide the support from the apps through an API

HOW

2024 Gordon Bell Climate Prize

Mixed precision, like all afternoon

Problem size	54,486,360 spatial locations across the globe at a spatial resolution of 0.034° (~3.5 km)
Category of achievement Type of method used	Scalability and peak performance Spherical Harmonic Transform (SHT) and Cholesky factorization
Results reported on basis of Precision reported System scale	Cholesky factorization Double and mixed-precision - 0.976 EFlop/s on 9,025 nodes of Frontier (36,100 AMD MI250X multi-chip module (MCM) GPUs) equivalent to 72,200 AMD Graphics Compute Dies (GCDs) - 0.739 EFlop/s on 1,936 nodes of Alps (7,744 NVIDIA Grace-Hopper Superchips (GH200)) - 0.243 EFlop/s on 1,024 nodes of Leonardo (4,096 NVIDIA A100 GPUs) - 0.375 EFlop/s on 3,072 nodes of Summit (18,432 NVIDIA V100 GPUs)
Measurement mechanism	Timers, Flops



Boosting Earth System Model Outputs And Saving PetaBytes in Their Storage Using Exascale Climate Emulators

Sameh Abdulah[1,7], Allison H. Baker[2,8], George Bosilca[3,9], Qinglei Cao[4,10], Stefano Castruccio[5,11], Marc G. Genton[1,7], David E. Keyes[1,7], Zubair Khalid[1,6,12], Hatem Ltaief[1,7], Yan Song[1,7], Georgiy L. Stenchikov[1,7], and Ying Sun[1,7]

[1]Extreme Computing & Statistics & Earth Science, King Abdullah University of Science and Technology, KSA
[2]Computational and Information Sciences Lab, NSF National Center for Atmospheric Research, USA
[3]NVIDIA, USA
[4]Department of Computer Science, Saint Louis University, USA
[5]Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, USA
[6] Department of Electrical Engineering, Lahore University of Management Sciences, Pakistan
[7]{Firstname.Lastname}@kaust.edu.sa [8]abaker@ucar.edu [9]gbosilca@nvidia.com
[10]qinglei.cao@slu.edu [11]scastruc@nd.edu [12]zubair.khalid@lums.edu.pk

***Abstract*—We present the design and scalable implementation of an exascale climate emulator for addressing the escalating computational and storage requirements of high-resolution Earth System Model simulations. We utilize the spherical harmonic transform to stochastically model spatio-temporal variations in climate data. This provides tunable spatio-temporal resolution and significantly improves the fidelity and granularity of climate emulation, achieving an ultra-high spatial resolution of 0.034°(~3.5 km) in space. Our emulator, trained on 318 billion hourly temperature data points from a 35-year and 31 billion daily data points from an 83-year global simulation ensemble, generates statistically consistent climate emulations. We extend linear solver software to mixed-precision arithmetic GPUs, applying different precisions within a single solver to adapt to different correlation strengths. The PaRSEC runtime system supports efficient parallel matrix operations by optimizing the dynamic balance between computation, communication, and memory requirements. Our BLAS3-rich code is optimized for systems equipped with four different families and generations of GPUs, scaling well to achieve 0.976 EFlop/s on 9,025 nodes (36,100 AMD MI250X multi-chip module (MCM) GPUs) of Frontier (nearly full system), 0.739 EFlop/s on 1,936 nodes (7,744 Grace-Hopper Superchips (GH200)) of Alps, 0.243 EFlop/s on 1,024 nodes (4,096 A100 GPUs) of Leonardo, and 0.375 EFlop/s on 3,072 nodes (18,432 V100 GPUs) of Summit.**

***Index Terms*—Dynamic runtime systems, High-performance computing, Mixed-precision computation, Spatio-temporal climate emulation, Spherical harmonic transform, Task-based programming models.**

I. JUSTIFICATION FOR THE GORDON BELL PRIZE

Exascale climate emulator developed using 318 billion hourly and 31 billion daily observations for generating climate emulations at ultra-high spatial resolution (0.034° ~3.5 km).

Authors are listed alphabetically by their last names.

Modeling climate data using spherical harmonics. Mixed-precision computations. PaRSEC dynamic runtime system. Running on 9,025 nodes on Frontier, 1,936 nodes on Alps, 1,024 nodes on Leonardo, and 3,072 nodes on Summit, with the hybrid Flop/s rates 0.976 EFlop/s, 0.739 EFlop/s, 0.243 EFlop/s, and 0.375 EFlop/s, respectively.

II. PERFORMANCE ATTRIBUTES

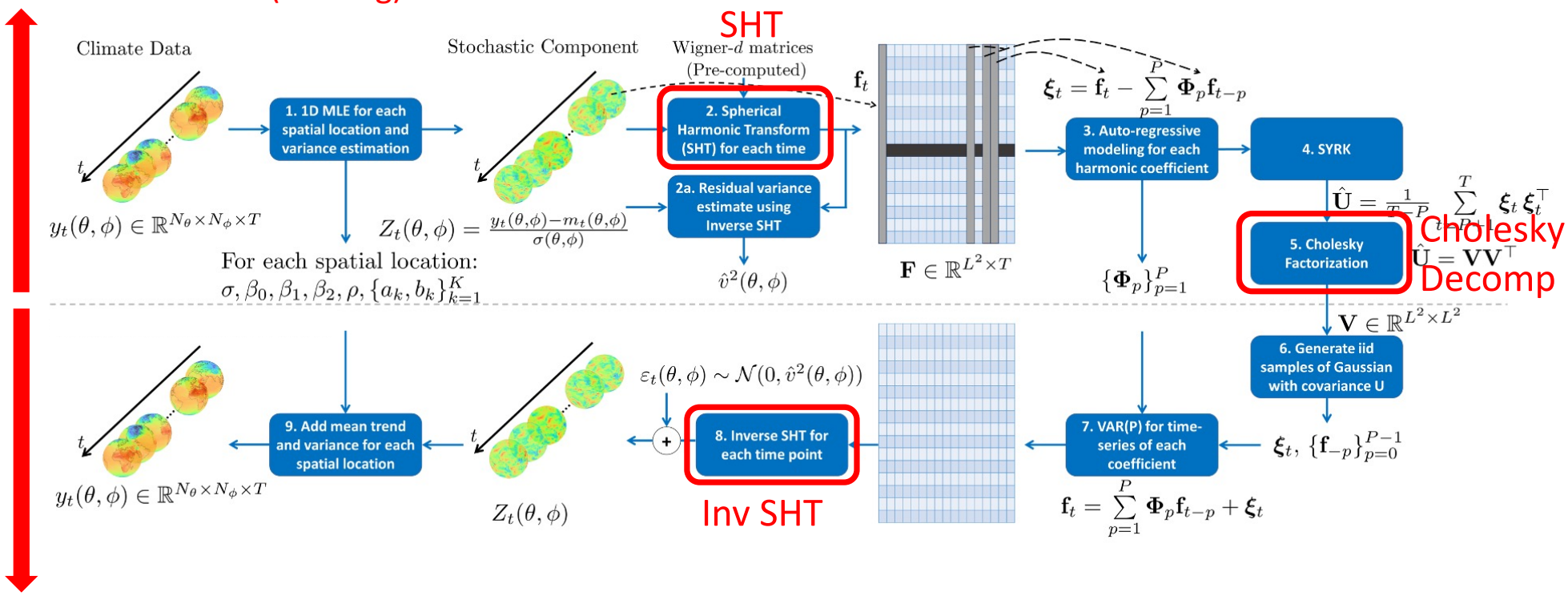
Problem size	54,486,360 spatial locations across the globe at a spatial resolution of 0.034° (~3.5 km)
Category of achievement Type of method used	Scalability and peak performance Spherical Harmonic Transform (SHT) and Cholesky factorization
Results reported on basis of Precision reported System scale	Cholesky factorization Double and mixed-precision - 0.976 EFlop/s on 9,025 nodes of Frontier (36,100 AMD MI250X multi-chip module (MCM) GPUs) equivalent to 72,200 AMD Graphics Compute Dies (GCDs) - 0.739 EFlop/s on 1,936 nodes of Alps (7,744 NVIDIA Grace-Hopper Superchips (GH200)) - 0.243 EFlop/s on 1,024 nodes of Leonardo (4,096 NVIDIA A100 GPUs) - 0.375 EFlop/s on 3,072 nodes of Summit (18,432 NVIDIA V100 GPUs)
Measurement mechanism	Timers, Flops

