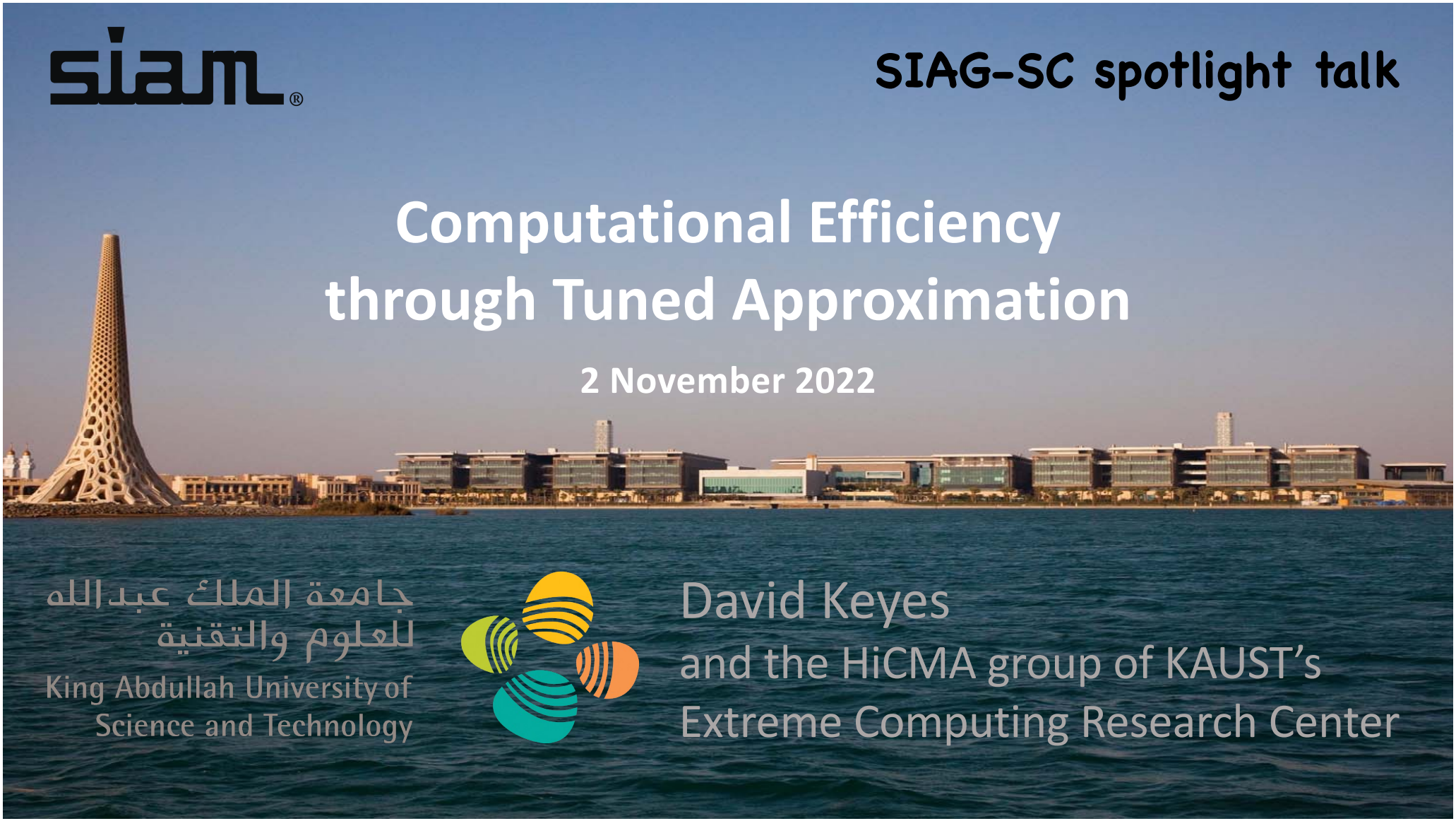


siam®

SIAG-SC spotlight talk

Computational Efficiency through Tuned Approximation

2 November 2022

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

David Keyes
and the HiCMA group of KAUST's Extreme Computing Research Center

Spotlight on

Efficiency

Supercomputing Spotlights is a webinar series presented by the SIAM Activity Group on Supercomputing to focus on raising awareness of high-performance computing opportunities and growing the community. Presentations emphasizing achievements and opportunities in HPC are intended for the broad international community, especially students and newcomers to the field.

feedback to:
david.keyes@kaust.edu.sa

Conclusions, up front

In a world of environmental and financial constraints, in which computational infrastructure demands a growing sector of lab budgets and global energy expenditure, HPC must address the need for *greater efficiency*.

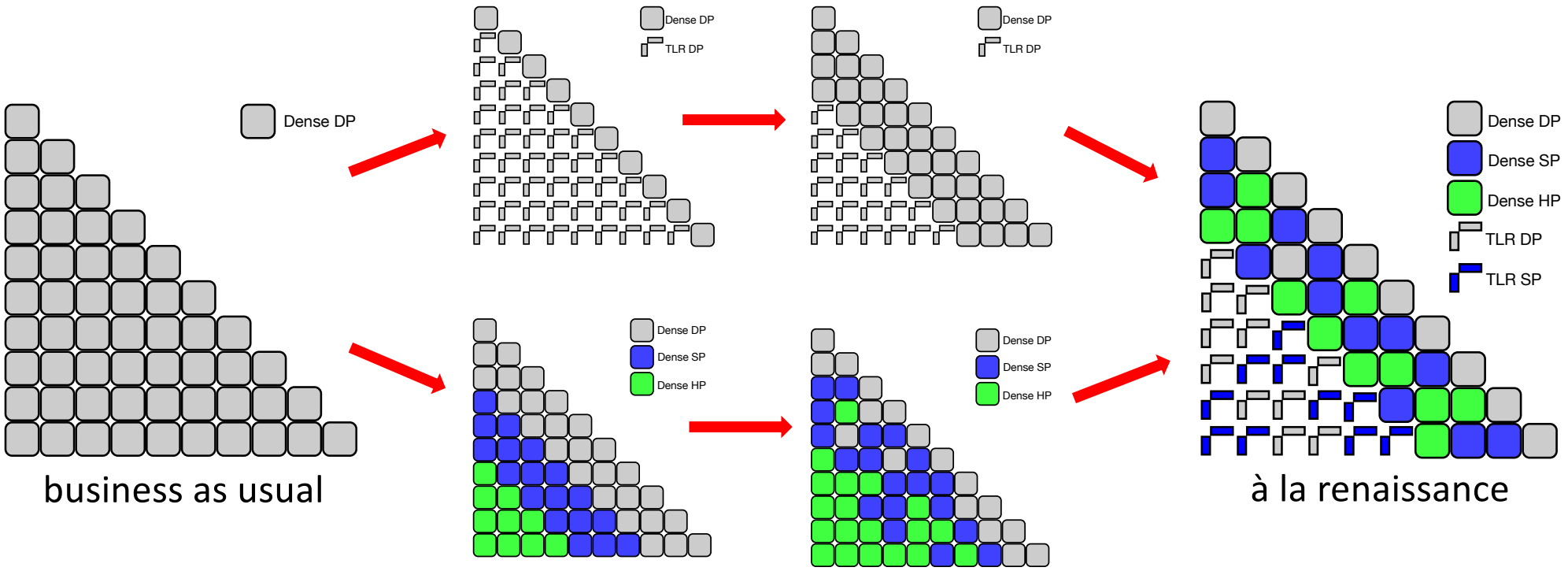
HPC has *excelled* at this historically in

- hardware
- algorithms
- redefining actual outputs of interest in applications

There are *new algorithmic* opportunities in

- reduced rank representations
- reduced precision representations

Computational efficiency through tuned approximation: our journey with *tile low rank* and *mixed precision*

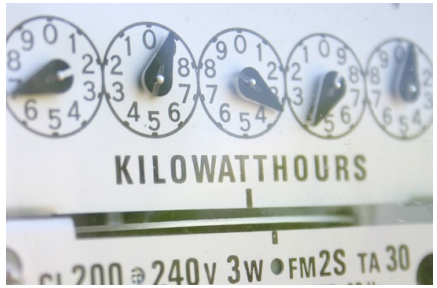


Don’t oversolve: maintain just enough accuracy for the application purpose

Economize on storage: no extra copies of the original matrix

An exaflop/s system is an energy hog

- Frontier (#1 on Top500) delivers about 1 Exaflop/s at about 50 Gigaflop/s per Watt
  - 20 MegaWatts consumed continuously
- Representative electricity cost is about $ 0.20 per KiloWatt-hour
  - $ 200 per MegaWatt-hour
- Powering an exaflop/s system costs about $ 4,000 per hour
  - 10 Kilohour per year (8,760, to be more precise)
  - $40 million annual electricity bill for an exaflop/s system
- Carbon footprint of a KiloWatt-hour is about 0.5 kg CO2-equivalent (improving!)
  - 10,000 kg CO2e hourly carbon footprint for an exaflop/s system
  - 100,000 metric tons CO2e annually
  - equivalent to 20,000 typical passenger cars in the USA



A 10% improvement in computational efficiency implies

- $4 million per year to invest elsewhere
- equivalent of 2,000 cars off the road for year

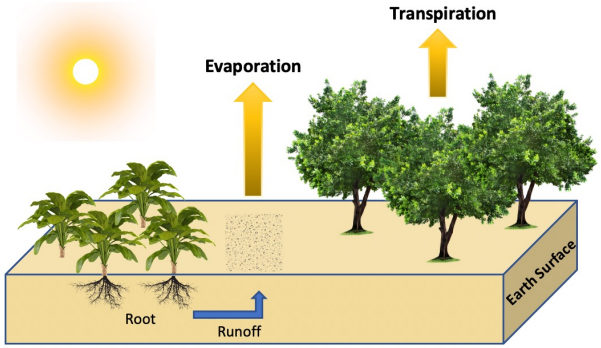
An exaflop/s system is an energy hog

10% is significant!

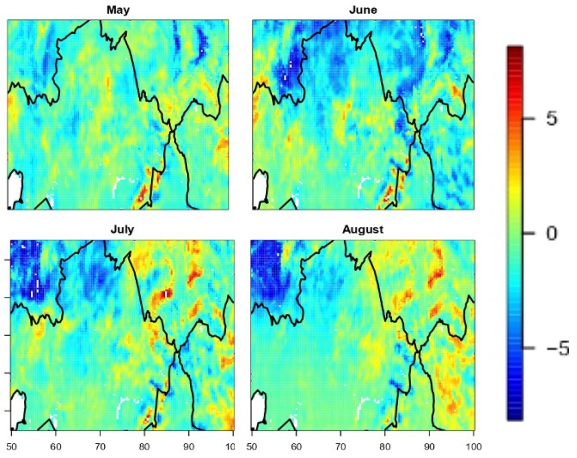
What about 10X?

Efficiency (“science per Joule”) improvement in HPC?

