Efficient Computation
Through Tuned Approximation

ATPESC
1 August 2023

David Keyes
and the HiCMA group of KAUST’s
Extreme Computing Research Center
ATPESC is the best!

You could ask *me* why, but just ask:
Leighton Wilson, ATPESC 2019
University of Michigan, PhD 2021

Now at Cerebras, *which he first heard about at ATPESC 2019*, from Rob Schreiber

Now a co-author of mine on a Gordon Bell Finalist paper, 2023
What is “extreme” computing?

“Extreme” can mean:

• extreme in scale (as in number of nodes or cores)
• extreme in low memory bandwidth per core (as in CPUs)
• extreme in low memory capacity per core (as in GPUs)
• extreme in low power constraints (as in remote “edge” devices, like telescopes)
• extreme in real-time constraints (as in data-streaming apps, like particle colliders)
• extreme in long running times (as in low-scaling apps, like some density functional theory codes)
Why an extreme computing research center?

For “extreme” applications, from
• simulation, data analytics, and machine learning

The ECRC
• develops algorithms – with theoretical backing where possible
• develops, deploys, and supports efficient portable open-source software implementations
• develops the next generation exascale workforce
• aids scientists & engineers needing to enter extreme regimes
• collaborates with vendors who commercialize some of the software
• tracks emerging architectures (e.g., WSEs, FPGAs, quantum)
Some home-grown software targeting extremes

Updated annually for SC’xy, at https://github.com/ecrc
Externally hosted software, too

Two PETSc developers with extensive line commits are in the ECRC:

Lisandro Dalcin
PETSc, petsc4py, mpi4py, mpi4py_fft, shem4py, ...

Stefano Zampini
PETSc, OpenFOAM, deal.ii, MFEM, CEED, ...
I’ve been well set up by previous speakers

Tom (OLCF): One cabinet of Frontier (2022) has 10% more compute power than Titan (2009) for 22X less electrical power

Giri (NVIDIA): Given a problem size and required accuracy, what is the lowest total cost of ownership to get there?

Murali (ANL): Dataflow eliminates memory traffic and overhead

Kelly (NextSilicon): The next revolution in hardware is software

Mike (ANL): We no longer drive the vendors

Tim (Intel): Data movement dominates!!!

Thomas (ANL): You are young and not yet totally jaded, so I share my dream with you!
I’ve been well set up by previous speakers

Tom (OLCF): One cabinet of Frontier (2022) has 10% more compute power than Titan (2009) for 22X less electrical power

Giri (NVIDIA): Given a problem size and required accuracy, what is the lowest total cost of ownership to get there?

Murali (ANL): Dataflow eliminates memory traffic and overhead

Kelly (NextSilicon): The next revolution in hardware is software

Mike (ANL): We no longer drive the vendors

Tim (Intel): Data movement dominates!!!

Thomas (ANL): You are young and not yet totally jaded, so I share my dream with you!
Conclusions, up front

As computational infrastructure demands a growing sector of research budgets and global energy expenditure, we must *all* address the need for greater efficiency.

As a community, we have excelled at this historically in three aspects:
- architectures
- applications (redefining *actual outputs of interest*)
- algorithms

There are *new algorithmic* opportunities in:
- reduced rank representations
- reduced precision representations
Our journey in tuned approximation began in 2018 with these time traces…

... for factorization of a dense 54K covariance matrix on four 32-core nodes of a Cray XC-40

Dense
Tile-based
Cholesky factorization
(Chameleon)

Tile low rank
(TLR)
Cholesky factorization
(HiCMA)

- TLR scores a lower percentage of peak (after squeezing out flops)
- TLR has poorer load balance (a higher percentage of idle time (red) vs. computation (green))
- TLR scales less efficiently (less able to cover data motion with computation)
- TLR is, however, **10X superior in time** for required application accuracy, at **about 65% of average power** compared to dense

Akbudak, Ltaief, Mikhalev, Charara & K., *Exploiting Data Sparsity for Large-scale Matrix Computations*, Euro-Par 2018
Computational efficiency through *tuned approximation*: a 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
Efficiency (“science per Joule”) improvement in HPC?