III. OVERVIEW OF THE PROBLEM

Climate change, evident in rising temperatures, extreme weather events, sea-level rise, and ecosystem disruption, poses significant risks and urgently requires action due to intensified heatwaves, storms, droughts, floods, and biodiversity loss [1], [2]. We stand at a critical juncture where converging

c/o S. Abdulah (KAUST) *et al.*, SC'24 (Gordon Bell Prize)

Major kernels: Spherical Harmonic Transform and Cholesky decomposition



c/o S. Abdulah (KAUST) *et al.*, SC'24 (Gordon Bell Prize)

HPC vibe coding prompt #16

```
Avoid over-resolving and over-solving
```

Why – defaulting to tight truncation errors, tight inner tolerances, and high precision is often (usually?) wasteful

How – *e.g.*, set inner resolutions and tolerances based on tolerances for outputs of interest, start loose, tighten upon convergence of outer loop, if necessary

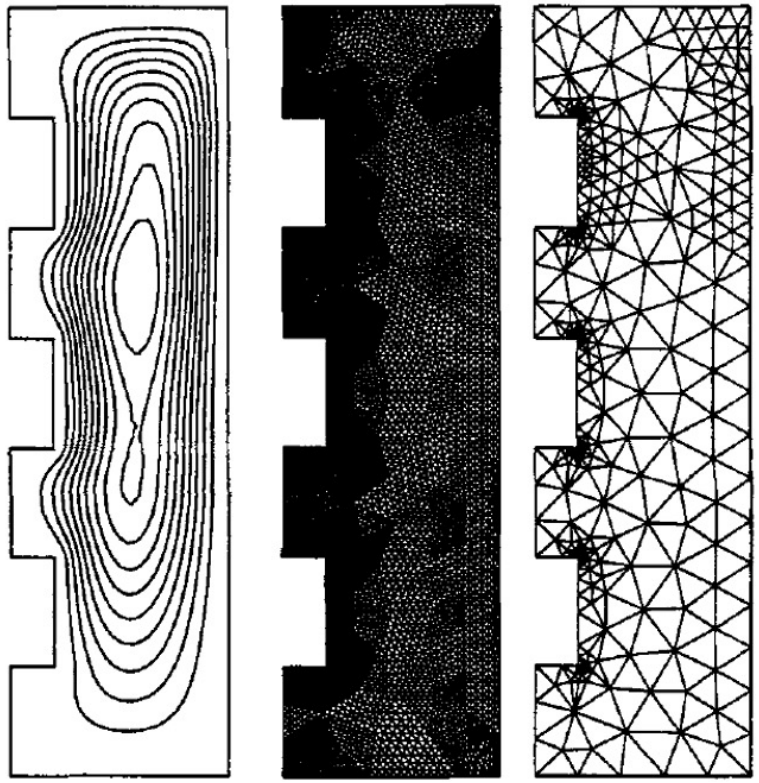
[User]

Avoid over-resolving with respect to output accuracy requirements

Sometimes, the output of interest from a computation is not a solution to high accuracy everywhere, but a *functional* of the solution to a *specified accuracy*, e.g.:

- compute the convective heat flux across a fluid-solid boundary, obtainable without globally uniform accuracy
- use low fidelity surrogates in early inner iterations of "outer loop problems"

c/o L. Machiels (MIT) *et al.*, JCP (2001)



Machiels, Peraire & Patera, *A posteriori FE Output Bounds for the Incompressible NS Equations*, (2001), J. Comp. Phys. **172**:401

HPC vibe coding prompt #17

```
Maximize "science per Joule"
```

Why – reducing the carbon footprint now rivals all other aspects of "high performance"

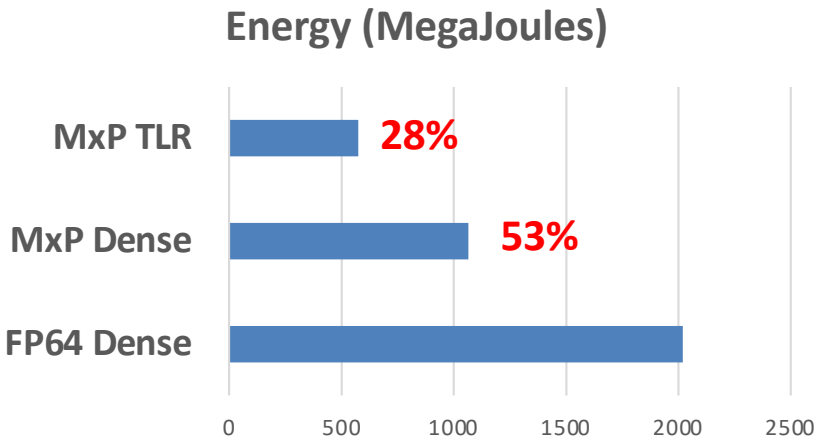
How – *e.g.*, combine mixed precision with low rank