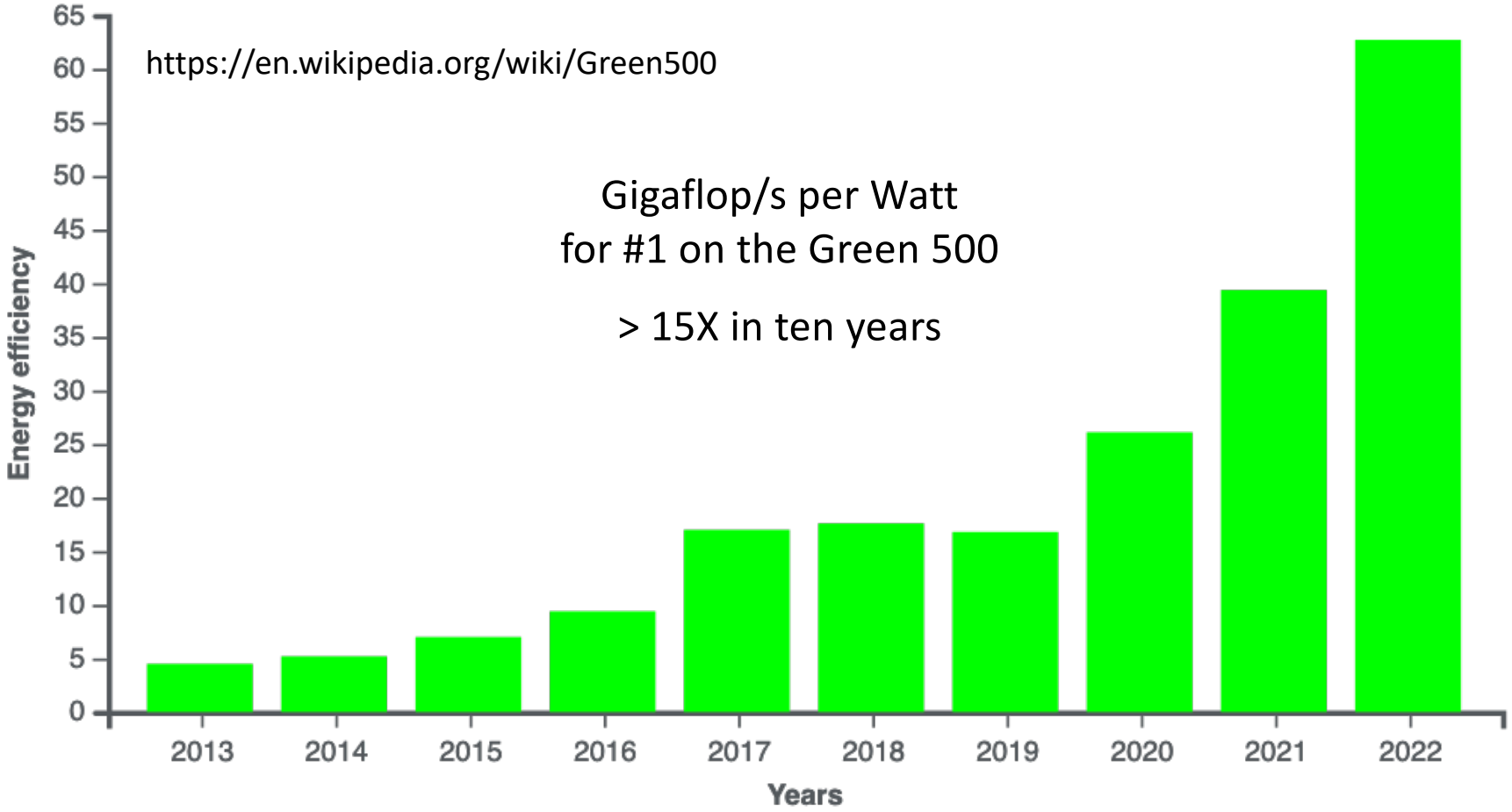
- We consider 3 categories of efficiency improvement
  - from hardware
  - from algorithms
  - from redefining the application objective
- Along the way, we briefly introduce High Performance Statistical Computing (HPSC)
- We preview a 2022 Gordon Bell finalist to spotlight efficiency improvements in kernel linear algebra operations from exploiting
  - rank structure (related to smoothness)
  - precision structure (related to magnitudes)
- We briefly review some properties of the Laplacian
  - for context of efficiency improvements



time series evapotranspiration



HPC hardware efficiency tracked by the Green 500



HPC algorithmic efficiency tracked by Poisson solvers

Consider a Poisson solve in a 3D n x n x n box; natural ordering gives bandwidth of n^2

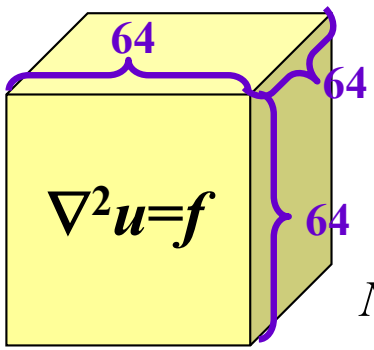
Year	*Method*	*Reference*	*Storage*	*Flops*
1947	GE (banded)	Von Neumann & Goldstine	n^5	n^7
1950	Optimal SOR	Young	n^3	$n^4 \log n$
1971/77	MILU-CG	Reid/Van der Vorst	n^3	$n^{3.5} \log n$
1984	Full MG	Brandt	n^3	n^3

If $n = 64$, this implies an overall reduction in flops of ~16 million *

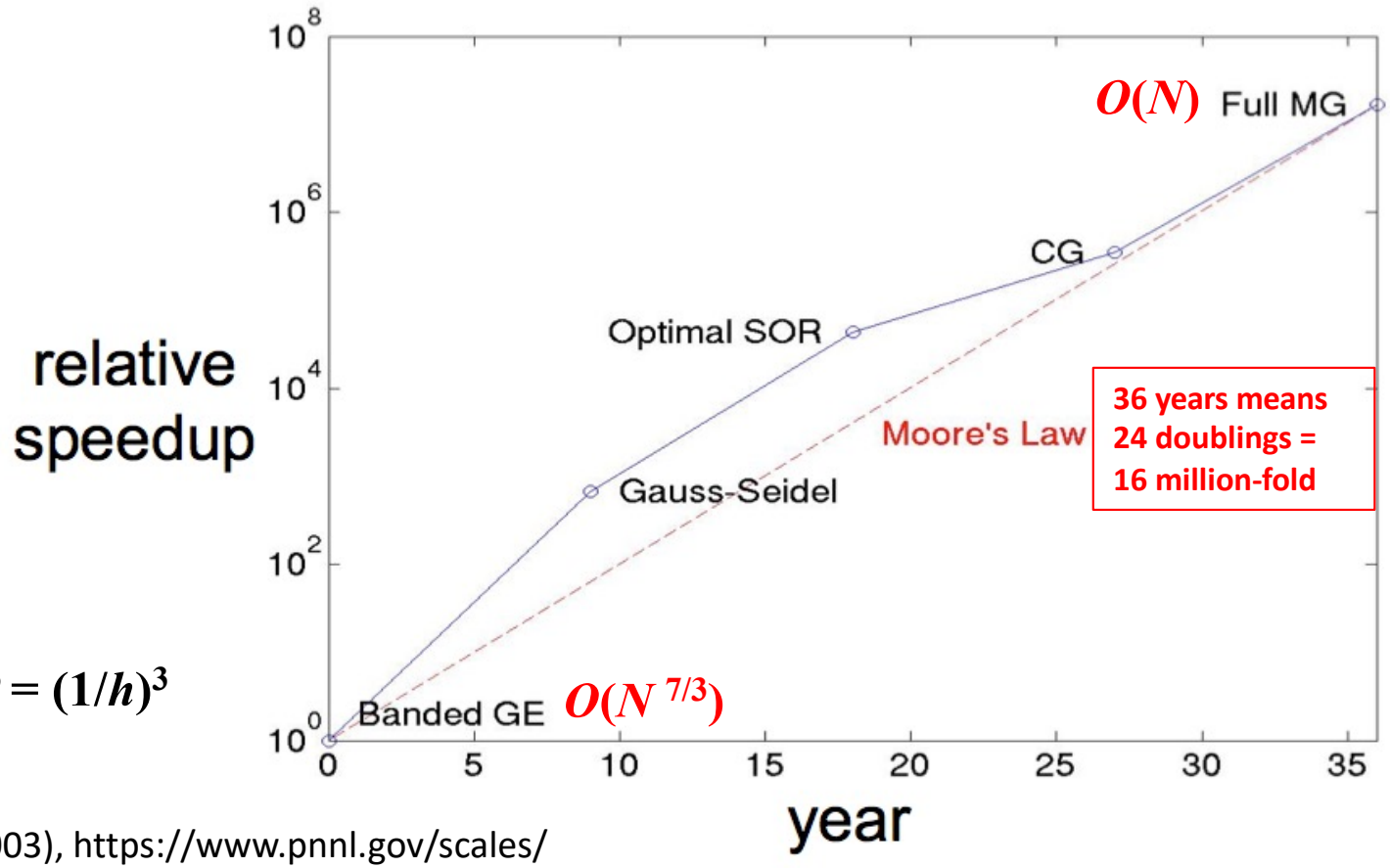
*Six months is reduced to 1 second (recall: 3.154 x 10^7 seconds per year)

"Algorithmic Moore's Law"

HPC progresses even faster in algorithms than in hardware: *example of Poisson's equation in a 3D box with 2nd-order FD*



$N = n^3 = (1/h)^3$



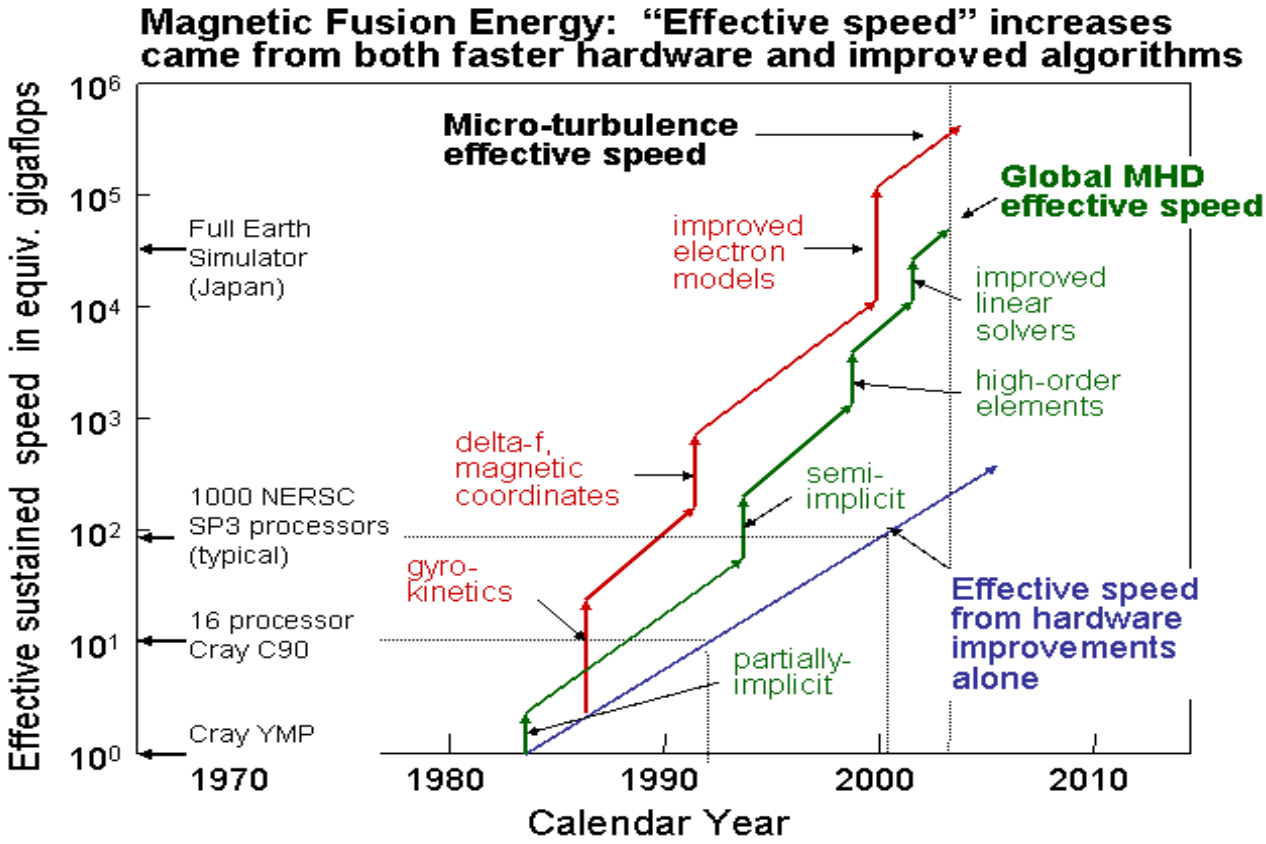
Keyes *et al.*, SCaLeS Rpt. Vol. 1 (2003), https://www.pnnl.gov/scales/

“Algorithmic Moore’s Law” for fusion energy simulations

GKT in red

MHD in green

Moore’s Law in blue



“Semi-implicit”:

All waves treated implicitly, but still stability-limited by transport

“Partially implicit”:

Fastest waves filtered, but still stability-limited by slower waves

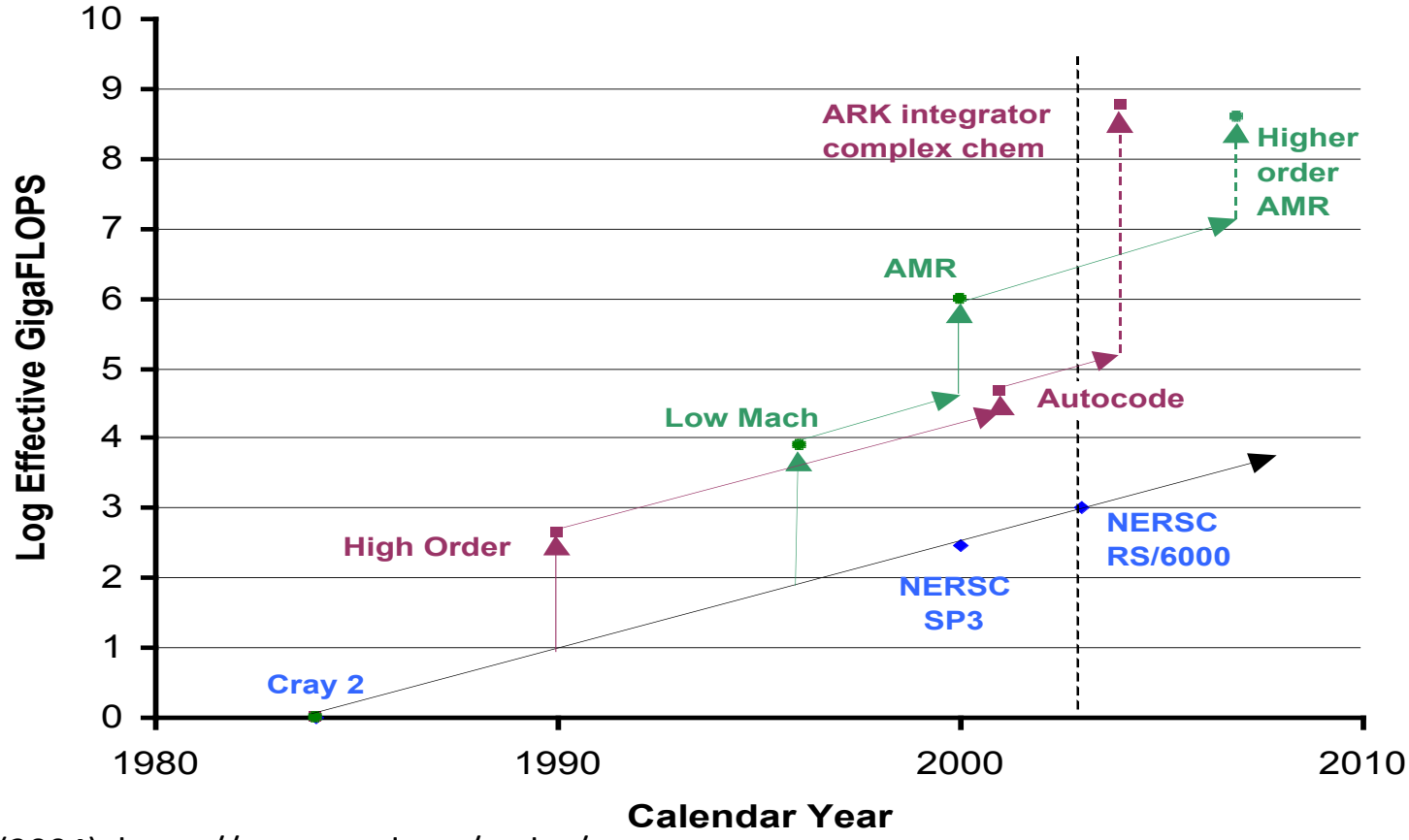
Keyes *et al*., SCaLeS Rpt. Vol 2 (2004), https://www.pnnl.gov/scales/

“Algorithmic Moore’s Law” for combustion simulations

Complex kinetics in maroon

CFD in green

Moore’s Law in blue



Keyes *et al*., SCaLeS Rpt. Vol. 2 (2004), https://www.pnnl.gov/scales/

Algorithms improve exponents; Moore only adjusts the base

To scale to extremes, one must start with algorithms with optimal asymptotic complexity, $O(N \log^p N)$, $p = 0, 1, 2$. These are typically recursively hierarchical.

Some such algorithms through the decades:

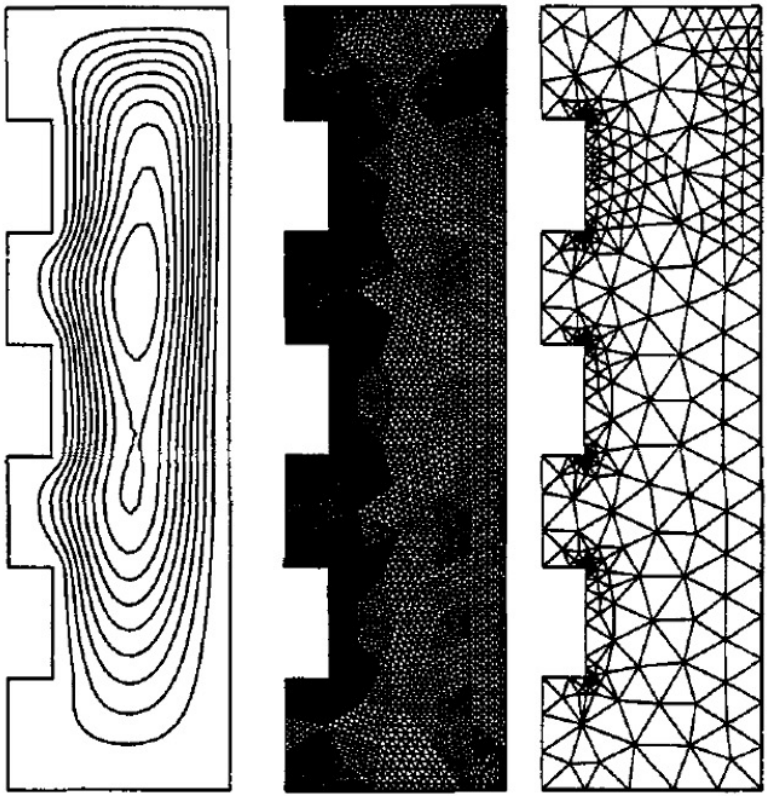
- Fast Fourier Transform (1960's): $N^2 \rightarrow N \log N$
- Multigrid (1970's): $N^{4/3} \log N \rightarrow N$
- Fast Multipole (1980's): $N^2 \rightarrow N$
- Sparse Grids (1990's): $N^d \rightarrow N (\log N)^{d-1}$
- $\mathcal{H}$ matrices (2000's): $N^3 \rightarrow k^2 N (\log N)^2$
- Randomized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

"With great computational power comes great algorithmic responsibility."
– Longfei Gao (PhD, 2013, KAUST AMCS)

Application efficiency from redefining the objective

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.

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



temperature contour | conservative mesh | output bound mesh (flux to 1%)

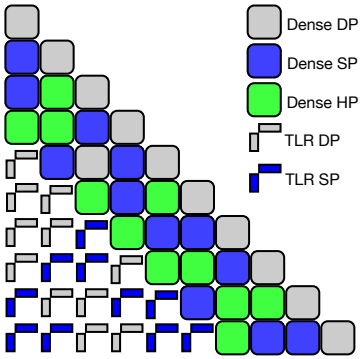
Machiels, Peraire & Patera, *A posteriori FE Output Bounds for the Incompressible NS Equations*, (2001), J. Comp. Phys. **172**:401

Summary so far

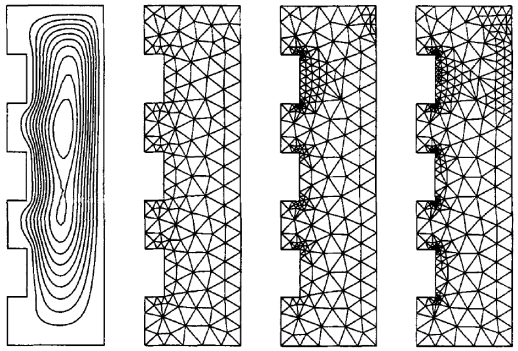
Improving the "science per Joule" (or per unit time) involves:

architecture

algorithm/software



application



In a fortunate world, these are orthogonal: the *desired app* can employ the *best algorithm* on the most *efficient hardware.*

Lessons from the 1D Laplacian

Two concepts that we need to understand in our pursuit of computational efficiency in linear algebra, namely

- conditioning (with its implications on precision)
- rank structure

can be motivated with reference to the 1D Laplacian (to be precise, its negative $-\Delta$), discretized here to second-order in FD, FE, or FV:

$$\begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix}$$

Laplacian has ill-conditioned scaling

Let $n = 1/h$ and consider Dirichlet end conditions with n-1 interior points. Then:

$$\lambda_1 = 2\,[1 - \cos \pi/n] \sim (\pi/n)^2$$
$$\lambda_{n-1} = 2\,[1 - \cos (n-1)\pi/n] \sim 4$$

As n gets large and the mesh resolves more Fourier components, the condition number grows like the square of the matrix dimension (inverse mesh parameter):

$$\kappa = \lambda_{n-1} / \lambda_1 \sim (4/\pi^2)\, n^2$$

In single precision real arithmetic, κ approaches the reciprocal of macheps (10^{-7}) for an n as small as 2^{10} ($\sim 10^3$). Laplacian-like operators arise throughout modeling and simulation (diffusion, electrostatics, gravitation, stress, graphs, etc.), implying $O(1)$ error in the result, so HPC has traditionally demanded double precision by default. GPUs were accepted only when they offered hardware DP (2008, NVIDIA GTX 280).

For the biharmonic, even double precision gives out at $n = 2^{10}$. Some multiscale codes require quadruple precision, often available only in software.

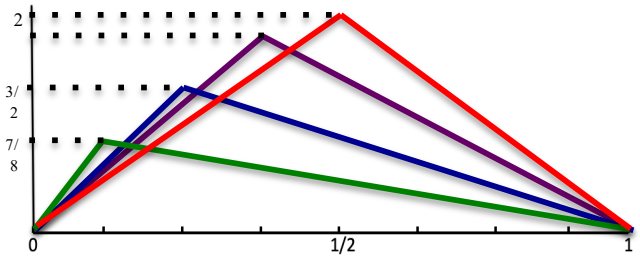
Laplacian has low-rank off-diagonal blocks

A is full-rank, but its off-diagonal blocks have low rank

$$A = \left[\begin{array}{ccc|ccccc} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ \hline & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{array}\right] \leftrightarrow = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \end{bmatrix}$$

Its inverse is dense, but it inherits the same rank structure

$$A^{-1} = \frac{1}{8} \times \left[\begin{array}{cccc|ccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 6 & 12 & 10 & 8 & 6 & 4 & 2 \\ 5 & 10 & 15 & 12 & 9 & 6 & 3 \\ \hline 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 15 & 10 & 5 \\ 2 & 4 & 6 & 8 & 10 & 12 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}\right] \leftrightarrow = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}$$

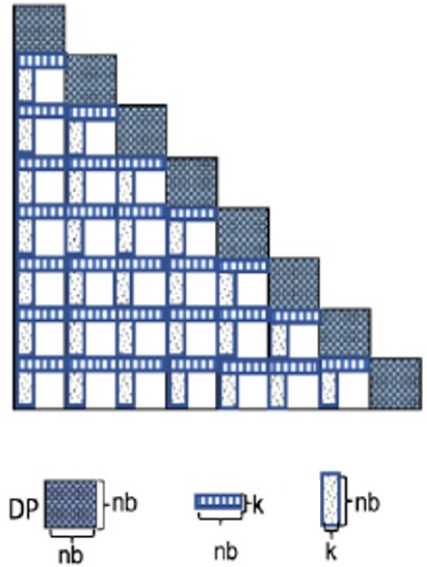


Now: a renaissance in numerical linear algebra (1)

It turns out that many formally dense matrices arising from

- covariances in statistics
- integral equations with displacement kernels
- Schur complements within discretizations of PDEs
- Hessians from PDE-constrained optimization
- nonlocal operators from fractional differential equations
- radial basis functions from unstructured meshing
- kernel matrices from machine learning applications

have exploitable low-rank structure in “most” their off-diagonal blocks.

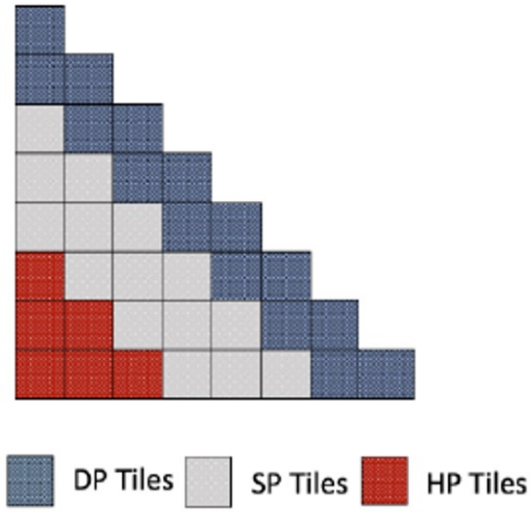


Now: a renaissance in numerical linear algebra (2)

It turns out that many matrices arising in applications have blocks of relatively small norm and can be replaced with reduced precision.

Of course, mixed precision algorithms have a long history, e.g., iterative refinement (1963, Wilkinson), where multiple copies of the matrix are kept in different precisions for different purposes.

There are many such new algorithms; see Higham & Mary, *Mixed precision algorithms in numerical linear algebra*, Acta Numerica (2022).