- We consider 3 categories of efficiency improvement
  - from architectures, applications, algorithms
- In 2022 & 2023 Gordon Bell finalist papers
- Efficiency improvements in kernel linear algebra operations from exploiting
  - rank structure (related to correlation smoothness)
  - precision structure (related to correlation magnitudes)
Time-to-solution addresses the energy “elephant”

Frontier (#1 on Top500) delivers about 1 Exaflop/s at about 50 Gigaflop/s per Watt

- **20 MegaWatts consumed continuously**

Representative electricity cost in US is $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

- 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:
- saves $4M/year
- takes 2,000 cars off the road

A 10X improvement:
- saves $36M/year
- takes 18,000 cars off the road

10X or more is achievable in many use cases
CO2 production equivalents

“Science per Joule” is a matter of planetary stewardship
Running on Frontier versus flying commercially

- Carbon footprint of a KiloWatt-hour is 0.5 kg CO2-equivalent
  - 10,000 kg CO2e hourly carbon footprint for an exaflop/s system (10 metric tons)
- Carbon footprint of one passenger-hour of commercial cruise Mach flight is about 0.25 metric tons CO2e
  - 1 hour of exaflop/s is roughly equivalent to 40 passenger-hours of flight

These 39 passengers cost about the same per hour as Frontier

These 3 passengers can fly from Jeddah to Chicago at the cost of one hour of Frontier

Justify your flight to ATPESC by efficient programming!

Better yet, please justify mine 😊
Architecture efficiency tracked by the Green 500

https://en.wikipedia.org/wiki/Green500

Gigaflop/s per Watt for #1 on the Green 500

> 15X in ten years
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”

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

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

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

*Six months is reduced to 1 second (recall: $3.154 \times 10^7$ 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*

\[ \nabla^2 u = f \]

\[ N = n^3 = (1/h)^3 \]


36 years means 24 doublings = 16 million-fold
“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

“Algorithmic Moore’s Law” for combustion simulations

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 (not exclusively) 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): $N^3 \rightarrow k^2 N (\log N)^2$
  - MLMC (2000’s): $N^{3/2} \rightarrow N (\log N)^2$
  - Randomized matrix algorithms (2010’s): $N^3 \rightarrow N^2 \log k$
  - ??? (2020’s): ??? \rightarrow ???
Hints for contributions for the 2020’s

*You* are going to replace woefully inefficient first-order convergent neural network training methods by, e.g.,

- communication-reduced hierarchically preconditioned second-order methods
- nonlinear matrix-free acceleration methods

*You* are going master hybridized mod-sim/ML workflows

- use few instances of high fidelity, high resolution simulations supplemented by many instances of machine-learned surrogates

“With great computational power comes great algorithmic responsibility.”

– Longfei Gao, ALCF (PhD 2013, KAUST)
Improving the “science per Joule” (or per unit time) involves:

- architecture
- application
- algorithm/software

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 we need to understand in our pursuit of computational efficiency in linear algebra:
• conditioning (implications on precision)
• rank structure (implications on sparsification)
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:

\[
\begin{pmatrix}
2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 \\
\end{pmatrix}
\]
Let \( n = 1/h \) and consider Dirichlet end conditions with \( n-1 \) interior points. Then:

\[
\begin{align*}
\lambda_1 &= 2 \left[ 1 - \cos \frac{\pi}{n} \right] \sim \left( \frac{\pi}{n} \right)^2 \\
\lambda_{n-1} &= 2 \left[ 1 - \cos \left( \frac{n-1}{n} \pi \right) \right] \sim 4
\end{align*}
\]