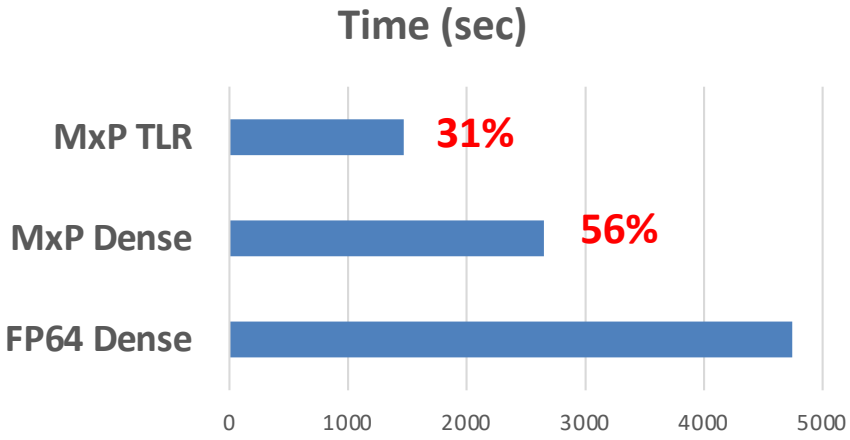
[User]

Energy and time savings go together

- Matérn 2D space kernel, matrix size 3.24M
- Solved to comparable accuracy by 3 algorithms
  - FP64 dense
  - adaptive mixed precision dense
  - adaptive mixed precision tile low rank

On 512 nodes of Shaheen III





c/o R. Alomairy (KAUST) *et al.*, IEEE HPEC (2025)

HPC vibe coding prompt #18

```
Reformulate before computing
```

Why – sometimes the task is defined in the “wrong” space

How – *e.g.*, turn BLAS1 vector-vector Hamming distance computations into BLAS3 GEMMS

[User]

Hamming distance between discrete vectors

Hamming distance H btw. rows i and j of X: $$H(i,j) = \sum_{k=1}^{m} \mathbb{1}(X_{ik} \neq X_{jk})$$

Initialize H to zero, then loop over each value val in X, updating with a GEMM:

$$vals = [A \quad C \quad G \quad T] \qquad H \leftarrow H + B^c_{val} \cdot B^T_{val}$$

B_{val} is bitwise incidence matrix of val in X; B^c_{val} is the bitwise complement matrix

$$X = \begin{bmatrix} A & T & G & C & A & A \\ T & A & C & G & T & A \\ G & C & A & T & G & C \\ C & G & T & A & C & G \\ T & G & C & A & T & C \\ A & C & G & T & A & G \end{bmatrix} \quad B_A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad B^c_A = 1 - B_A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$H = H_A + H_C + H_G + H_T \quad \begin{bmatrix} 0 & 5 & 6 & 6 & 6 & 3 \\ & 0 & 6 & 6 & 3 & 6 \\ & & 0 & 6 & 5 & 4 \\ & & & 0 & 4 & 5 \\ & & & & 0 & 6 \\ & & & & & 0 \end{bmatrix}$$

sym

c/o R. Alomairy (KAUST) *et al.,* ISC (2025)

Hamming distance performance on CPU and GPU

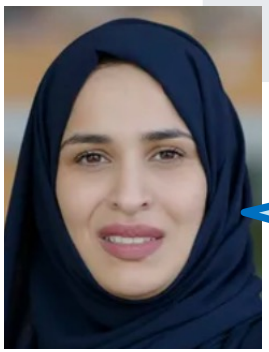
100K square matrix with 4 unique values (A, C, G, T)

	Intel Ice Lake dual 28-core 1008 GB			NVIDIA A100 SXM4 80 GB	
Algorithm	**Vector**	**Matrix**	**Matrix INT8**	**Matrix INT8**	**Matrix INT1**
Time (s)	40,878	241.23	143.94	16.70	8.84
Incremental Speedup	-	169x	1.67x	8.62x	1.89x
Overall Speedup	-	169x	284x	2,447x	4,624x

169x

8.62x

4,624x

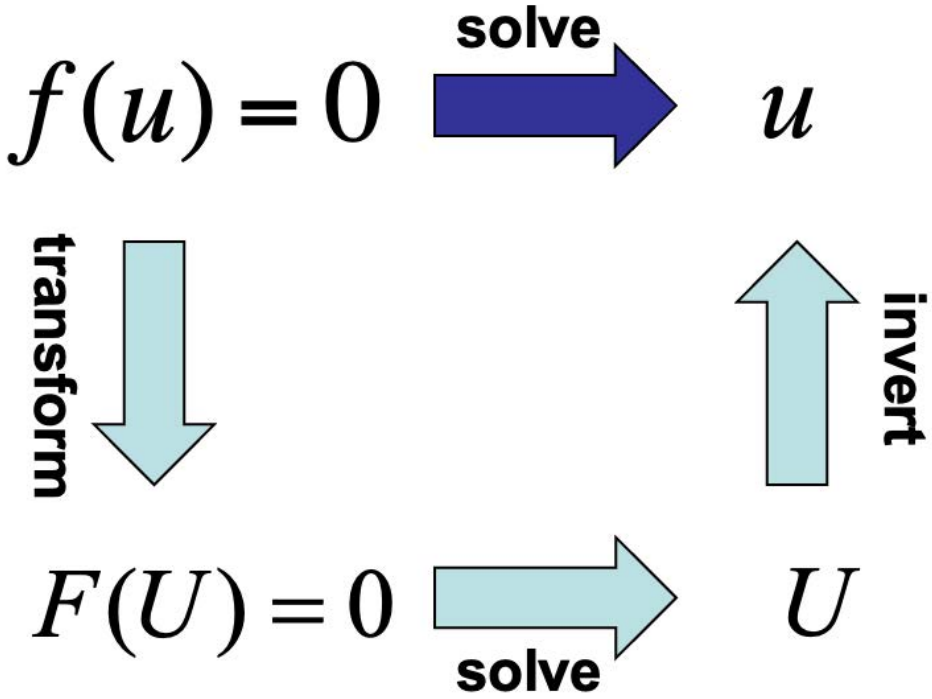


2025 IEEE TCHPC
Early Career Awardee

c/o R. Alomairy (KAUST) *et al.,* ISC (2025)

Reformulate applications before computing

- It is often difficult to solve a problem directly, as posed, compared with
  - transforming problem to new space
  - solving problem in new space
  - transforming back to original space
- Caveat: there is a sometimes a *conservation of difficulty* in transforming back



Generalize from a famous quotation

Previously:

"Think. Then discretize."

— Vladimir Rokhlin, Yale

Now:

"Think. Then prompt your vibe coder."