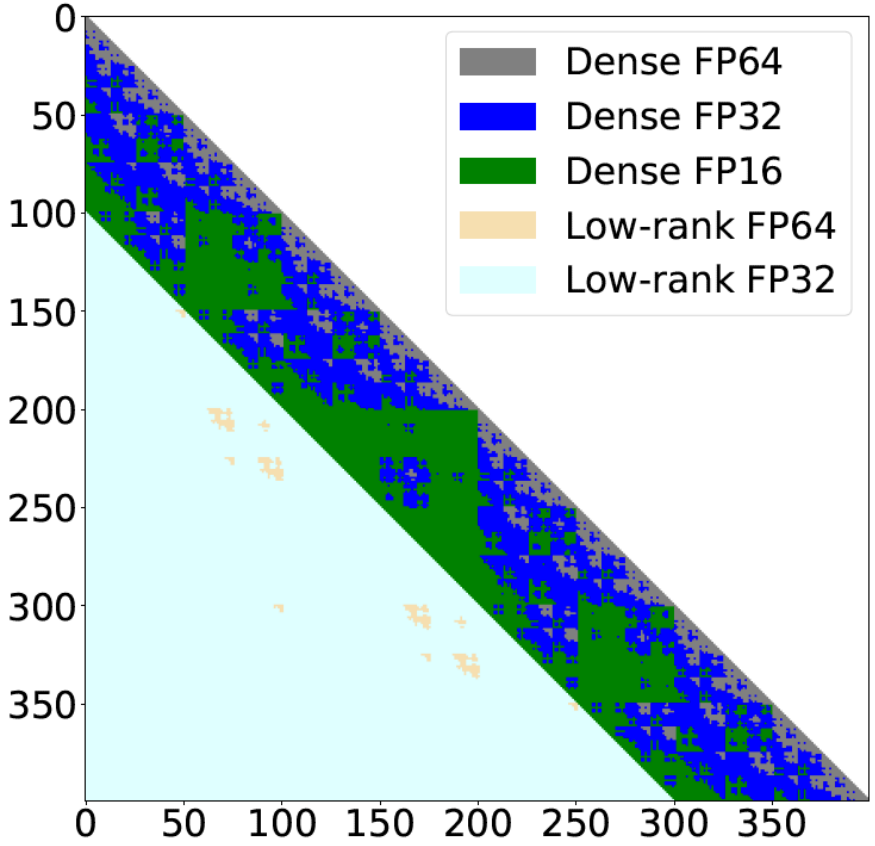
Now: a renaissance in numerical linear algebra (3)

Moreover, these ideas can be combined, as in this 1M x 1M dense symmetric covariance matrix:

- Original in DP: 4 TB
- Replacement: 0.915 TB

Smaller workingsets mean larger problems fit in GPUs and last-level caches on CPUs, for data movement savings

- Also, net computational savings
- Data structures and programs are more complex



Complexities of rank-structured factorizations

- "Straight" *LU* or LDL^T
  - Operations $O(N^3)$
  - Storage $O(N^2)$
- Tile 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 $O(k N)$
  - 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

Rank: a tuning knob

- Replace dense blocks with reduced rank representations, whether “born dense” or as arising during matrix operations
  - use high accuracy (high rank) to build “exact” solvers
  - use low accuracy (low rank) to build preconditioners
- Consider hardware parameters in tuning block sizes and maximum rank parameters, to complement mathematical considerations
  - e.g., cache sizes, warp sizes
- Select from already broad and ever broadening algorithmic menu to form low-rank blocks (next slide)
  - traditionally a flop-intensive vendor-optimized GEMM-based flat algorithm
- Implement in “batches” of leaf blocks
  - flattening trees in the case of hierarchical methods

Low-rank approximations for compressible tiles

Options for forming data sparse representations of the amenable off-diagonal blocks

- standard SVD: $O(n^3)$, too expensive, especially for repeated compressions after additive tile manipulations
- randomized SVD (Halko *et al*., 2011): $O(n^2 \log k)$ for rank k, requires only a small number of passes over the data, saving over the SVD in memory accesses as well as operations
- adaptive cross approximation (ACA) (Bebendorf, 2000): $O(k^2 n \log n)$, motivated by integral equation kernels
- matrix skeletonization (representing a matrix by a representative collection of row and columns), such as CUR, sketching, or interpolatory decompositions based on proxies

Algorithmic opportunities

With such new algorithms, today's HPC can extend many applications that possess

- memory capacity constraints (e.g., geospatial statistics, PDE-constrained optimization)
- energy constraints (e.g., remote telescopes)
- real-time constraints (e.g., wireless communication)
- running time constraints (e.g., chemistry, materials, genome-wide associations)

Dynamic runtimes for HPC implementations

- Uses task graph of sequential code
- Ensures that data dependencies are respected
- Schedules the tasks across appropriate available hardware resources
- Optimizes memory placement for nonuniform access
- Enhances software productivity by abstracting the hardware
- Examples (available for shared memory, distributed memory, and GPUs):
  - OmpSs
    - BSC, Barcelona
    - Pragma-based, extending OpenMP to asynch execution
  - StarPU
    - INRIA, Bordeaux
    - Unified runtime for heterogeneous multicore arch
  - PaRSEC
    - ICL, University of Tennessee
    - Parallel runtime scheduling and execution control

StarPU

PaRSEC

Example: covariance matrices from spatial statistics

- Climate and weather applications have many measurements located regularly or irregularly in a region; prediction is needed at other locations
- Modeled as realization of Gaussian or Matérn spatial random field, with parameters to be fit
- Leads to evaluating, inside an optimization loop, the log-likelihood function involving a large dense (but data sparse) covariance matrix $\boldsymbol{\Sigma}$

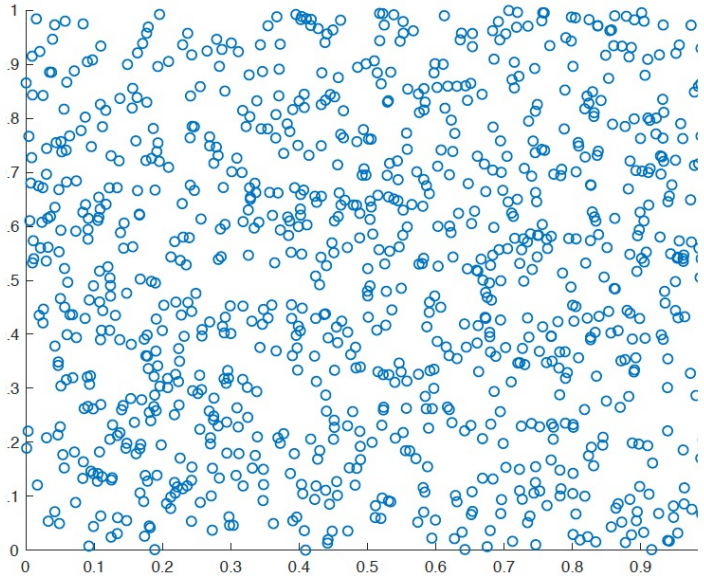
$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{Z}^T\Sigma^{-1}(\boldsymbol{\theta})\mathbf{Z} - \frac{1}{2}\log|\Sigma(\boldsymbol{\theta})|$$

- Apply inverse $\boldsymbol{\Sigma}^{-1}$ and determinant $|\boldsymbol{\Sigma}|$ with Cholesky

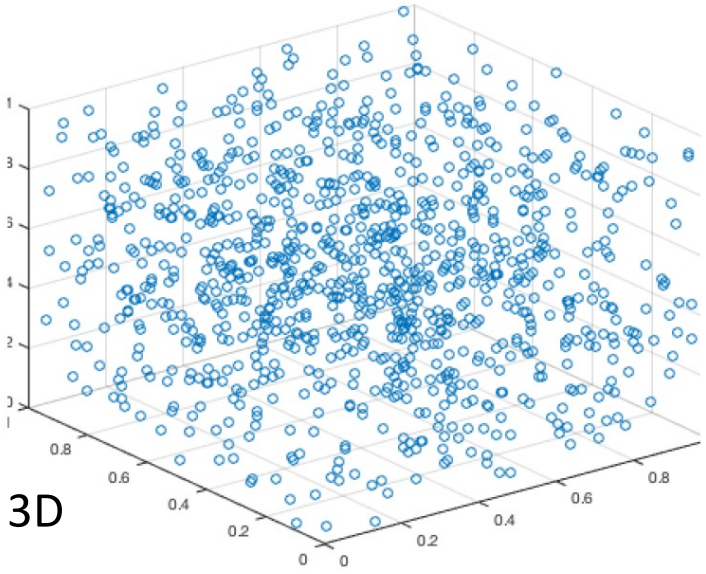
Synthetic scaling test

Random coordinate generation within the unit square or unit cube with Matérn kernel decay, each pair of points connected by square exponential decay, $a_{ij} \sim \exp(-c|x_i - x_j|^2)$

2D



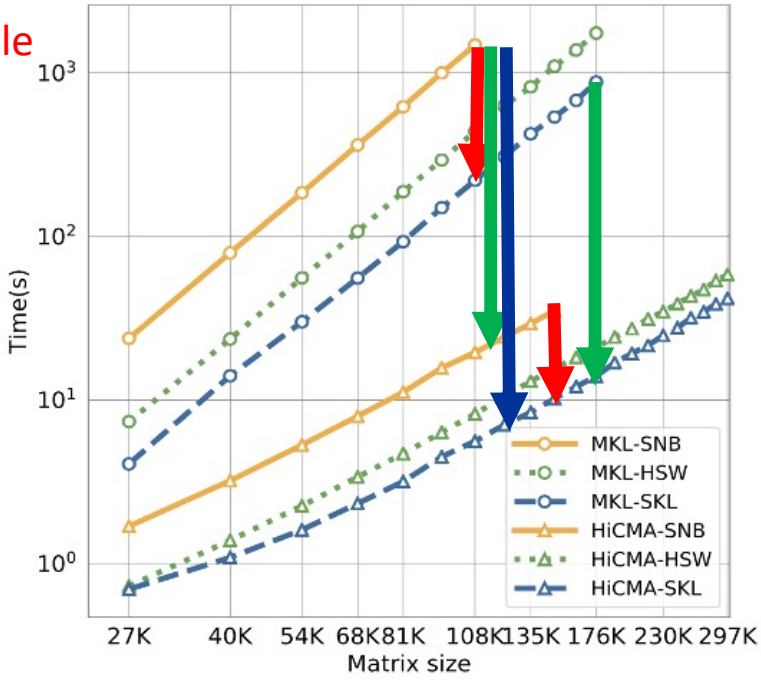
3D



HiCMA TLR vs. Intel MKL on shared memory

- Gaussian kernel to accuracy 1.0e-8 in each tile
- Three generations of Intel manycore (Sandy Bridge, Haswell, Skylake)
- Two generations of linear algebra (classical dense and tile low rank)

NB: log scale



classical

tile low rank w/StarPU

Red arrows: speedups from hardware, same algorithm

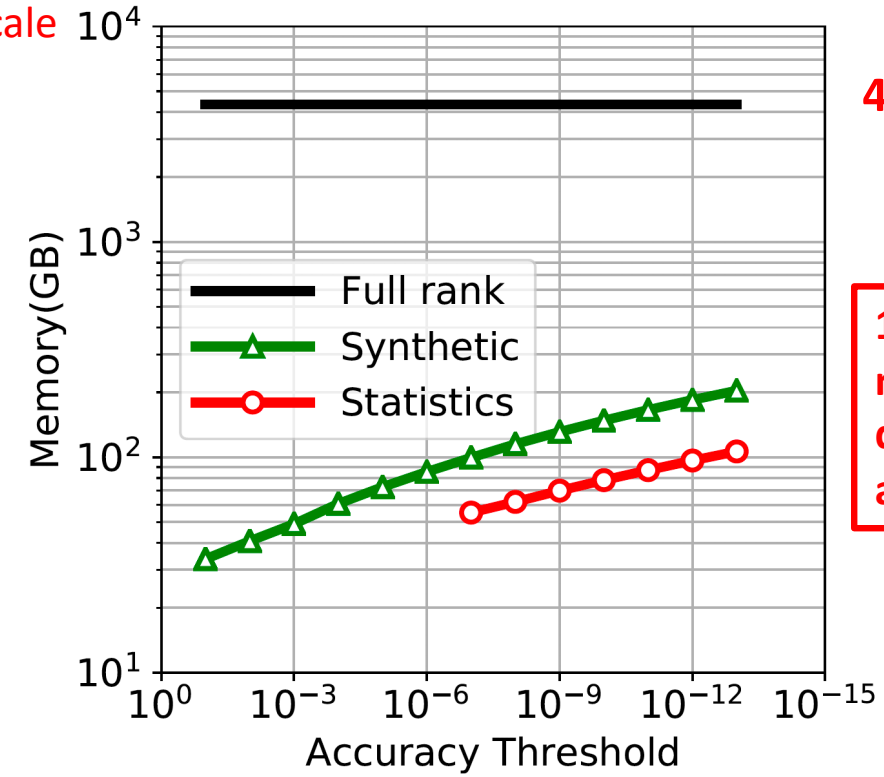
Green arrows: speedups from algorithm, same hardware

Blue arrow: from both

Akbudak, Ltaief, Mikhalev, Charara & K., *Exploiting Data Sparsity for Large-scale Matrix Computations,* Euro-Par 2018