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):

\[
\kappa = \frac{\lambda_{n-1}}{\lambda_1} \sim \left( \frac{4}{\pi^2} \right) n^2
\]

In single precision real arithmetic, \( \kappa \) 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.
A is full-rank, but its off-diagonal blocks have low rank

It's inverse is dense, but it inherits the same rank structure

$$A^{-1} = \frac{1}{8} \times$$

\[
\begin{bmatrix}
2 & -1 & & & & & \\
-1 & 2 & -1 & & & & \\
& -1 & 2 & & & & \\
& & -1 & 2 & -1 & & \\
& & & -1 & 2 & -1 & \\
& & & & -1 & 2 & \\
\end{bmatrix}
\]

\[
= \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

\[
= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}
\]

\[
= \begin{bmatrix}
4 & 3 & 2 & 1 \\
8 & 6 & 4 & 2 \\
12 & 9 & 6 & 3 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}
\]

\[
= \begin{bmatrix} 4 & 3 & 2 & 1 \end{bmatrix}
\]
It turns out that many formally dense matrices arising from

- **integral equations** with smooth Green’s functions
- **covariances** in statistics
- **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 (if well ordered)
It turns out that many matrices arising in applications have blocks of \textit{relatively small norm} and can be replaced with \textit{reduced precision}.

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, \textit{Mixed precision algorithms in numerical linear algebra}, Acta Numerica (2022).
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 working sets 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 LDLᵀ**
  - Operations $O(N³)$
  - Storage $O(N²)$

- **Tile low-rank** (Amestoy, Buttari, L’Excellent & Mary, *SISC*, 2016)
  - Operations $O(k^{0.5}N²)$
  - 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 $O(k^2N \log^2 N)$
  - Storage $O(kN)$
  - for strong admissibility, where $k$ is max rank of any compressed block
• 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)
- power constraints (e.g., remote telescopes)
- real-time constraints (e.g., wireless communication)
- running time constraints (e.g., chemistry, materials, genome-wide associations)
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 $\Sigma$

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

- Apply inverse $\Sigma^{-1}$ and determinant $|\Sigma|$ 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, $a_{ij} \sim \exp \left(-c|x_i - x_j|^2\right)$
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

- Red arrows: speedups from hardware, same algorithm
- Green arrows: speedups from algorithm, same hardware
- Blue arrow: from both

Akbudak, Ltaief, Mikhailov, 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

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

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
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, Mikhailov, Bosilca, Ltaief, K. & Dongarra, Extreme-Scale Task-Based Cholesky Factorization Toward Climate and Weather Prediction Applications. PASC’20 (ACM)
Two motivations for mixed precision

• Mathematical: (much) better than “no 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

<table>
<thead>
<tr>
<th>Peak Performance in TF/s</th>
<th>V100 NVLink</th>
<th>A100 NVLink</th>
<th>H100 SXM</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP64</td>
<td>7.5</td>
<td>9.7</td>
<td>34</td>
</tr>
<tr>
<td>FP32</td>
<td>19.5</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>FP64 Tensor Core</td>
<td>15</td>
<td>19.5</td>
<td>67</td>
</tr>
<tr>
<td>FP32 Tensor Core</td>
<td>8x</td>
<td>156</td>
<td>495</td>
</tr>
<tr>
<td>FP16 Tensor Core</td>
<td>120</td>
<td>312</td>
<td>989</td>
</tr>
</tbody>
</table>

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)

Reshaping Geostatistical Modeling and Prediction for Extreme-Scale Environmental Applications