Extreme computing universals

Reside high on the memory hierarchy	Memory
Reduce message number and volume Reduce frequency of synchronization Reduce span of synchronization	Networks
Embrace heterogeneity	Processors
Exploit high-order discretizations Exploit hierarchical representations Exploit data sparsity Exploit nested levels of parallelism Exercise the "right to reorder" Employ dynamic schedule and balancing Co-design algorithms with hardware Take resilience into algorithms Process on the fly	Algorithms
Code to specialized backends Avoid over-resolving and over-solving Maximize "science per Joule" Reformulate before computing	Users

Examples abound throughout ML, as well

Memory			
1	Reside high on the memory hierarchy		Design of a low-memory-footprint variant of the "Adam" algorithm (the method that was used to train deep learning models for over a decade) which works very well in practice.
Network			
2	Reduce frequency of synchronization	I have several tens of papers on this topic; for example, the first paper which combines this with trick 4 (reduce message volume) and achieves a combined theoretical effect (i.e., double communiation acceleration via both trick 2 and trick 4). I did not cite this paper here; but can add it if you want.	The first paper showing that "local training" (do extra computation, such as SGD steps, on each node to reduce the frequency of synchronization) provably leads to fewer communication rounds (from cond number to square root of cond number) - for strongly convex distributed training/optimization (without the need to assume data/function similarity across the nodes), for any accuracy target. This problem was open for a decade, with thousands of works trying to solve it. This method was subsequently used (in concert with other tricks) by "Together AI" to train a foundational model over geographically distributed data sources/nodes.
3	Reduce span of synchronization		Asynchronous stochastic Gradient Descent (Asynchronous SGD) is a cornerstone method for parallelizing learning in distributed machine learning. However, its performance suffers under arbitrarily heterogeneous computation times across workers, leading to suboptimal time complexity and inefficiency as the number of workers scales. While several Asynchronous SGD variants have been proposed, recent findings by Tyurin & Richtárik (NeurIPS 2023) reveal that none achieve optimal time complexity, leaving a significant gap in the literature. In this paper, we propose Ringmaster ASGD, a novel Asynchronous SGD method designed to address these limitations and tame the inherent challenges of Asynchronous SGD. We establish, through rigorous theoretical analysis, that Ringmaster ASGD achieves optimal time complexity under arbitrarily heterogeneous and dynamically fluctuating worker computation times. This makes it the first Asynchronous SGD method to meet the theoretical lower bounds for time complexity in such scenarios.
4	Reduce message volume	Communication compression in distributed optimization	The first paper which shows that one can achieve provable communication acceleration (in distributed optimization) through gradient compression (i.e., communication compression). The key is the invention of the "DIANA" algorithm/trick - the first variance reduction technique for the reduction of the error/variance introduced by (stochastic unbiased) communication compression. This paper won the 2023 Charles Broyden Prize. The DIANA method was originally developed by me and my students in another paper; the award winning paper I am citing in here is a follow-up work which cleans up the theory in several ways, most notably by generalizing DIANA and the theory to arbitrary unbiased compressors.
Processor			
5	Embrace heterogeneity	Despite the long history of SGD (70+ years); it turns out we did not know what is the optimal parallel SGD method in the presence of nodes with heterogeneous compute nodes (in terms of their computation power). Rennala SGD (citation #1) is the first theoretically optimal parallel SGD method for finding a stationary point of the expectation of smooth nonconvex functions (in the data homogeneous case), and Malenia SGD is the first optimal SGD methods in the data heterogeneous case (also reference 1). Reference 2 is the first optimal SGD in the presence of both	Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimization methods are no exception: design of efficient parallel optimization methods and tight analysis of their theoretical properties are important research endeavors. While the minimax complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a new protocol that generalizes the classical oracle framework approach. Using this protocol, we establish minimax complexities for parallel optimization methods that have access to an unbiased stochastic gradient oracle with bounded variance. We consider a fixed computation model characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradient. We prove lower bounds and develop optimal algorithms that attain them. Our results have surprising consequences for the literature of asynchronous optimization methods.
Algorithm			
6	Exploit high order	The first successful combination of Newton method and communication compression in the sense that we can prove that compressing Hessians (and fixing the compression error via a novel error feedback method) leads to provable benefits in terms of communication complexity compared to vanilla distributed Newton. The convergence theory has rates which do not depend on the condition number (just like Newton's rate does not depend on the cond number) - despite the use of lossy communication compression! We have written 5-10 papers on this topic; I am	Inspired by recent work of Islamov et al (2021), we propose a family of Federated Newton Learn (FedNL) methods, which we believe is a marked step in the direction of making second-order methods applicable to FL. In contrast to the aforementioned work, FedNL employs a different Hessian learning technique which i) enhances privacy as it does not rely on the training data to be revealed to the coordinating server, ii) makes it applicable beyond generalized linear models, and iii) provably works with general contractive compression operators for compressing the local Hessians, such as Top-K or Rank-R, which are vastly superior in practice. Notably, we do not need to rely on error feedback for our methods to work with contractive compressors. Moreover, we develop FedNL-PP, FedNL-CR and FedNL-LS, which are variants of FedNL that support partial participation, and globalization via cubic regularization and line search, respectively, and FedNL-BC, which is a variant that can further benefit from bidirectional compression of gradients and models, i.e., smart uplink gradient and smart downlink model compression. We prove local convergence rates
7	Exploit hierarchical representations	FFT,MG,FMM,SG,MLMC,H2, etc.: to shave off powers of N in complexity	
8	Exploit data sparsity	meet the curse of dimensionality with the blessing of adaptive, accuracy-preserving low rank and low precision	In Federated Learning (FL), both client resource constraints and communication costs pose major problems for training large models. In the centralized setting, sparse training addresses resource constraints, while in the distributed setting, local training addresses communication costs. Recent work has shown that local training provably improves communication complexity through acceleration. In this work we show that in FL, naive integration of sparse training and acceleration fails, and we provide theoretical and empirical explanations of this phenomenon. We introduce Sparse-ProxSkip, addressing the issue and implementing the efficient technique of Straight-Through Estimator pruning into sparse training. We demonstrate the performance of Sparse-ProxSkip in extensive experiments.