Memory footprint for TLR fully DP matrix of size 1M

NB: log scale



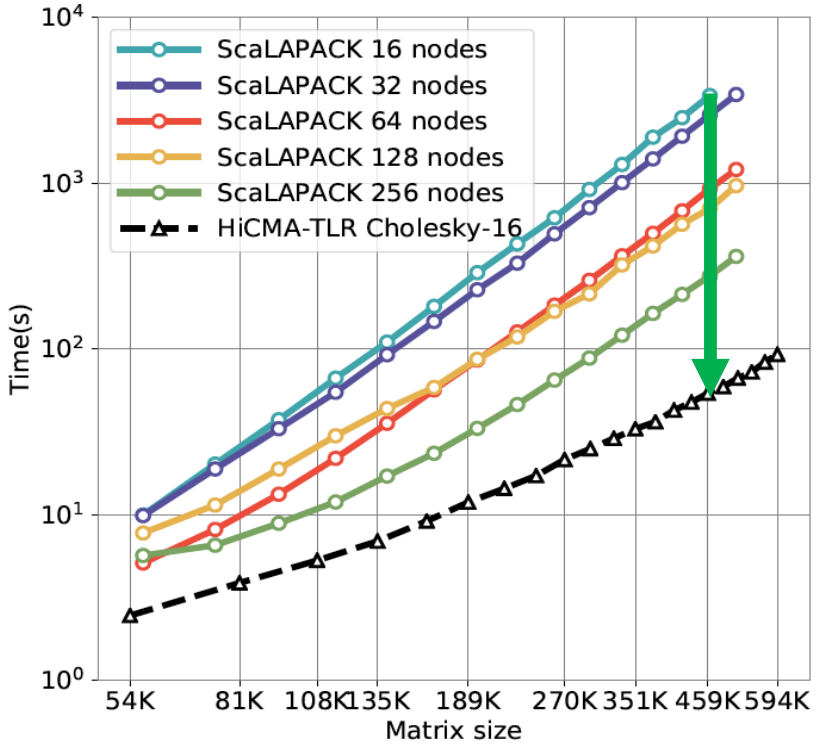
4 TB

1 to 2 orders of magnitude less, depending upon accuracy (x-axis)

Akbudak, Ltaief, Mikhalev, Charara & K., *Exploiting Data Sparsity for Large-scale Matrix Computations*, EuroPar 2018

HiCMA TLR vs. ScaLAPACK on distributed memory

NB: log scale

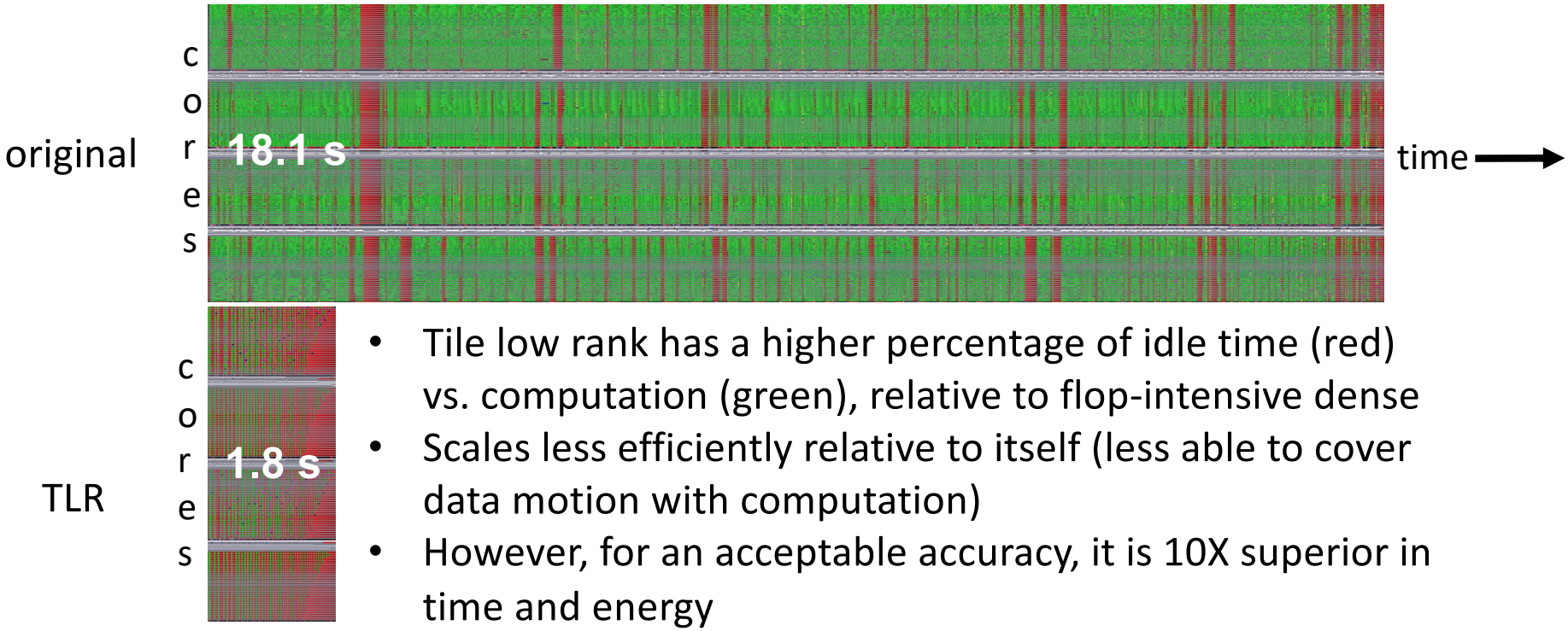


Green arrow: speedup from algorithm, same 16 nodes

Shaheen II at KAUST: a Cray XC40 system with 6,174 compute nodes, each of which has two 16-core Intel Haswell CPUs running at 2.30 GHz and 128 GB of DDR4 main memory

Akbudak, Ltaief, Mikhalev, Charara & K., *Exploiting Data Sparsity for Large-scale Matrix Computations,* Euro-Par 2018

Comparing execution traces for Cholesky factorization



- Tile low rank has a higher percentage of idle time (red) vs. computation (green), relative to flop-intensive dense
- Scales less efficiently relative to itself (less able to cover data motion with computation)
- However, for an acceptable accuracy, it is 10X superior in time and energy

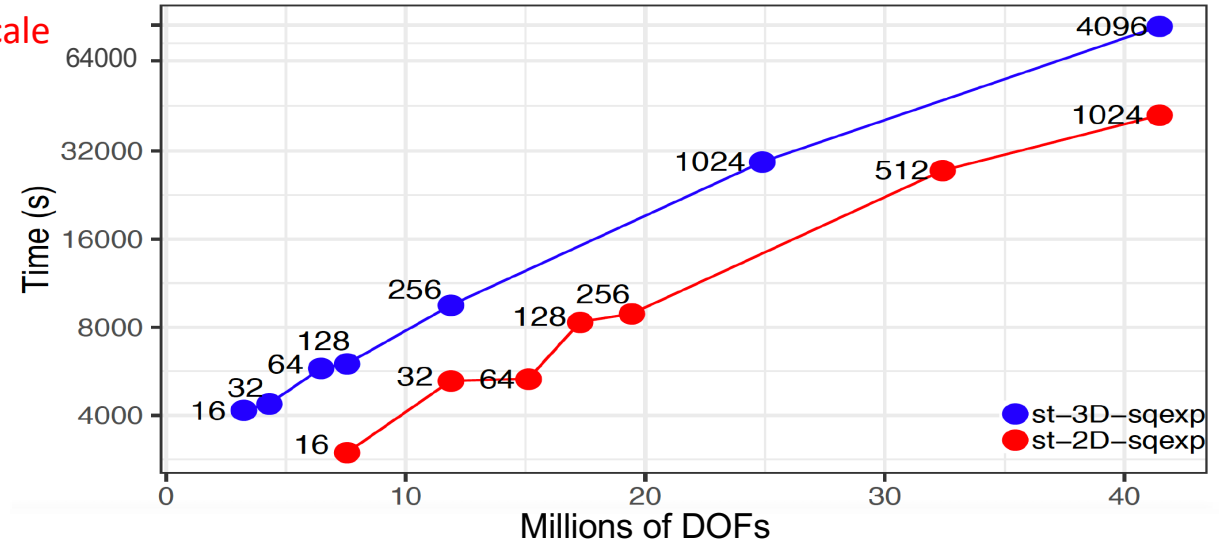
Akbudak, Ltaief, Mikhalev, Charara & K., *Exploiting Data Sparsity for Large-scale Matrix Computations,* Euro-Par 2018

Extreme Tile Low Rank

Cholesky factorization of a TLR matrix derived from Gaussian covariance of random distributions, up to 42M DOFs, on up to 4096 nodes (131,072 cores) of a Cray XC40

- would require 7.05 PetaBytes in dense DP (using symmetry)
- would require 77 days by ScaLAPACK (at the TLR rate of 3.7 Pflop/s)

NB: log scale



Fully dense computation would have cost about $1.03M in electricity and generated about 2500 metric tons of CO2e

Cao, Pei, Akbudak, Mikhalev, Bosilca, Ltaief, K. & Dongarra, *Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications*. PASC'20 (ACM)

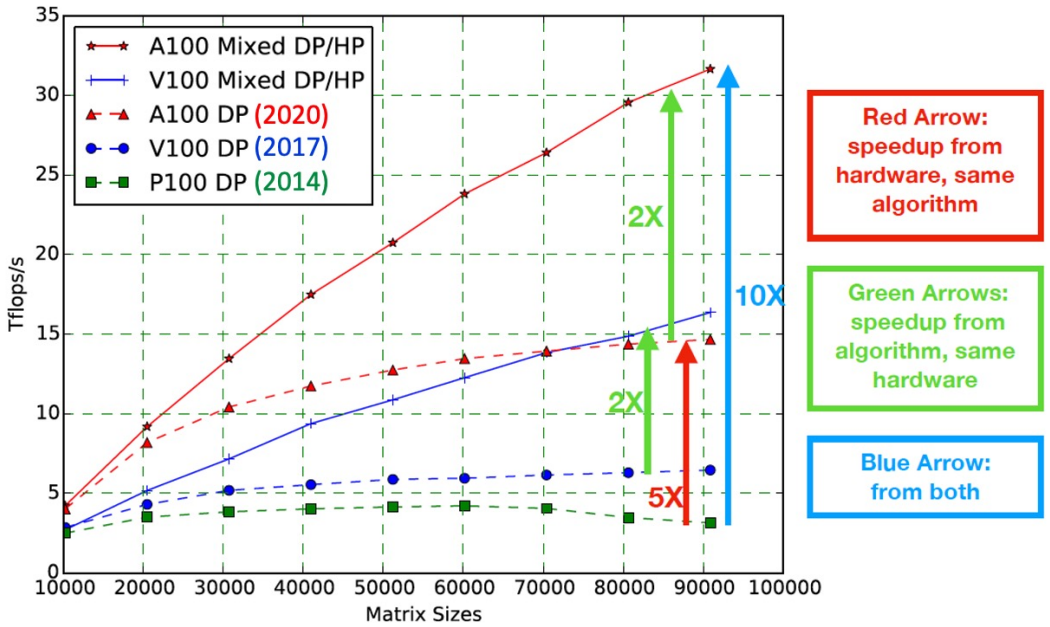
Motivations for mixed precision

- Mathematical: (much) better than “zero precision”
  - Statisticians often approximate remote diagonals as *zero* after performing a diagonally clustered space-filling curve ordering, so their coefficients must be orders of magnitude down from the diagonals
  - not just *smoothly decaying* in the low-rank sense, but actually *small*
- Computational: faster time to solution
  - hence lower energy consumption and higher performance, especially by exploiting heterogeneity

Peak Performance in TF/s	V100 NVLink	A100 NVLink	H100 SXM
FP64	7.5	9.7	34
FP32		19.5	67
FP64 Tensor Core	15	19.5	67
FP32 Tensor Core	8x	156 16x	495 16x
FP16 Tensor Core	120	312	989
	rel. 2017	rel. 2020	rel. 2023

Mixed precision geospatial statistics on GPUs

- Gaussian kernel to accuracy 1.0e-9 in each tile
- Three generations of NVIDIA GPU (Pascal, Volta, Ampere)
- Two generations of linear algebra (double precision and mixed DP/HP)

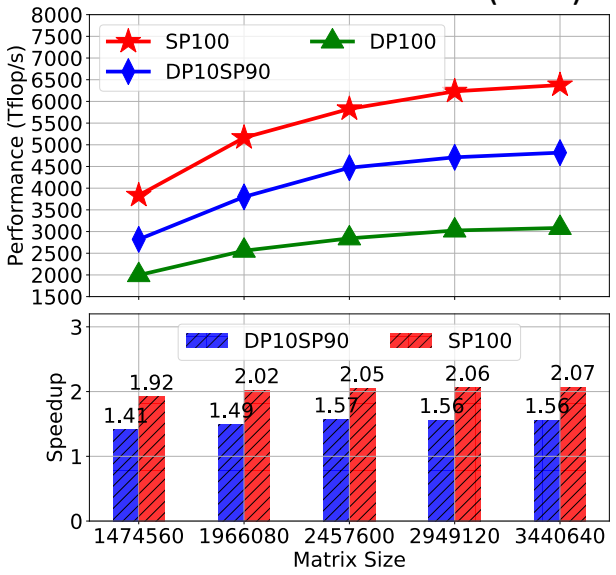


Ltaief, Genton, Gratadour, K. & Ravasi, 2022, *Responsibly Reckless Matrix Algorithms for HPC Scientific Applications*, Computing in Science and Engineering