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

II. PERFORMANCE ATTRIBUTES

<table>
<thead>
<tr>
<th>Performance Attributes</th>
<th>Our submission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Size</td>
<td>Nine million geospatial locations\textsuperscript{1}</td>
</tr>
<tr>
<td>Category of achievement</td>
<td>Time-to-solution and scalability</td>
</tr>
<tr>
<td>Type of method used</td>
<td>Maximum Likelihood Estimation (MLE)</td>
</tr>
<tr>
<td>Results reported on basis of</td>
<td>Whole application</td>
</tr>
<tr>
<td>Precision reported</td>
<td>Double, single, and half precision</td>
</tr>
<tr>
<td>System scale</td>
<td>16K Fujitsu A64FX nodes of Fugaku\textsuperscript{1}</td>
</tr>
<tr>
<td>Measurement mechanism</td>
<td>Timers; FLOPS; Performance modeling</td>
</tr>
</tbody>
</table>
GB’22 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
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 plots]
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

- 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 $p = 1$)
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
ExaGeoStat is a high performance software package for computational geostatistics on many-core systems. The Maximum Likelihood Estimation (MLE) method is used to optimize the likelihood function for a given spatial set. MLE provides an efficient way to predict missing observations in the context of climate/weather forecasting applications. ExaGeoStat further exploits the data sparsity of the covariance matrix to address the curse of dimensionality. In particular, ExaGeoStat supports the factorized (LU) approximation and mixed-precision computations to model anisotropic, multivariate, space and space-time problems. This translates into a reduction of the memory footprints and the algorithmic complexity of the MLE operation, while still maintaining the overall fidelity of the underlying model.
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 V100
- AMD EPYC
- Intel X86
- #2 Fugaku
- #30 Hawk
- #5 Summit
- #104 Shaheen-2
The log-likelihood function: \( \ell(\theta) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma(\theta)| - \frac{1}{2} Z^\top \Sigma(\theta)^{-1} Z. \)

- Optimization over \( \theta \) to maximize the likelihood function estimation until convergence
  - generate the covariance matrix \( \Sigma(\theta) \) using a specified kernel
  - evaluate the log determinant and the inverse operations, which require a Cholesky factorization of the given covariance matrix
  - update \( \theta \)

- 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

\[ C(r; \theta) = \frac{\theta_1}{2^{\theta_3-1} \Gamma(\theta_3)} \left( \frac{r}{\theta_2} \right)^{\theta_3} K_{\theta_3} \left( \frac{r}{\theta_2} \right) \]

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

Space/Time Nonseparable Kernel

\[ C(h, u) = \frac{\sigma^2}{a |u|^{2\alpha} + 1} M_{\nu} \left( \frac{||h||/a_s}{(a |u|^{2\alpha} + 1)^{\beta/2}} \right) \]

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

Multivariate Parsimonious Kernel

\[ C_{ij}(||h||; \theta) = \frac{\rho_{ij} \sigma_i \sigma_j}{2^{\nu_{ij}} \Gamma(\nu_{ij})} \left( \frac{||h||}{a} \right)^{\nu_{ij}} K_{\nu_{ij}} \left( \frac{||h||}{a} \right) \]

Multivariate Flexible Kernel

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

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

\[ \rho_Z(h) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left( \frac{4 \sqrt{2} \gamma}{\phi} \right)^{\nu} K_{\nu} \left( 4 \sqrt{2} \frac{\gamma}{\phi} \right) \]

Powered Exponential Kernel

\[ C(r; \theta) = \theta_0 \exp \left( \frac{r^{\theta_2}}{\theta_1} \right) \]
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) $u_{\text{high}}$ and $u_{\text{low}}$, 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

$$\frac{\| A_{ij} \|_F}{\left(\frac{\| A \|_F}{n_T}\right)} < \frac{u_{\text{high}}}{u_{\text{low}}}$$

is stored in low precision; otherwise kept in high
• The mixed precision tiled matrix $\mathcal{A}$ thus formed satisfies

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

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

Accuracy on synthetic 2D space dataset

MLE parameters

- Variance parameter $\theta_0$
- Range parameter $\theta_1$
- Smoothness parameter $\theta_2$

Degree of correlation:

- Weak
- Medium
- Strong

Diagnostic scores for different distributions:

- Left-Skewed
  - $Q_1$, $Q_2$, $Q_3$
- Symmetric
  - $Q_1$, $Q_2$, $Q_3$
- Right-Skewed
  - $Q_1$, $Q_2$, $Q_3$
### Accuracy on real 3D (2D space + time) dataset

<table>
<thead>
<tr>
<th>Variants</th>
<th>Variance ($\theta_0$)</th>
<th>Range ($\theta_1$)</th>
<th>Smoothness ($\theta_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense FP64</td>
<td>1.0087</td>
<td>3.7904</td>
<td>0.3164</td>
</tr>
<tr>
<td>MP+dense</td>
<td>0.9428</td>
<td>3.8795</td>
<td>0.3072</td>
</tr>
<tr>
<td>MP+dense/TLR</td>
<td>0.9247</td>
<td>3.7756</td>
<td>0.3068</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variants</th>
<th>Range-time ($\theta_3$)</th>
<th>Smoothness-time ($\theta_4$)</th>
<th>Nonsep-param ($\theta_5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense FP64</td>
<td>0.0101</td>
<td>3.4890</td>
<td>0.1844</td>
</tr>
<tr>
<td>MP+dense</td>
<td>0.0102</td>
<td>3.4941</td>
<td>0.1860</td>
</tr>
<tr>
<td>MP+dense/TLR</td>
<td>0.0102</td>
<td>3.5858</td>
<td>0.1857</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variants</th>
<th>Log-Likelihood (llh)</th>
<th>MSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense FP64</td>
<td>-136675.1</td>
<td>0.9345</td>
</tr>
<tr>
<td>MP+dense</td>
<td>-136529.0</td>
<td>0.9348</td>
</tr>
<tr>
<td>MP+dense/TLR</td>
<td>-136541.8</td>
<td>0.9428</td>
</tr>
</tbody>
</table>
Performance on up to 16K nodes of Fugaku

- ~3x less time for same size
- ~3x greater size for same time

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

strong correlation

memory footprint
1.6 TB
memory footprint
0.9 TB
memory footprint
3.8 TB
memory footprint
1.8 TB

default dense double is ~4 TB
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
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
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 Ostrououchov, ORNL
Scaling the “Memory Wall” for Multi-Dimensional Seismic Processing with Algebraic Compression on Cerebras CS-2 Systems

Hatem Ltaief\textsuperscript{1,2}, Yuxi Hong\textsuperscript{1,2}, Leighton Wilson\textsuperscript{3,4}, Mathias Jacquelin\textsuperscript{3,4}, Matteo Ravasi\textsuperscript{1,2}, and David Keyes\textsuperscript{1,2}

\textsuperscript{1}Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal, KSA
\textsuperscript{2}\{Firstname.Lastname\}@kaust.edu.sa
\textsuperscript{3}Cerebras Systems Inc., Sunnyvale, California, USA
\textsuperscript{4}\{Firstname.Lastname\}@cerebras.net
GB’23 collaborators

Group42 (Abu Dhabi), KAUST Supercomputing Core Lab and:

Leighton Wilson
Mathias Jacquelin
Yuxi Hong
Hatem Ltaief
Matteo Ravasi
Cerebras CS-2 Wafer-Scale Engine (WSE)
I. JUSTIFICATION FOR THE GORDON BELL PRIZE