9	Exploit nested levels of parallelism	First paper showing that parallelizing across 2 levels of a tree structure in federated learning leads to provable communication complexity advantages.	We study distributed optimization methods based on the {\em local training (LT)} paradigm: achieving communication efficiency by performing richer local gradient-based training on the clients before parameter averaging. Looking back at the progress of the field, we {\em identify 5 generations of LT methods: 1) heuristic, 2) homogeneous, 3) sublinear, 4) linear, and 5) accelerated. The 5th generation, initiated by the ProxSkip method of Mishchenko, Malinovsky, Stich and Richtárik (2022) and its analysis, is characterized by the first theoretical confirmation that LT is a communication acceleration mechanism. Inspired by this recent progress, we contribute to the 5th generation of LT methods by showing that it is possible to enhance them further using {\em variance reduction}. While all previous theoretical results for LT methods ignore the cost of local work altogether, and are framed purely in terms of the number of communication rounds, we show that our methods can be substantially faster in terms of the {\em total training cost} than the state-of-the-art method ProxSkip in theory and practice in the regime when local computation is
10	Exercise the "right to reorder"	arithmetic-neutral blockings and fusings to spatial or temporal locality	
11	Employ dynamic schedule and balancing	A dynamic/adaptive communication compression strategy which leads to theoretical and practical benefits over previous approaches.	We propose a family of adaptive integer compression operators for distributed Stochastic Gradient Descent (SGD) that do not communicate a single float. This is achieved by multiplying floating-point vectors with a number known to every device and then rounding to integers. In contrast to the prior work on integer compression for SwitchML by Sapio et al. (2021), our IntSGD method is provably convergent and computationally cheaper as it estimates the scaling of vectors adaptively. Our theory shows that the iteration complexity of IntSGD matches that of SGD up to constant factors for both convex and non-convex, smooth and non-smooth functions, with and without overparameterization. Moreover, our algorithm can also be tailored for the popular all-reduce primitive and shows promising empirical performance.
13	Take resilience into algorithms	Here we prove that Byzantine robustness (a type of resilience against attacks) can be achieved with faster convergence rates than before, via the insight that variance reduction can be used to improve these algorithms in theory and practice. As a bonus, we enhance thes emethods with communication compression for enhanced communication complexity. This is paper #1. In paper #2, we are the first to show that Byz robustness can be achieved in the partial participation setting - a very difficult setting for Byzantine robustness.	Byzantine-robustness has been gaining a lot of attention due to the growth of the interest in collaborative and federated learning. However, many fruitful directions, such as the usage of variance reduction for achieving robustness and communication compression for reducing communication costs, remain weakly explored in the field. This work addresses this gap and proposes Byz-VR-MARINA - a new Byzantine-tolerant method with variance reduction and compression. A key message of our paper is that variance reduction is key to fighting Byzantine workers more effectively. At the same time, communication compression is a bonus that makes the process more communication efficient. We derive theoretical convergence guarantees for Byz-VR-MARINA outperforming previous state-of-the-art for general non-convex and Polyak-Łojasiewicz loss functions. Unlike the concurrent Byzantine-robust methods with variance reduction and/or compression, our complexity results are tight and do not rely on restrictive assumptions such as boundedness of the gradients or limited compression. Moreover, we provide the first analysis of a
14	Process on the fly		inefficiencies using more computation than required, especially when computation times vary across devices. If the computation times were known in advance, training could be fast and resource-efficient by assigning more tasks to faster workers. The challenge lies in achieving this optimal allocation without prior knowledge of the computation time distributions. In this paper, we propose ATA (Adaptive Task Allocation), a method that adapts to heterogeneous and random distributions of worker computation times. Through rigorous theoretical analysis, we show that ATA identifies the optimal task allocation and performs comparably to methods with prior knowledge of computation times. Experimental results further demonstrate that ATA is resource-efficient, significantly reducing costs compared to the greedy approach, which can be arbitrarily expensive depending on the number of workers.
User			
15	Code to specialized backends	hourglass model -- present one API for kernels of multiple applications and multiple architecture targets	
16	Avoid over-resolving and over-solving	We show that inexact computation of orthogonalization (e.g., via a few steps of Newton-Schulz) can be beneficial in theory when training a ML model via the Muon algorithm from Dec 2024 -- a new method replacing the Adam/AdamW optimizers for practical deep learning.	The Muon optimizer has rapidly emerged as a powerful, geometry-aware alternative to AdamW, demonstrating strong performance in large-scale training of neural networks. However, a critical theory-practice disconnect exists: Muon's efficiency relies on fast, approximate orthogonalization, yet all prior theoretical work analyzes an idealized, computationally intractable version assuming exact SVD-based updates. This work moves beyond the ideal by providing the first analysis of the inexact orthogonalized update at Muon's core. We develop our analysis within the general framework of Linear Minimization Oracle (LMO)-based optimization, introducing a realistic additive error model to capture the inexactness of practical approximation schemes. Our analysis yields explicit bounds that quantify performance degradation as a function of the LMO inexactness/error. We reveal a fundamental coupling between this inexactness and the optimal step size and momentum: lower oracle precision requires a smaller step size but larger momentum parameter. These findings elevate the approximation procedure (e.g., the number of Newton-Schulz
17	Consider science per Joule	ultimate metric of merit is not flop/s, parallel efficiency, or percentage of peak	
18	Reformulate before computing	The reformulation in paper #1 is clear from the title and abstract already. The second paper is a highly cited work which shows how one can think of minibatch SGD via a special/novel stochastic reformulation of the problem. This is useful for the design and analysis of minibatch SGD methods.	We develop a family of reformulations of an arbitrary consistent linear system into a stochastic problem. The reformulations are governed by two user-defined parameters: a positive definite matrix defining a norm, and an arbitrary discrete or continuous distribution over random matrices. Our reformulation has several equivalent interpretations, allowing for researchers from various communities to leverage their domain specific insights. In particular, our reformulation can be equivalently seen as a stochastic optimization problem, stochastic linear system, stochastic fixed point problem and a probabilistic intersection problem. We prove sufficient, and necessary and sufficient conditions for the reformulation to be exact. Further, we propose and analyze three stochastic algorithms for solving the reformulated problem -- basic, parallel and accelerated methods -- with global linear convergence rates. The rates can be interpreted as condition numbers of a matrix which depends on the system matrix and on the reformulation parameters. This gives rise to a new phenomenon which we call stochastic preconditioning, and which refers to