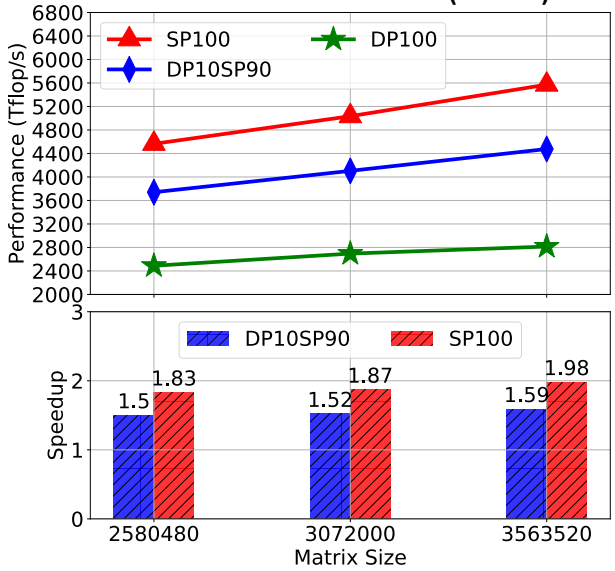
Mixed precision geospatial statistics on distributed memory

- Covariance matrices from 3D geospatial statistics
- Different mixes of DP (from 100% to 1%), SP, and HP on three architectures
- Speedups up to ~2.5X

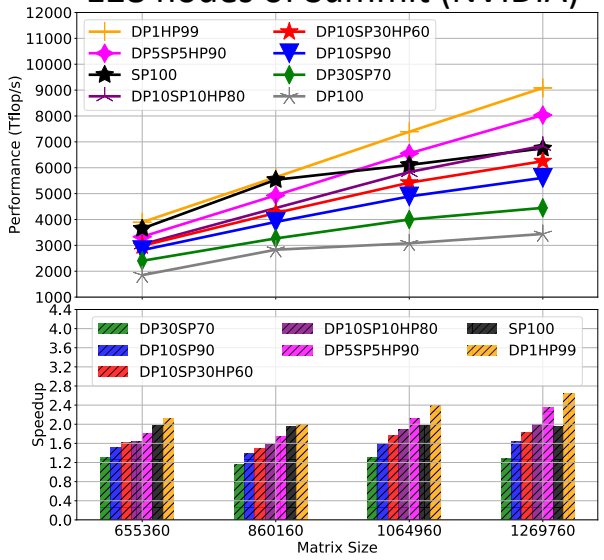
4096 nodes of Shaheen-2 (Intel)



1536 nodes of HAWK (AMD)



128 nodes of Summit (NVIDIA)



Cao *et al.*, *Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications,* ACM PASC'20
Abdulah *et al., Accelerating Geostatistical Modeling and Prediction With Mixed-Precision Computations,* IEEE TPDS'21

2022 Gordon Bell Prize finalist story

Geostatistical Modeling and Inference at Extreme Scales via Tile Low Rank and Mixed Precision

Qinglei Cao
Yu Pei
George Bosilca
Jack Dongarra *

Innovative Computing Laboratory, UTK

Sameh Abdulah
Rabab Alomairy
Pratik Nag
Hatem Ltaief
Ying Sun
Marc Genton
David Keyes

Extreme Computing Research Center, KAUST

* 2022 Turing Award Recipient

Finalist paper

Reshaping Geostatistical Modeling and Prediction for Extreme-Scale Environmental Applications

Qinglei Cao[2,6], Sameh Abdulah[1,5], Rabab Alomairy[1,5], Yu Pei[2,6], Pratik Nag[1,5], George Bosilca[2,7], Jack Dongarra[2,3,4,7], Marc G. Genton[1,5], David E. Keyes[1,5], Hatem Ltaief[1,5], and Ying Sun[1,5]

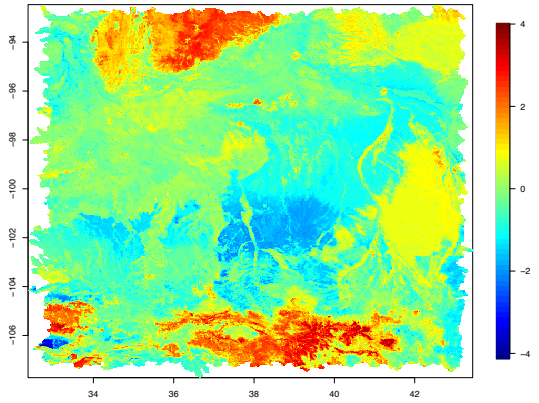
II. PERFORMANCE ATTRIBUTES

Performance Attributes	Our submission
Problem Size	Nine million geospatial locations[1]
Category of achievement	Time-to-solution and scalability
Type of method used	Maximum Likelihood Estimation (MLE)
Results reported on basis of	Whole application
Precision reported	Double, single, and half precision
System scale	16K Fujitsu A64FX nodes of Fugaku[1]
Measurement mechanism	Timers; FLOPS; Performance modeling

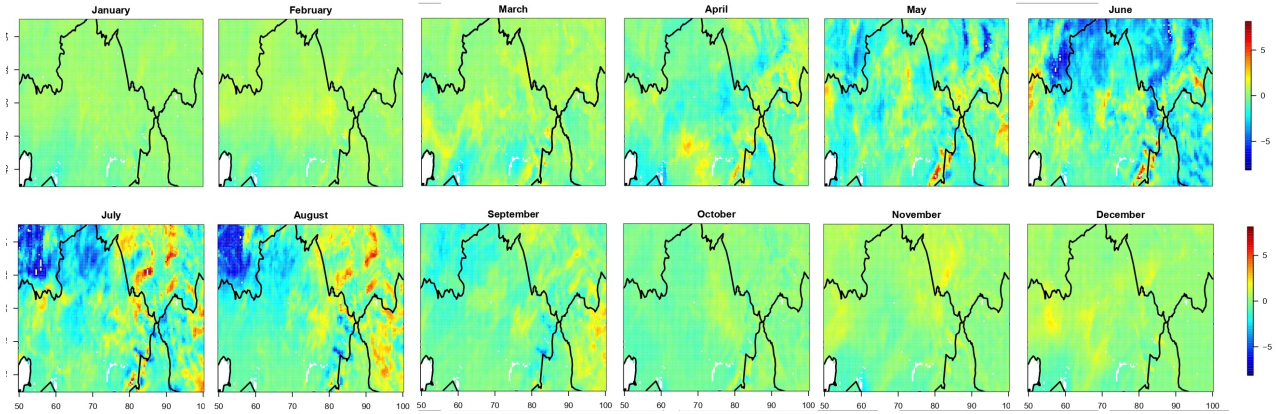
Performance results herein are not final, to be improved with more access to tune on a top system

App: spatial & spatio-temporal environmental statistics

Space and space-time modeling using Maximum Likelihood Estimation (MLE) on two environmental datasets



2D soil moisture data
at the top layer of the
Mississippi River basin



2021 monthly evapotranspiration (ET)
over Central Asia

[means are subtracted out in these graphs]

Statistical “emulation” (complementary to simulation)

- Predicts quantities directly from data (*e.g.*, weather, climate)
  - assumes a correlation model
  - data may be from observations or from first-principles simulations
  - statistical alternative to large-ensemble simulation averages
- Relied upon for economic and policy decisions
  - predicting demands, engineering safety margins, mitigating hazards, siting renewable resources, etc.
  - such applications are among principal supercomputing workloads
- Whereas simulations based on PDEs are usually memory bandwidth-bound, emulations based on covariance matrices are usually compute-bound (achieve a high % of bandwidth peak)

The computational ~~challenge~~ opportunity

- Contemporary observational datasets can be huge
  - Collect p observations at each of n locations $Z_p(x_n, y_n, z_n, t_n)$
  - Find optimal fit of the observations Z to a plausible function
  - Infer values at missing locations of interest
- Maximum Likelihood Estimate (MLE)
  - model for estimating parameters required to perform inference
- Complexity:
  - Arithmetic cost: solve systems with and calculate determinant of n-by-n covariance matrix
  - $O((pn)^3)$ floating-point operations and $O((pn)^2)$ memory
  - Memory footprint: 10^6 locations require 4 TB memory (double precision, invoking symmetry, for $p=1$)

Motivation: High Performance Computational Statistics (HPCS)

"Increasing amounts of data are being produced (e.g., by remote sensing instruments and numerical models), while techniques to handle millions of observations have historically lagged behind… Computational implementations that work with irregularly-spaced observations are still rare." - Dorit Hammerling, NCAR, July 2019

1M × 1M dense sym DP matrix requires 4 TB, $N^3 \sim 10^{18}$ Flops

Traditional approaches:

Global low rank

Zero outer diagonals

Better approaches:

Hierarchical low rank

Reduced precision outer diagonals

Is this problem important?

The potential for this combination in spatial statistics generally is high… The authors have demonstrated ***controllable and high accuracy typical of universal double precision, while exploiting mostly half precision, and keeping relatively few tiles clustered around the diagonal in their original fully dense format.*** *The result is reduction in time to solution of an order of magnitude or more, with the ratio of improvement growing with problem size, but already transformative.*

-- Professor Sudipto Banerjee, UCLA

Is this problem important?

The innovations described in numerical linear algebra and in dynamic runtime task scheduling deliver an order of magnitude or more of reduction in execution time for a sufficiently large spatial or spatial-temporal data set using the Maximum Likelihood Estimation (MLE) and kriging paradigm. Perhaps more importantly, ***by reducing the memory footprint of such models, they allow much larger datasets to be accommodated within given computational resources. The advance this creates for spatial statisticians – geophysical and otherwise – is potentially immense****, given that this result is now available through ExaGeoStat.*

--Professor Doug Nychka, Colorado School of Mines

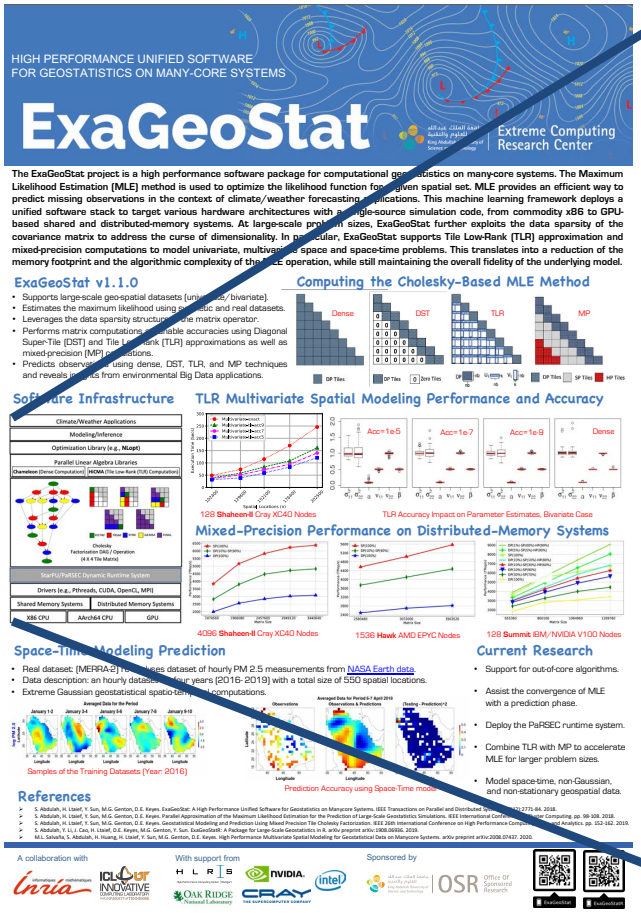
Is this problem important?