High-performance matrix-vector multiplication using low-rank approximation. Memory layout optimizations and batched executions on massively parallel Cerebras CS-2 systems. Leveraging AI-customized hardware capabilities for seismic applications for a low-carbon future. Application-worthy accuracy (FP32) with a sustained bandwidth of 92.58PB/s (for 48 CS-2s) would constitute the third-highest throughput from June’23 Top500.
<table>
<thead>
<tr>
<th>Performance Attributes</th>
<th>Our submission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Size</td>
<td>Broadband 3D seismic dataset</td>
</tr>
<tr>
<td></td>
<td>(≈ 20k sources and receivers and frequencies up to 50Hz)</td>
</tr>
<tr>
<td>Category of achievement</td>
<td>Sustained bandwidth</td>
</tr>
<tr>
<td></td>
<td>Scalability</td>
</tr>
<tr>
<td>Type of method used</td>
<td>Algebraic compression</td>
</tr>
<tr>
<td>Results reported on basis of</td>
<td>Whole application (for GPU cluster)</td>
</tr>
<tr>
<td></td>
<td>Main kernel (for Cerebras cluster)</td>
</tr>
<tr>
<td>Precision reported</td>
<td>Single precision complex</td>
</tr>
<tr>
<td>System scale</td>
<td>Up to 48 Cerebras CS-2 systems, i.e.,</td>
</tr>
<tr>
<td></td>
<td>35,784,000 processing elements¹</td>
</tr>
<tr>
<td>Measurement mechanism</td>
<td>Timers; Memory accesses; Performance modeling</td>
</tr>
</tbody>
</table>
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.
Roofline models of a 6-shard configuration compared with other solutions

Roofline models of a 48-shard configuration compared with current top 5
Conventional algorithms for MDD would not have mapped onto the Cerebras CS-2 engines because their $N^3$ arithmetic complexity is prohibitive. Only the algebraically compressed form of the problem fits. All parts of this interdisciplinary project are thus necessary for its success.

As the title indicates, this team is ‘scaling the memory wall’ that has loomed over computational science & engineering at the high end for, by now, three decades. Their algorithms and CS-2 implementation have enormous implications for our community, since their application is representative of many important CS&E problems.

– Professor Omar Ghattas, U Texas
For the past 3 decades, we have needed large-scale convolutions for multiple applications to tackle subsurface challenges – which are now greater than ever for the energy transition, such as rapid, wide-scale monitoring of subsurface hydrogen storage – but have never achieved it due to the unsurmountable bottleneck imposed by the size of datasets (starting at TBs).

This project, with its balanced focus on accuracy and practical performance, is likely to finally break through a decades-old barrier in geophysical imaging.

– Dr. Ivan Vasconcelos, Shearwater Geoservices
The impact that the efficient implementation of multi-dimensional convolution with low-rank tiles that Ltaief and co-authors have developed is better understood if we bear in mind that multidimensional convolution and deconvolution are ubiquitous operations in seismic processing.

This new implementation may lead to a drastic reduction of the turnaround time of seismic data processing projects. The consequence is that the decision-makers, regardless of whether they use seismic images for conventional hydrocarbon exploration or for other applications, will receive valuable information in a timely manner.

– Dr. Claudio Bagaini, SLB (Schlumberger)
History does not repeat itself, but it often rhymes.
– Samuel Clemens (Mark Twain)

- In 1989, the Gordon Bell Prize went to a seismic application run on the CM-2, a system designed for AI.

- In 2023, will the Gordon Bell Prize go to a seismic application run on the CS-2, a system designed for AI?
Hourglass model of software

https://github.com/ecrc/hicma
As computational infrastructure demands a growing sector of research budgets and global energy expenditure, we must all address the need for greater efficiency.

As a community, we have excelled at this historically in three aspects:

- architectures
- applications (redefining *actual outputs of interest*)
- algorithms

There are *new algorithmic* opportunities in:

- reduced rank representations
- reduced precision representations
Sustainable computing – two meanings

Computing sustainably
• or at least efficiently – not computing more than necessary for a given scientific target

Computing to *support* sustainability
• renewable energy
• affordable energy
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; see: https://cemse.kaust.edu.sa/hicma/join-hicma
- Please enquire if interested at ecrc.opportunities@kaust.edu.sa
CS&E (and HPC) in KAUST’s DNA

CSE 2009: The World’s First CSE University
June 15, 2009

The King Abdullah University of Science and Technology, scheduled to welcome its first class of students in September, sponsored a reception in Miami on March 2, the first day of the SIAM Conference on Computational Science and Engineering. David Keyes and Omar Ghattas, involved in different ways in the new venture, hosted the reception and made informal presentations to the assembled crowd.