c/o P. Richtarik (KAUST)

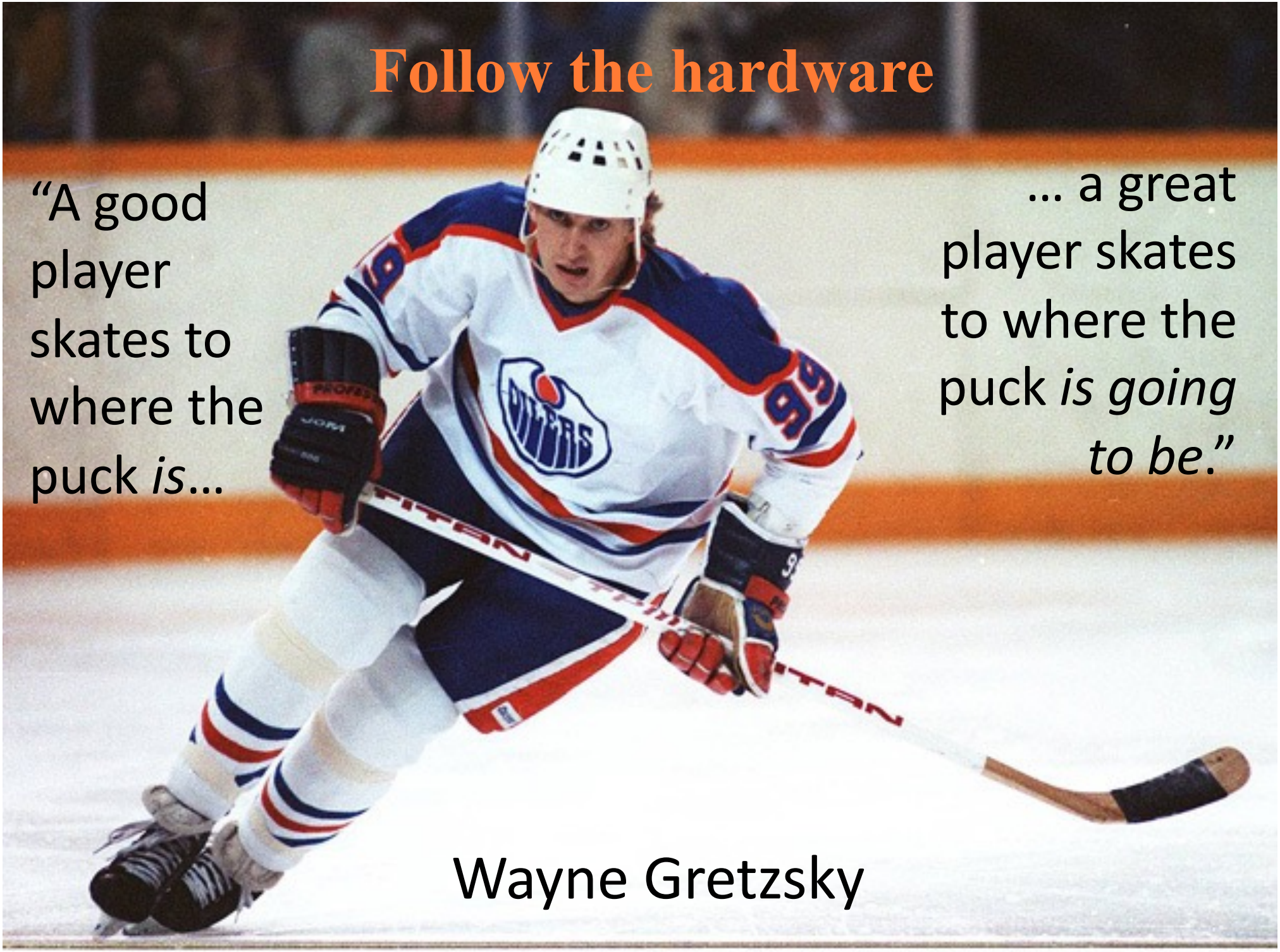
Request

You have a world of fresher, more interesting, more compelling examples out there, for each of the 18 universals.

Likely, there are also more “universals” ☺.

Please send me yours, to be included in the next version of this talk.

Follow the hardware

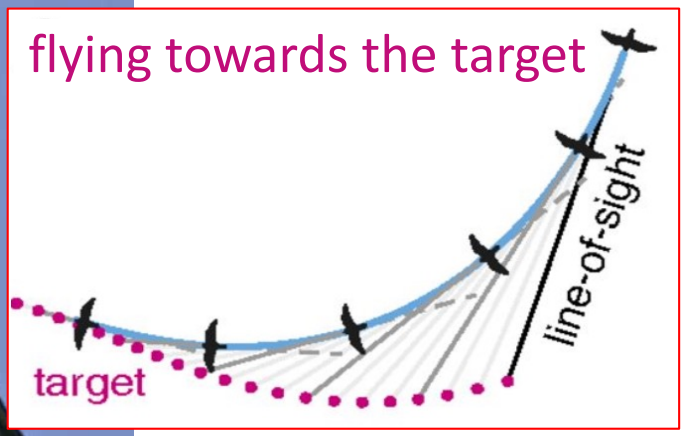


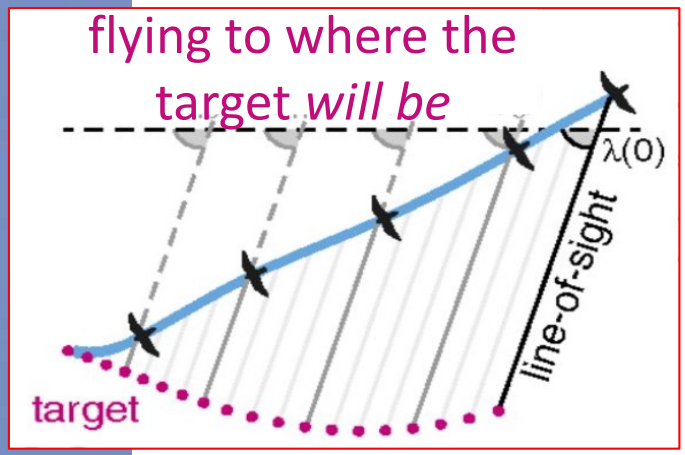
"A good player skates to where the puck *is*…

… a great player skates to where the puck *is going to be*."

Wayne Gretzsky

A falcon flies to where the prey *will be* …





… rather than where it is

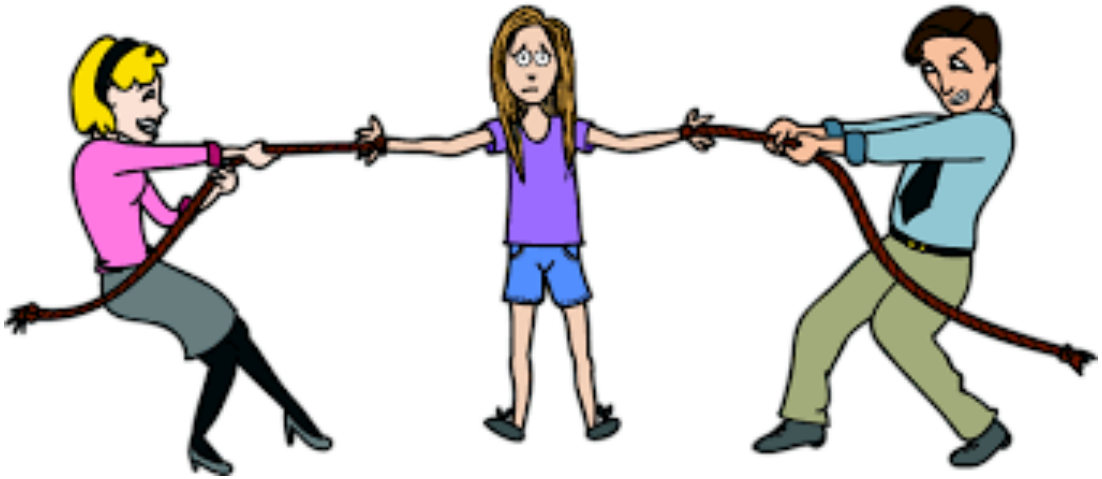
C. H. Brighton, et al. PNAS (2017)

Takeaway: full employment

Algorithms must span a widening gulf …

adaptive algorithms

ambitious applications
increasingly dynamic and unstructured

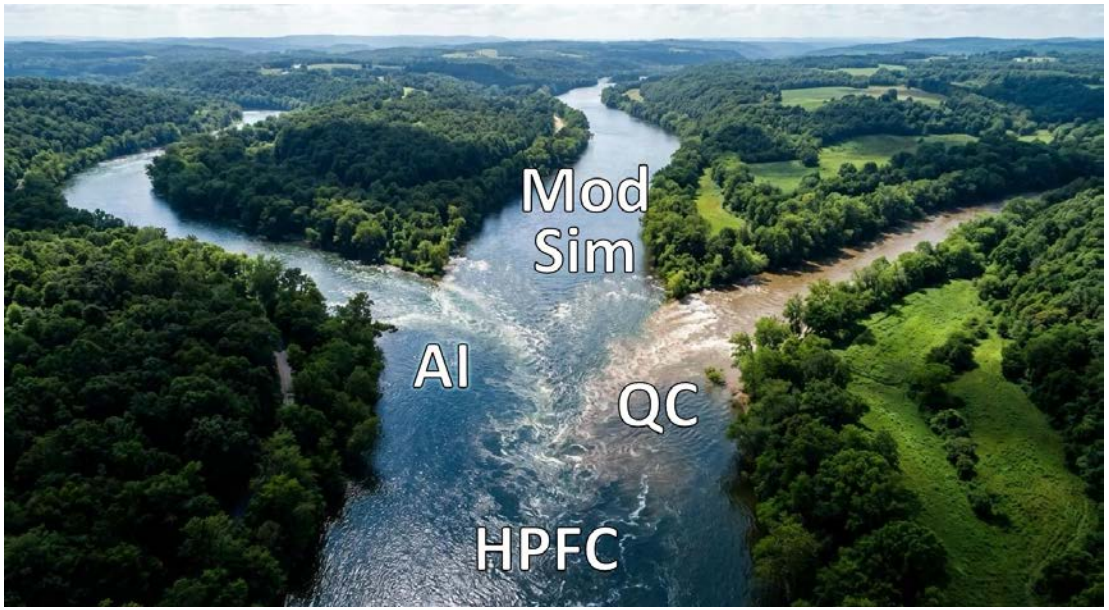


austere architectures
increasingly optimized for uniformity

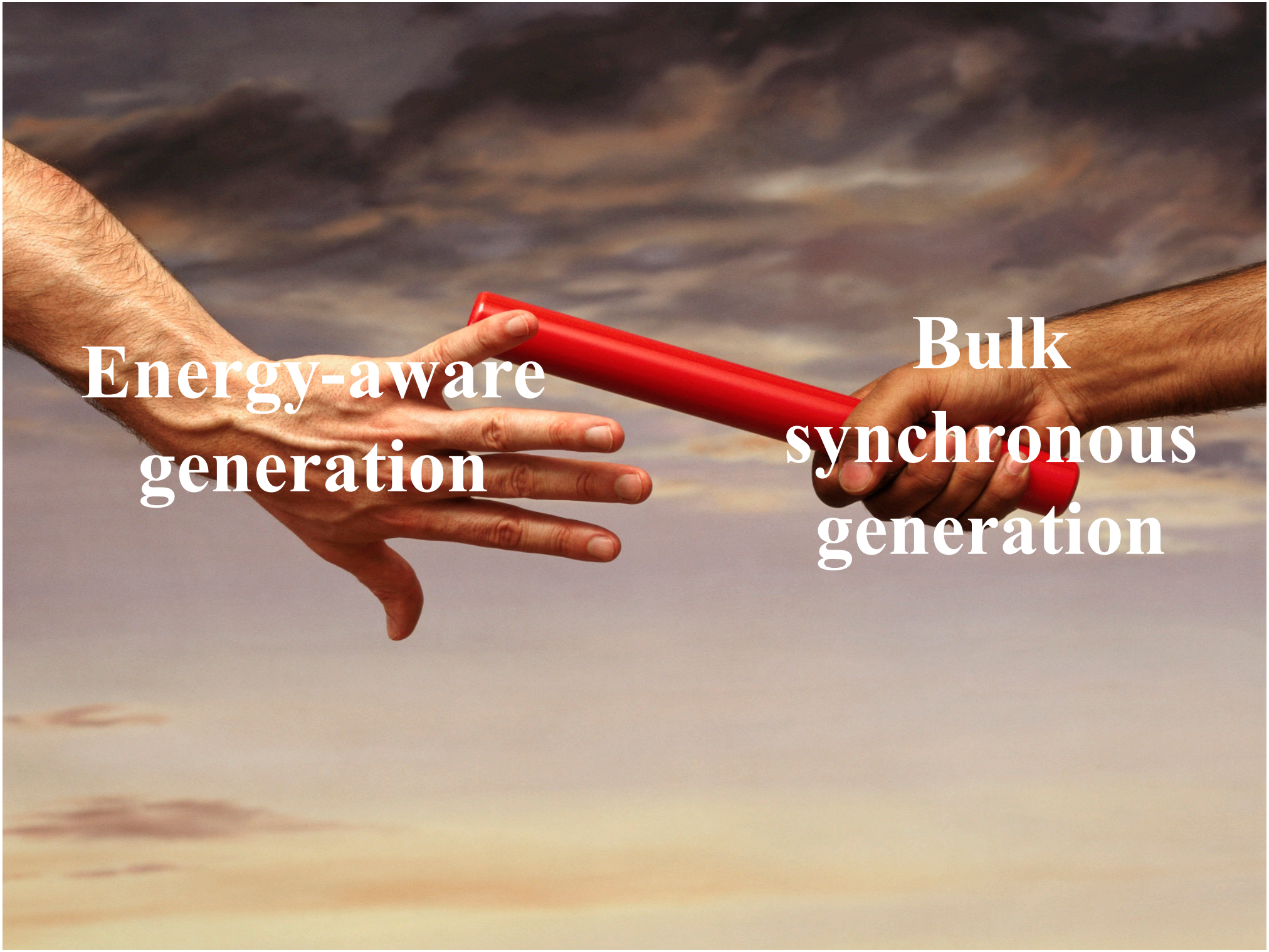
… a full employment program for algorithm developers!