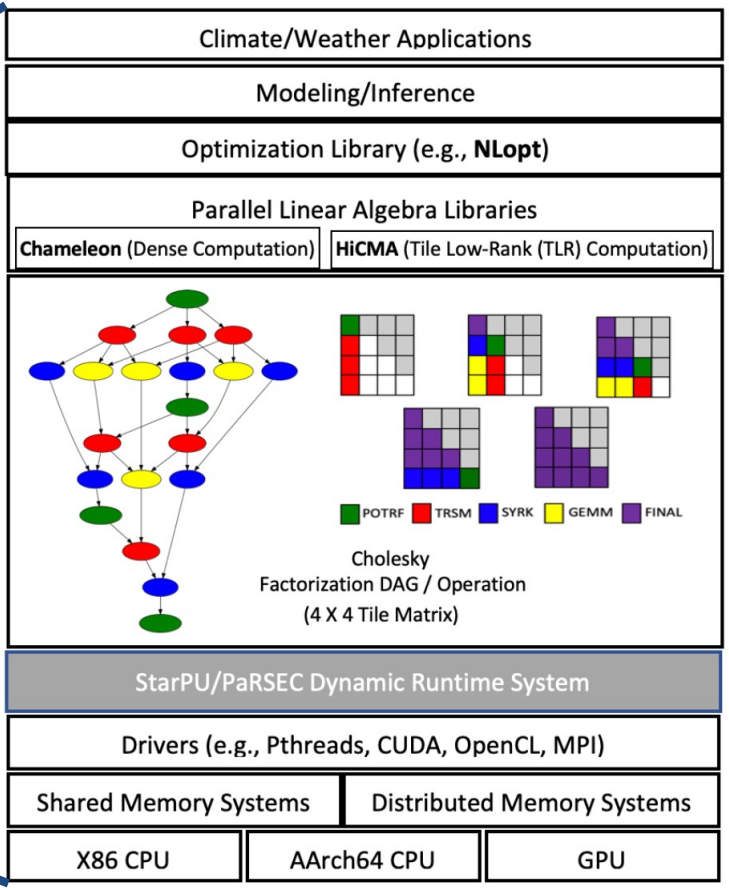
An especially attractive aspect of the submission is the innovation that it required in the a64fx ARM architecture of Fugaku, namely the accumulation in 32 bits of the 16-bit floating point multiply. I regard this aspect of the KAUST-UT-RIKEN collaboration of abiding benefit beyond the particular application of this submission.

As you know, my mottos for data science are that "Statistics is the 'Physics' of Data" and "Statistics is to Machine Learning as Physics is to Engineering." ***Your Gordon Bell campaign is accelerating the use of spatial statistics to allow it to keep up with exascale hardware.***

-- Dr. George Ostrouchov, ORNL

https://github.com/ecrc/exageostat

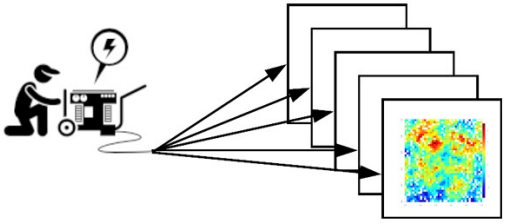


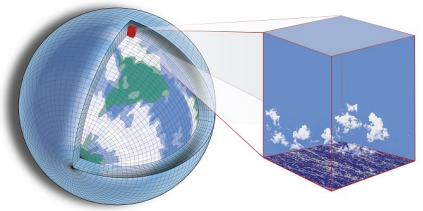


Sameh Abdulah,
Research Scientist
ECRC, KAUST

ExaGeoStat's 3-fold framework

- Synthetic Dataset Generator
  - Generates large-scale geospatial datasets which can be used separately as benchmark datasets for other software packages
- Maximum Likelihood Estimator (MLE)
  - Evaluates the maximum likelihood function on large-scale geospatial datasets
  - Supports dense full machine precision, Tile Low-Rank (TLR) approximation, low-precision approximation accuracy, and now TLR-MP
- ExaGeoStat Predictor
  - Infers unknown measurements at new geospatial locations from the MLE model





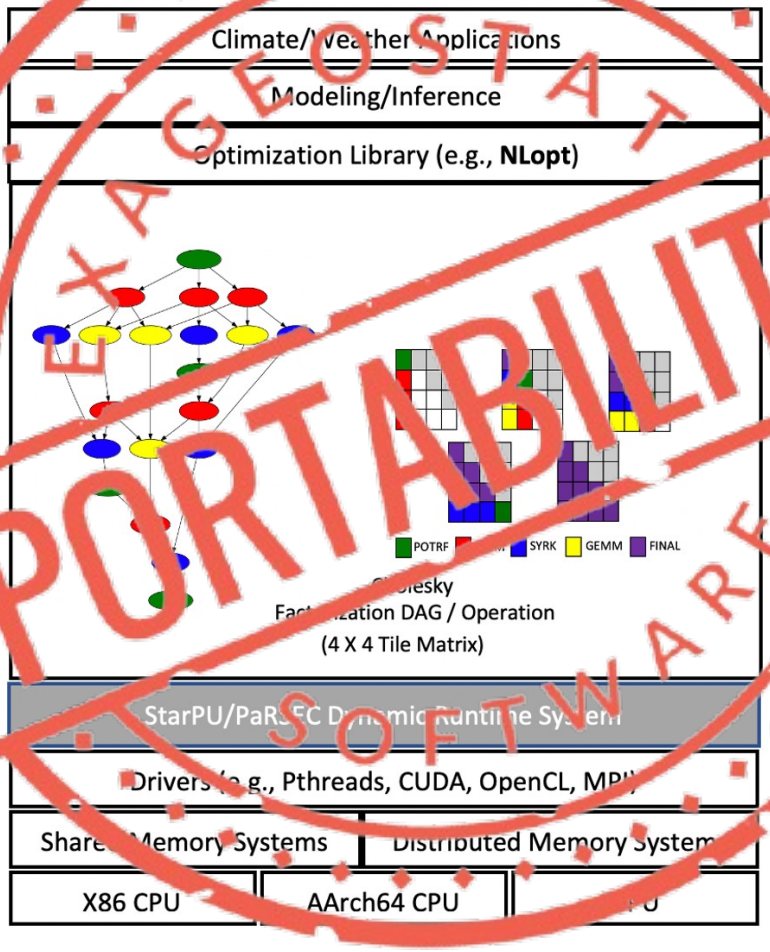
The portable ExaGeoStat software stack

Fujitsu A64FX

NVIDIA V 00

AMD EPYC

Intel X86



#1 Fugaku

#2 Summit

#2 HAWK

#89 Shaheen-2

Maximum Likelihood Estimator (MLE)

- The log-likelihood function: $\ell(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\mathbf{Z}^{\top}\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{Z}.$
- Optimization over $\boldsymbol{\theta}$ to maximize the likelihood function estimation until convergence
  - generate the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ using a specified kernel
  - evaluate the log determinant and the inverse operations, which require a Cholesky factorization of the given covariance matrix
  - update $\boldsymbol{\theta}$
- NLOPT* is typically used to maximize the likelihood
- Parallel PSwarm optimization algorithm runs several likelihood estimation steps at the same time (an embarrassingly parallel outer loop)

*open-source library by Prof. Steve Johnson of MIT

Covariance functions supported in ExaGeoStat

Univariate Matern Kernel

$$C(r;\boldsymbol{\theta}) = \frac{\theta_1}{2^{\theta_3-1}\Gamma(\theta_3)}\left(\frac{r}{\theta_2}\right)^{\theta_3}\mathcal{K}_{\theta_3}\left(\frac{r}{\theta_2}\right)$$

(3 parameters to fit: variance, range, smoothness)

Space/Time Nonseparable Kernel

$$C(\mathbf{h},u) = \frac{\sigma^2}{a_t|u|^{2\alpha}+1}\mathcal{M}_\nu\left\{\frac{\|\mathbf{h}\|/a_s}{(a_t|u|^{2\alpha}+1)^{\beta/2}}\right\},$$

(6 parameters to fit, add: time-range, time-smoothness, and separability)

Multivariate Parsimonious Kernel

$$C_{ij}(\|\mathbf{h}\|;\boldsymbol{\theta}) = \frac{\rho_{ij}\sigma_{ii}\sigma_{jj}}{2^{\nu_{ij}-1}\Gamma(\nu_{ij})}\left(\frac{\|\mathbf{h}\|}{a}\right)^{\nu_{ij}}\mathcal{K}_{\nu_{ij}}\left(\frac{\|\mathbf{h}\|}{a}\right)$$

Tukey g-and-h Non-Gaussian Field with Kernel

$$\rho_Z(h) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(4\sqrt{2\nu}\frac{h}{\phi}\right)^{\nu}\mathcal{K}_\nu\left(4\sqrt{2\nu}\frac{h}{\phi}\right)$$

Multivariate Flexible Kernel

$$C(\mathbf{h};u) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)\,(a|u|^{2\alpha}+1)^{\delta+\beta d/2}}\left(\frac{c\|\mathbf{h}\|}{(a|u|^{2\alpha}+1)^{\beta/2}}\right)^{\nu} \times K_\nu\left(\frac{c\|\mathbf{h}\|}{(a|u|^{2\alpha}+1)^{\beta/2}}\right), \quad (\mathbf{h};u)\in\mathbb{R}^d\times\mathbb{R},$$

Powered Exponential Kernel

$$C(r;\boldsymbol{\theta}) = \theta_0\exp\left(\frac{-r^{\theta_2}}{\theta_1}\right)$$

How to choose the rank?

- Tiles are compressed to low rank based on user-supplied tolerance parameter, based on the first neglected singular value-vector pair.
- A tile-centric, structure-aware heuristic decides at runtime whether the tile should remain in low rank form or converted back to dense, based on estimates of the overheads of maintaining and operating with the compressed form.
- The structure-aware runtime decision is based only the estimated number of flops and time to solution, while the precision-aware runtime decision (next slide) is based only on the accuracy requirements of representing the matrix in the Frobenius norm.

How to choose the precision?

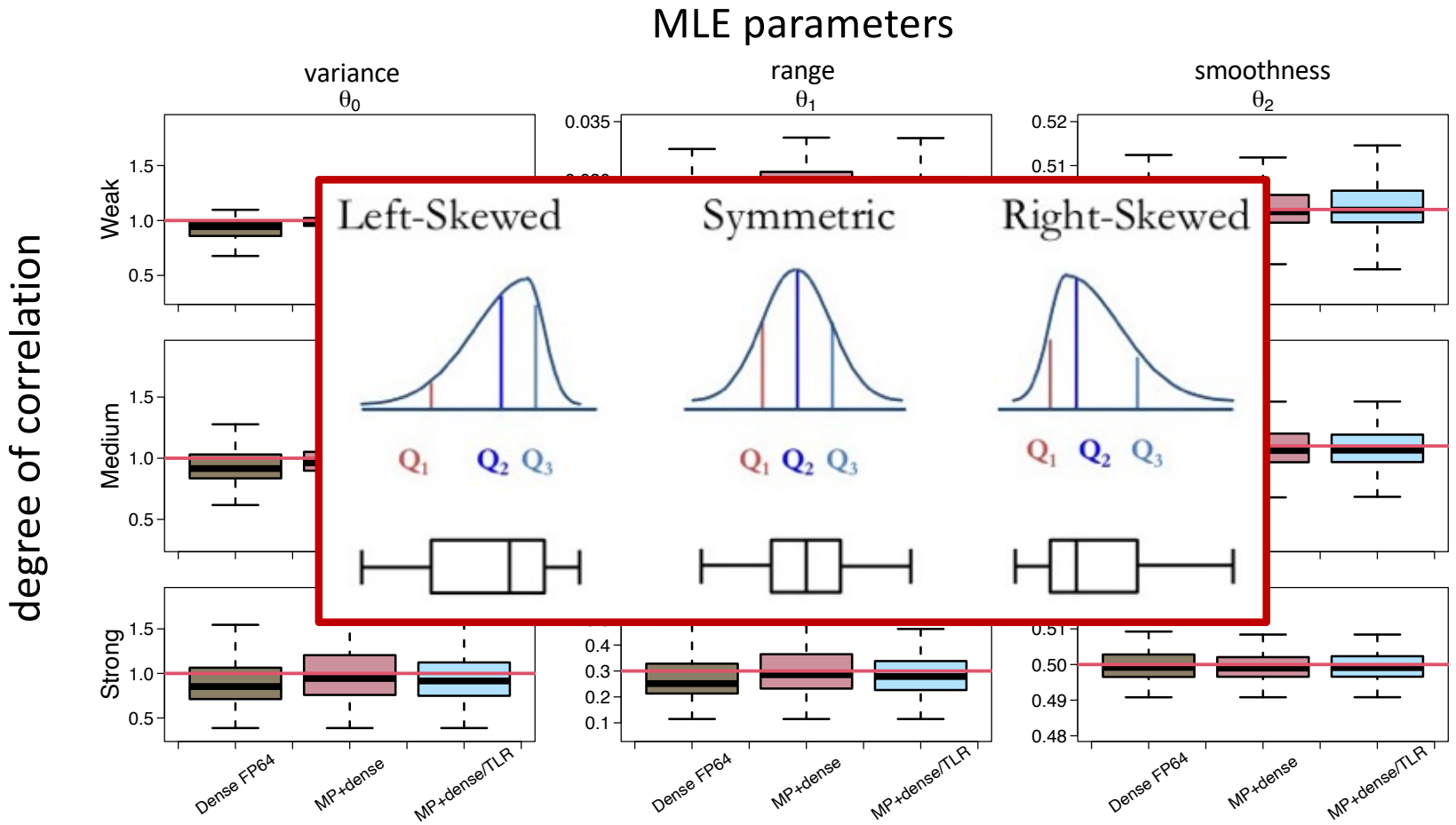
- Consider 2-precision case, with machine epsilons (unit roundoffs) u_{high} and u_{low}, resp.
- Let $\| A \|_F$ be the Frobenius norm of the global matrix square matrix A, which is computable by streaming A through just once
- Let n_T be the number of tiles in each dimension of A
- Then any tile A_{ij} such that $n_T \| A_{ij} \|_F / \| A \|_F < u_{high} / u_{low}$ is stored in low precision; otherwise kept in high
- The mixed precision tiled matrix $\mathcal{A}$ thus formed satisfies

$$\| \mathcal{A} - A \|_F < u_{high} \| A \|_F$$

- Generalizes to multiple precisions
- Tiles can be converted dynamically at runtime