Most readers will know something of KAUST, which for the record is a graduate-only (master’s and doctoral) university being constructed in Saudi Arabia, on the eastern edge of the Red Sea, not far from Jeddah. Keyes, the inaugural chair of KAUST’s Mathematical and Computer Sciences and Engineering Division, offered examples of research areas of particular interest to Saudi Arabia and the region that will be emphasized; among them are geophysics, seismology, reservoir modeling, CO2 sequestration, photovoltaics, stress-tolerant agriculture, desalination, catalysis, and materials, along with the applied mathematics and computer science required to support them.

Sizeable recruitment ads for KAUST have appeared in many recent issues of SIAM News, often side by side with ads placed by partners of the new university, such as the KAUST–UT Austin Academic Excellence Alliance. Ghattas, as director of the alliance, has been recruiting faculty for KAUST’s Earth and Environmental Sciences and Engineering Division. The week of the SIAM conference, the NA Digest ran a recruitment notice for numerical analysts, posted by Nick Trefethen on behalf of the KAUST-funded Oxford Centre for Collaborative Applied Mathematics.

Other research alliances and partnerships are in place. Stanford, for example, is recruiting faculty in applied math and computer science, as well as providing guidance in curriculum development; the initial KAUST curriculum in those disciplines is similar to Stanford’s, Keyes said in Miami. Cornell is a...
Shaheen-1 and Shaheen-2 ranks over time

#14 globally (2009)

#7 globally (2015)

222 TF/s ~25X 5.53 PF/s

Shaheen I Shaheen II

Shaheen-3’s Grace-Hopper chips will come in 1Q ‘24

<table>
<thead>
<tr>
<th>#</th>
<th>Site</th>
<th>Manufacturer</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>Rmax  (Pflop)</th>
<th>Power (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oak Ridge National Laboratory</td>
<td>HPE</td>
<td>Frontier HPE Cray EX235a, AMD EPYC 64C 2.0GHz, Instinct MI300 AI, Slingshot-11</td>
<td>USA</td>
<td>8,699,904</td>
<td>1,194</td>
<td>22.7</td>
</tr>
<tr>
<td>2</td>
<td>RIKEN Center for Computational Science</td>
<td>Fujitsu</td>
<td>Fugaku Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D</td>
<td>Japan</td>
<td>7,630,848</td>
<td>442.0</td>
<td>29.9</td>
</tr>
<tr>
<td>3</td>
<td>EuroHPC / CSC</td>
<td>HPE</td>
<td>LUMI HPE Cray EX235a, AMD EPYC 24C 2.0GHz, Instinct MI300 AI, Slingshot-11</td>
<td>Finland</td>
<td>2,069,760</td>
<td>309.1</td>
<td>6.0</td>
</tr>
<tr>
<td>4</td>
<td>EuroHPC / CINECA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Oak Ridge National Laboratory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With 4608 AMD “Genoa” CPUs & 2800 NVIDIA “Hopper” GPUs (in 700 “Grace-Hopper” ARM-NVIDIA CPU-GPU nodes), KAUST’s Shaheen-3 will pack approximately 25 + 100 Pflop/s – would be #6 on the Top 500 list if on the floor today