Not *yet* easily vibe-coded ☹

But check back next year ☺



Reside high on the memory hierarchy	Memory
Reduce message number and volume Reduce frequency of synchronization Reduce span of synchronization	Network
Embrace heterogeneity	Processor
Exploit high-order discretizations Exploit hierarchical representations Exploit data sparsity Exploit nested levels of parallelism Exercise the "right to reorder" Employ dynamic schedule and balancing Co-design algorithms with hardware Take resilience into algorithms Process on the fly	Algorithm
Code to specialized backends Avoid over-resolving and over-solving Maximize "science per Joule" Reformulate before computing	User


Energy-aware generation
Bulk synchronous generation

Bibliography of “How” examples (1/4)

W. Boukaram, D. E. Keyes, S. Li, Y. Liu, G. Turkiyyah. **Linear Complexity H² Direct Solver for Fine-Grained Parallel Architectures.** *IMA Journal of Numerical Analysis*, special issue in memory of Nick Higham, 2026. [arXiv:2509.11152]

E. Solomonik, J. Demmel. **Communication-Optimal Parallel 2.5D Matrix Multiplication and LU Factorization Algorithms.** *Proc. Euro-Par 2011: 17th International European Conference on Parallel and Distributed Computing*, LNCS 6853, pp. 90–102, Springer, 2011. **Distinguished Paper Prize.** Source

P. Ghysels, W. Vanroose. **Hiding Global Synchronization Latency in the Preconditioned Conjugate Gradient Algorithm.** *Parallel Computing*, 40(7):224–238, 2014. doi:10.1016/j.parco.2013.06.001

L. Luo, X.-C. Cai, D. E. Keyes. **A Nonlinear Elimination Preconditioned Inexact Newton Algorithm.** *SIAM Journal on Scientific Computing*, 43(5):S317–S344, 2021. doi:10.1137/21M1416138

G. Greene-Diniz, D. Z. Manrique, W. Sennane, Y. Outeiral, D. Erglis, A. Bagusetty, D. Muñoz-Ramo, et al. **Modelling Carbon Capture on Metal-Organic Frameworks with Quantum Computing.** *EPJ Quantum Technology*, 9(1):37, 2022. doi:10.1140/epjqt/s40507-022-00155-w

Bibliography of “How” examples (2/4)

M. Hutchinson, A. Heinecke, H. Pabst, G. Henry, M. Parsani, D. E. Keyes, **Efficiency of High Order Spectral Element Methods on Petascale Architectures**. *Proceedings of the International Supercomputing Conference (ISC’16), J. M. Kunkel et al., eds., Lecture Notes in Computer Science, Springer, Vol. 9697, pp. 449–466, doi:10.1007/978-3-319-41321-123*, 2016.

D. Sushnikova, G. Turkiyyah, E. Chow, D. Keyes, **H2-MG: A multigrid method for hierarchical rank structured matrices.** *SIAM Journal on Scientific Computing, 48:A286-A308, doi:10.1137/24M171944X,* 2026.

H. Ltaief, Y. Hong, L. Wilson, M. Jacquelin, M. Ravasi, D. E. Keyes. **Scaling the 'Memory Wall' for Multi-Dimensional Seismic Processing with Algebraic Compression on Cerebras CS-2 Systems.** *Proceedings of SC'23: International Conference for High Performance Computing, Networking, Storage, and Analysis*, Article 6, pp. 6:1–6:12, ACM/IEEE, 2023. **ACM Gordon Bell Prize Finalist.** Source

S. Abdulah, M. L. O. Salvaña, Y. Sun, D. E. Keyes, M.G. Genton. **High-Performance Statistical Computing (HPSC): Challenges, Opportunities, and Future Directions.** *WIREs Computational Statistics*, 17(4):e70052, 2025. doi:10.1002/wics.70052

Bibliography of “How” examples (3/4)

K. Anderson, D. Kaushik, W. Gropp, B. Smith, D. Keyes. **Achieving High Sustained Performance in an Unstructured Mesh CFD Application.** *Proceedings of SC'99: ACM/IEEE Conference on High Performance Networking and Computing*, ACM/IEEE, 1999. **ACM Gordon Bell Prize.** Source

H. Ltaief, P. Luszczek, A. Haider, J. Dongarra. **Solving the Generalized Symmetric Eigenvalue Problem using Tile Algorithms on Multicore Architectures.** *Parallel Computing 38(1–2):37–51,* 2012.

M. Mortensen, L. Dalcin, D. Keyes*,* **mpi4py-fft: Parallel Fast Fourier Transforms with MPI for Python***, Journal of Open Source Software, 4(36), 1340, doi:10.21105/joss.01340, 2019.*

E. Gorbunov, S. Horváth, P. Richtárik, G. Gidel. **Variance Reduction is an Antidote to Byzantines: Better Rates, Weaker Assumptions and Communication Compression as a Cherry on the Top.** *Proceedings of ICLR 2023* (International Conference on Learning Representations), 2023. arXiv:2206.00529

J. Ren, H. Ltaief, S. Zampini, D. Keyes. **Cheetah: Optimizing Execution Pipelines for Matrix-Free Finite Element Operators on GPUs**, *ACM Proceedings of the International Conference on Supercomputing*, 2026 (to appear).

Bibliography of “How” examples (4/4)

S. Abdulah, A. Baker, G. Bosilca, Q. Cao, S. Castruccio, M. G. Genton, D. E. Keyes, Z. Khalid, H. Ltaief, Y. Song, G. Stenchikov, Y. Sun. **Boosting Earth System Model Outputs and Saving PetaBytes in their Storage using Exascale Climate Emulators.** *Proceedings of SC'25: International Conference for High Performance Computing, Networking, Storage, and Analysis*, ACM/IEEE, 2025. **ACM Gordon Bell Prize.** Source

L. Machiels, J. Peraire, A. T. Patera. **A Posteriori Finite-Element Output Bounds for the Incompressible Navier-Stokes Equations: Application to a Natural Convection Problem.** *Journal of Computational Physics*, 172(2):401–425, 2001. doi:10.1006/jcph.2001.6769

R. Alomairy, S. Abdulah, Q. Cao, M. G. Genton, D. E. Keyes, H. Ltaief. **Sustainably Modeling a Sustainable Future Climate.** *2025 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, September 2025. Source

R. Alomairy, Q. Cao, H. Ltaief, D. Keyes, A. Edelman, **Scalable Hamming Distance Computation Using Accelerated Matrix Transformations.** *ISC High Performance 2025 Research Paper Proceedings* (40th International Conference), Prometeus GmbH, pp. 1–12, 2025. Source


جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology
Thank you!