Higham & Mary, *Mixed Precision Algorithms in Numerical Linear Algebra* (2022), Acta Numerica, pp. 347-414

Accuracy on synthetic 2D space dataset



Accuracy on real 3D (2D space + time) dataset

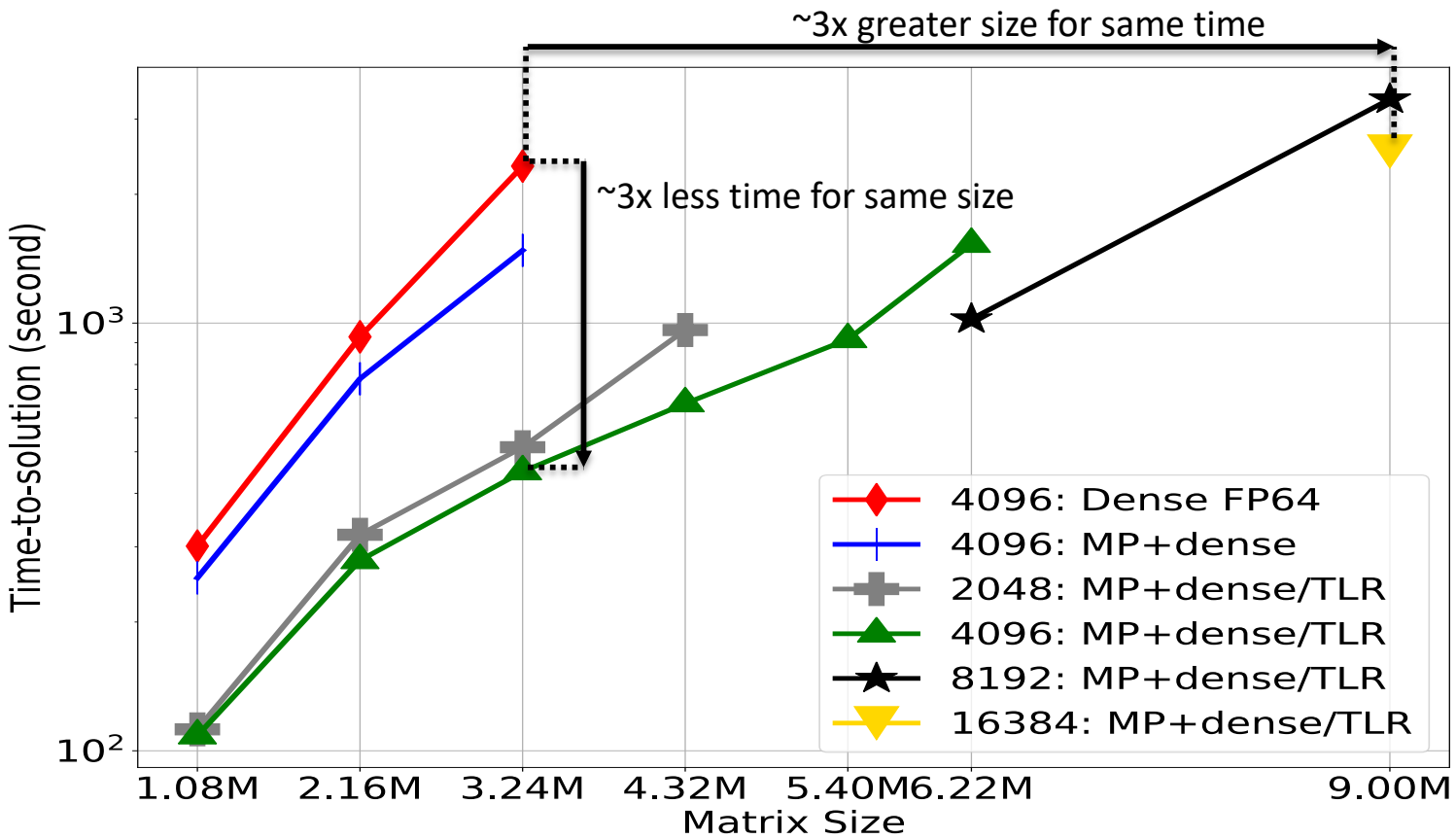
Variants	Variance (θ_0)	Range (θ_1)	Smoothness (θ_2)
Dense FP64	1.0087	3.7904	0.3164
MP+dense	0.9428	3.8795	0.3072
MP+dense/TLR	0.9247	3.7756	0.3068

Variants	Range-time (θ_3)	Smoothness-time (θ_4)	Nonsep-param (θ_5)
Dense FP64	0.0101	3.4890	0.1844
MP+dense	0.0102	3.4941	0.1860
MP+dense/TLR	0.0102	3.5858	0.1857

Variants	Log-Likelihood (llh)	MSPE
Dense FP64	-136675.1	0.9345
MP+dense	-136529.0	0.9348
MP+dense/TLR	-136541.8	0.9428

mean-square prediction error

Performance on up to 16K nodes of Fugaku



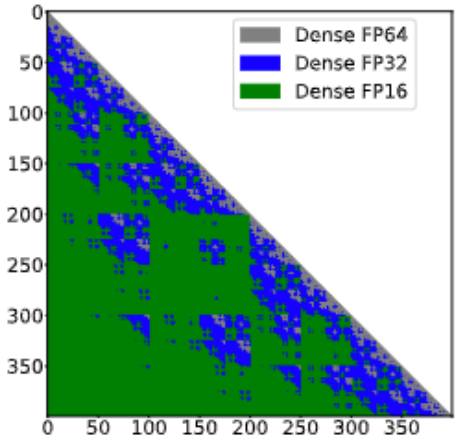
To be improved:

Still tuning runtime system PaRSEC on Fugaku's 32GB/node

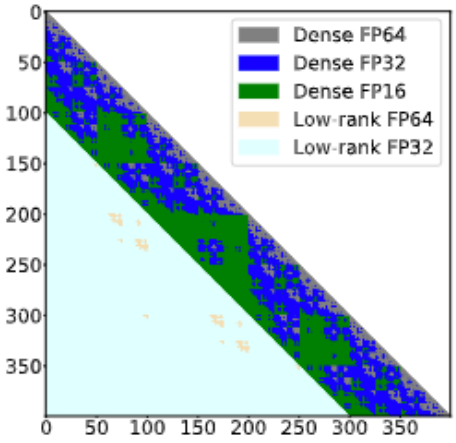
Tile map for 2D space kernel with ~1M points

370 tiles of size 2700 in each dimension

weak correlation

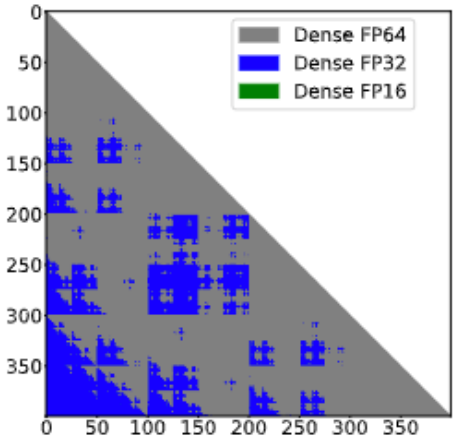


memory footprint
1.6 TB

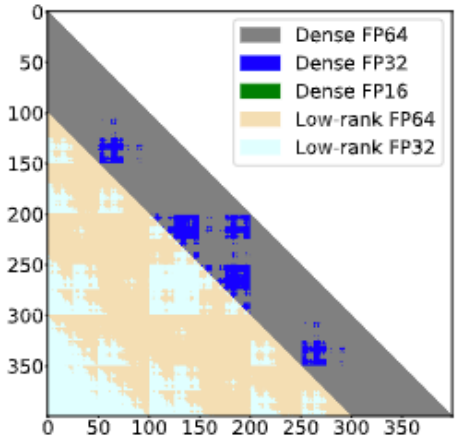


memory footprint
0.9 TB

strong correlation



memory footprint
3.8 TB

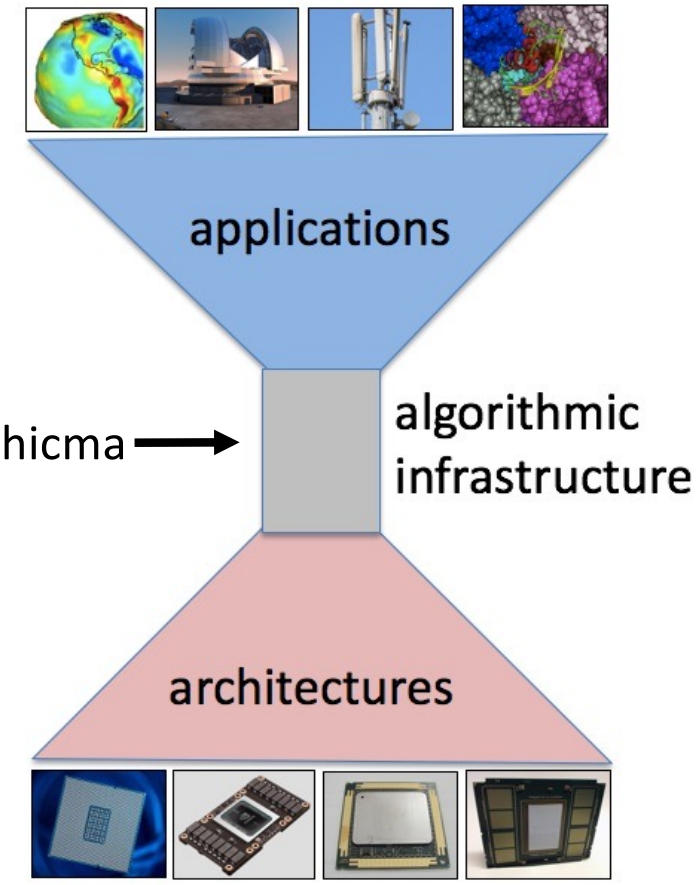


memory footprint
1.8 TB

default dense double is ~4 TB

Hourglass model of software

https://github.com/ecrc/hicma



Conclusions recapped

In a world of environmental and financial constraints, in which computational infrastructure demands a growing sector of lab budgets and global energy expenditure, HPC must address the need for *greater efficiency*.

HPC has *excelled* at this historically in

- hardware
- algorithms
- redefining actual outputs of interest in applications

There are *new algorithmic* opportunities in

- reduced rank representations
- reduced precision representations

For follow-up

- *Parallel Approximation of the Maximum Likelihood Estimation for the Prediction of Large-Scale Geostatistics Simulations,* S. Abdulah, H. Ltaief, Y. Sun, M. G. Genton & D. Keyes, 2018 IEEE International Conference on Cluster Computing (CLUSTER), 2018, pp. 98-108, doi: 10.1109/CLUSTER.2018.00089.
- *Hierarchical Algorithms on Hierarchical Architectures*, D. Keyes, H. Ltaief & G. Turkiyyah, 2020, Philosophical Transactions of the Royal Society, Series A 378:20190055, doi 10.1098/rsta.2019.0055
- *Responsibly Reckless Matrix Algorithms for HPC Scientific Applications*, H. Ltaief, M. G. Genton, D. Gratadour, D. Keyes & M. Ravasi, 2022, Computing in Science and Engineering, doi 10.1109/MCSE.2022.3215477
- *Reshaping Geostatistical Modeling and Prediction for Extreme-Scale Environmental Applications*, Q. Cao, S. Abdulah, R. Alomairy, Y. Pei, P. Nag, G. Bosilca, J. Dongarra, M. G. Genton, D. E. Keyes, H. Ltaief & Y. Sun, 2022, in proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'22), IEEE Computer Society (ACM Gordon Bell Finalist, to appear).
- *Mixed Precision Algorithms in Numerical Linear Algebra*, 2022, N. J. Higham & T. Mary, Acta Numerica, pp. 347—414, doi:10.1017/S0962492922000022.

Thank you, Collaborators!

KAUST Supercomputing Core Lab, HLRS-Stuttgart, Oak Ridge LCF, RIKEN, and:


Qinglei Cao


Yu Pei


George Boslica


Jack Dongarra


Rabab Alomairy


Pratik Nag


Sameh Abdulah


Hatem Ltaief


Ying Sun


Marc Genton


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

Want to contribute to computationally efficient infrastructure?

- Contributions are required up and down the software tool chain of many applications.
- The HiCMA group in the Extreme Computing Research Center at KAUST periodically has post-doc openings.
- Please enquire of Principal Research Scientist Hatem Ltaief, if interested, at

hatem.ltaief@kaust.edu.sa