E. Strohmaier, Top500, ISC’23
Why? 114 KAUST faculty supercompute

- Aamir Farooq
- Ajay Jasra
- Alexandre Rosado
- Andrea Fratalocchi
- Arnab Pain
- Athanasios Tzavaras
- Atif Shamim
- Basem Shihada
- Bernard Ghanem
- Boon Ooi
- Brande Wulff
- Burton Jones
- Carlos Duarte
- Charlotte Hauser
- Cristian Picoreanu
- Daniel Peter
- Daniele Boffi
- David Ketcheson
- David Keyes
- Deanna Lacoste
- Dominik Michels
- Enzo Di Fabrizio
- Eric Feron
- Francesca Benzoni
- Frederic Laquai
- Gabriel Wittum
- Geert Jan Witkamp
- Georgiy Stenchikov
- Haavard Rue
- Hakan Bagci
- Hernando Ombao
- Himanshu Mishra
- Hong Im
- Hossein Fariborzi
- Hussein Hoteit
- Hussam Alshareef
- Ibrahim Hoteit
- Ingo Pinnau
- Iman Roqan
- Ivan Viola
- Jean-Marie Basset
- Jerry Schuster
- Jesper Tegner
- Jinchao Xu
- Johannes Vrouwenvelder
- Jorge Gascon
- Jr-Hau He
- Kim Choon Ng
- Kuo-Wei Huang
- Lain-Jong Li
- Luigi Cavallo
- Magdy Mahfouz
- Magnus Rueping
- Mani Sarathy
- Marc Genton
- Marco Canini
- Mark Tester
- Markus Hadwiger
- Mario Lanza
- Martin Heeney
- Martin Mai
- Matteo Parsani
- Matteo Ravasi
- Matthew McCabe
- Meriem Taous Laleg
- Min Suk Cha
- Mohamed Eddaoudi
- Mohamed Elhoseiny
- Mohammad Younis
- Nikos Hadjichristidis
- Noreidine Ghaffour
- Omar Knio
- Omar Mohammed
- Panos Kalnis
- Pascal Saikaly
- Pedro Castano
- Peter Richtarik
- Peter Schmid
- Peter Wonka
- Pierre Magistretti
- Raphael Huser
- Raul Tempone
- Robert Hoehndorf
- Rod Wing
- Salim Al-Babili
- Samir Hamdan
- Shadi Fatayer
- Shehab Elsayed
- Shuyu Sun
- Sigurdur Thoroddsen
- Slim Alouini
- Stefaan Dewolf
- Stefan Arild
- Suk Chung
- Suzana Nuñes
- Tadd Truscott
- Tadeusz Patzek
- Takashi Gojobori
- Tareq AlNaffouri
- Tariq AlKhalifa
- Thomas Anthopoulos
- Udo Schwingenschloegl
- Valerio Orlando
- William Roberts
- Volker Vahrenkamp
- Xiangliang Zhang
- Xiaohang Li
- Xin Gao
- Xixiang Zhang
- Ying Sun
- Yu Han
- Yves Gnanou
- Zhiping Lai

61% of all faculty
big trends

The Arab World Prepares the Exascale Workforce

Figure 1. Fifteen “universals” of exascale computing.

- Architectural Imperatives
  - Reside “high” on the memory hierarchy, close to the processing elements
  - Rely on SIMD/SIMT-amenable batches of tasks at fine scale
  - Reduce synchrony in frequency and/or span
  - Reduce communication in number and/or volume of messages
  - Exploit heterogeneity in processing, memory, and networking elements

- Strategies in Practice
  - Exploit extra memory to reduce communication volume
  - Perform extra flops to require fewer global operations
  - Use high-order discretizations to manipulate fewer DOFs (with more ops per DOF)
  - Adapt floating point precision to output accuracy requirements
  - Take more resilience into algorithm space, out of hardware/systems space

- Strategies in Progress
  - Employ dynamic scheduling capabilities, for example, dynamic runtime systems based DAGs
  - Code to specialized “back-ends” while preserving high-level APIs to general users
  - Exploit data sparsity to meet “curse of dimensionality” with “blessing of low rank”
  - Process “on the fly” rather than storing all at once (especially large dense matrices)
  - Co-design algorithms with hardware, including computing in the network or in memory
For follow-up


Thank you

شاكرا

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology