# Large-Scale Spatial Data Science with ExaGeoStat

**Marc G. Genton**

In collaboration with S. Abdulah, Y. Sun, H. Ltaief, D. Keyes

Spatio-Temporal Statistics and Data Science (stsds.kaust.edu.sa)
Statistics Program (stat.kaust.edu.sa)
King Abdullah University of Science and Technology

August 5, 2022

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

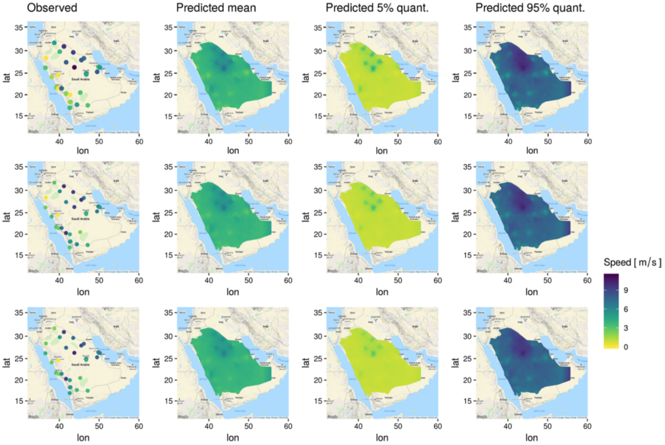
STATISTICS @KAUST

# Part I

## Spatial Statistics Overview

# Spatial Data Example

Wind speed (hourly) at 28 stations in Saudi Arabia in June 2010




Source: Lenzi, A., and Genton, M. G. (2020), Spatio-temporal probabilistic wind vector forecasting over Saudi Arabia, *Annals of Applied Statistics*, 14, 1359-1378.

# Matérn Covariance Function

We consider mean-zero Gaussian random fields with Matérn covariance

The popular parameterization of Matérn covariance function:

$$\mathrm{cov}\{Z(\boldsymbol{s}_i), Z(\boldsymbol{s}_j)\} = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\|\boldsymbol{s}_i - \boldsymbol{s}_j\|}{\beta} \right)^{\nu} K_{\nu} \left( \frac{\|\boldsymbol{s}_i - \boldsymbol{s}_j\|}{\beta} \right) + \tau^2 \mathbb{1}_{\{i=j\}}$$

where $K_\nu(\cdot)$ is the modified Bessel function of the second kind of order $\nu$, $\Gamma(\cdot)$ is the Gamma function, and $\mathbb{1}$ is the indicator function

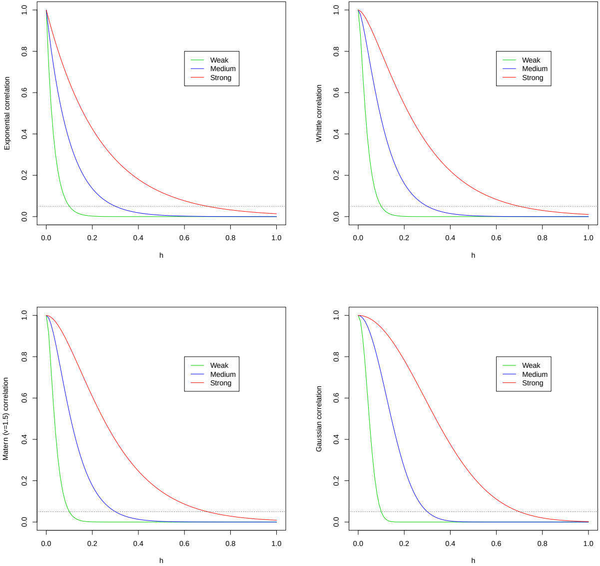
The four parameters determining the covariance structure are:
the partial sill $\sigma^2$, range $\beta > 0$, smoothness $\nu > 0$, and nugget $\tau^2$

# Effective Range

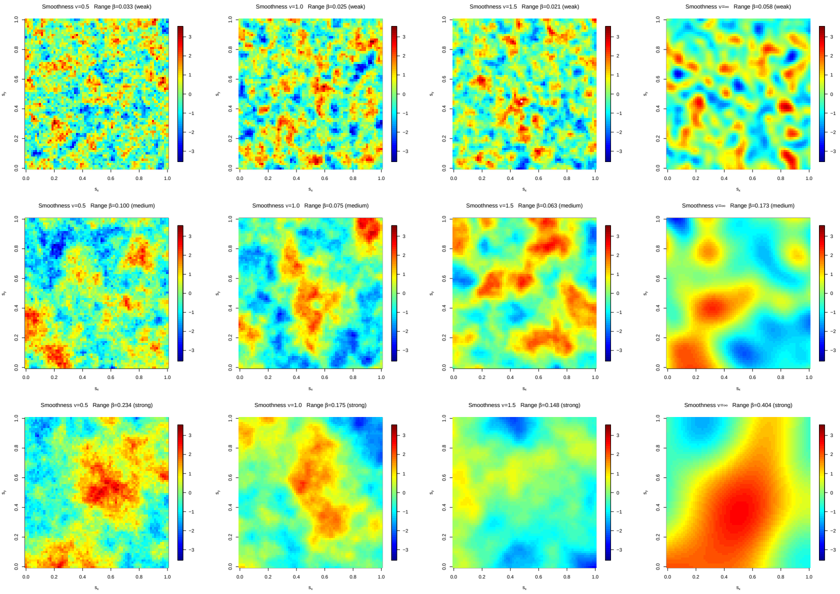
The effective range is the distance at which the correlation function reaches a small value, e.g. 5%

On the unit square, one can say the dependence is
weak / medium / strong
if the effective range is for example
0.1 / 0.3 / 0.7

# Plots of Matérn Covariance Functions



# Simulated Gaussian Random Fields using Matérn



# Covariance Parameter Estimation with Likelihoods

For simplicity, we focus on zero-mean stationary Gaussian random fields
The log-likelihood for $n$ locations:

$$\ell(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\boldsymbol{Z}^\top\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{Z}$$

where

$$\boldsymbol{Z} = \begin{pmatrix} Z(\boldsymbol{s}_1) \\ \vdots \\ Z(\boldsymbol{s}_n) \end{pmatrix}, \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{pmatrix} C(\boldsymbol{s}_1, \boldsymbol{s}_1; \boldsymbol{\theta}) & \dots & C(\boldsymbol{s}_1, \boldsymbol{s}_n; \boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ C(\boldsymbol{s}_n, \boldsymbol{s}_1; \boldsymbol{\theta}) & \dots & C(\boldsymbol{s}_n, \boldsymbol{s}_n; \boldsymbol{\theta}) \end{pmatrix}$$

- Log determinant and linear solver require a Cholesky factorization of the given covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$
- Cholesky factorization requires $O(n^3)$ floating point operations and $O(n^2)$ memory

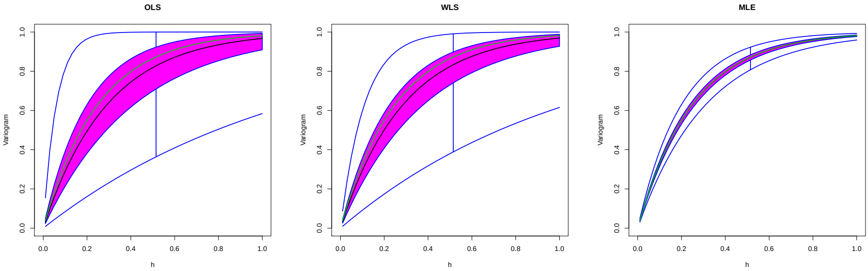
# Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimator

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \boldsymbol{Z}^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{Z}\}\end{aligned}$$

# Simulated Data Example

- Exponential variogram: $2\gamma(h) = 1 - \exp(-h/\theta)$ with $\theta = 0.25$
- Mean-zero GP generated at 400 random locations in unit square
- Estimate $\theta$ by OLS, WLS, MLE with 1000 replicates
- Functional boxplots (Sun and Genton, 2011)
- Note: no outlier detection
  (the factor is set to be large in order to see the variability better)
- More in Yan and Genton (2018)



# Prediction

For Gaussian random fields, kriging coincides with the conditional mean

$$\begin{pmatrix} \boldsymbol{Z} \\ Z(\boldsymbol{s}_0) \end{pmatrix} \sim N_{n+1}\left(\boldsymbol{0}, \begin{pmatrix} \boldsymbol{\Sigma}(\boldsymbol{\theta}) & \boldsymbol{k}(\boldsymbol{\theta}) \\ \boldsymbol{k}(\boldsymbol{\theta})^\top & C(\boldsymbol{s}_0, \boldsymbol{s}_0; \boldsymbol{\theta}) \end{pmatrix}\right)$$

where $\boldsymbol{k}(\boldsymbol{\theta}) = \left(C(\boldsymbol{s}_1, \boldsymbol{s}_0; \boldsymbol{\theta}), \ldots, C(\boldsymbol{s}_n, \boldsymbol{s}_0; \boldsymbol{\theta})\right)^\top$
The conditional distribution is

$$Z(\boldsymbol{s}_0)|\boldsymbol{Z} \sim N\Big(\boldsymbol{k}(\boldsymbol{\theta})^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{Z}, C(\boldsymbol{s}_0, \boldsymbol{s}_0; \boldsymbol{\theta}) - \boldsymbol{k}(\boldsymbol{\theta})^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{k}(\boldsymbol{\theta})\Big)$$

- Solution of system of linear equation $\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{Z}$ also needs a Cholesky factorization of $\boldsymbol{\Sigma}(\boldsymbol{\theta})$

# When Size of Dataset Becomes Large

- $O(n^3)$ floating point operations and $O(n^2)$ memory requirements for exact computations
  - High-Performance Computing is needed when $n$ is large
  - *ExaGeoStat* software

- Various approximation methods have been proposed to ease the computation and memory burden

- **2021 KAUST Competition on Spatial Statistics for Large Datasets** investigates the performance of different approximation methods with large synthetic data generated by *ExaGeoStat* (`https://cemse.kaust.edu.sa/stsds/2021-kaust-competition-spatial-statistics-large-datasets`)

  Huang, H., Abdulah, S., Sun, Y., Ltaief, H., Keyes, D. E., and Genton, M. G. (2021), "Competition on spatial statistics for large datasets (with discussion)," *Journal of Agricultural, Biological, and Environmental Statistics*, **26**, 580-595.

# 2021 KAUST Competition on Spatial Statistics for Large Datasets

- Launched November 23, 2020; Ended February 1, 2021
- 29 research teams worldwide registered and 21 teams successfully submitted their results
- Competition consists of four parts

| | Task | Data model | Data size |
|---|---|---|---|
| 1a | GP estimation | GP | 90,000 |
| 1b | prediction | GP | predict 10,000 conditional on 90,000 |
| 2a | prediction | Tukey $g$-and-$h$ | predict 10,000 conditional on 90,000 |
| 2b | prediction | GP & Tukey $g$-and-$h$ | predict 100,000 conditional on 900,000 |

- Metric for GP estimation: Mean Loss Efficiency (MLOE) and Mean Misspecification of the Mean Square Error (MMOM)
- Metric for prediction: RMSE

# 2021 KAUST Competition on Spatial Statistics for Large Datasets

Top winners:

- Sub-competition 1a:
  1 SpatStat-Fans: smoothed full-scale approximation
  2 GpGp: Vecchia's approximation
  3 RESSTE(CL/krig): composite likelihoods
- Sub-competition 1b:
  1 RESSTE(CL/krig): plug-in kriging (composite likelihood estimates)
  2 HCHISS: plug-in kriging (Vecchia's approximation estimates)
  3 Chile-Team: plug-in kriging (conditional pairwise likelihood estimates)
- Sub-competition 2a:
  1 RESSTE(Tukey-g-h-trans-GPGP): TGH + *GPGP* kriging
  3 GpGp(quick): basis functions for nonstat cov + *GPGP* kriging
  3 HMatrix: hierarchical matrix approximation
  3 RESSTE (nonpara-trans-GPGP): nonpara + *GPGP* kriging
- Sub-competition 2b:
  2 RESSTE(nonpara-trans-GPGP): TGH + *GPGP* kriging
  2 RESSTE(Tukey-g-h-trans-GPGP): nonpara + *GPGP* kriging
  2 Tohoku-University: covariance tapering

# 2021 KAUST Competition on Spatial Statistics for Large Datasets

| Sub-competition | Submission | Score | Rank |
|---|---|---|---|
| 1a | ExaGeoStat(estimated-model) | 154 | 0 |
| | SpatStat-Fans | 156 | 1 |
| | GpGp | 186 | 2 |
| | RESSTE(CL/krig) | 229 | 3 |
| 1b | ExaGeoStat(true-model) | 72 | 0 |
| | RESSTE(CL/krig) | 78 | 1 |
| | ExaGeoStat(estimated-model) | 79 | 1.5 |
| | HCHISS | 93 | 2 |
| | Chile-Team | 113 | 3 |

# 2021 KAUST Competition on Spatial Statistics for Large Datasets

| Sub-competition | Submission | Score | Rank |
|---|---|---|---|
| 2a | RESSTE(Tukey-g-h-GpGp) | 7 | 1 |
| | HMatrix | 8 | 3 |
| | RESSTE(nonparametric-GpGp) | 8 | 3 |
| | GpGp(quick) | 8 | 3 |
| 2b | Tohoku-University | 4 | 2 |
| | RESSTE(Tukey-g-h-GpGp) | 4 | 2 |
| | RESSTE(nonparametric-GpGp) | 4 | 2 |

# 2021 KAUST Competition on Spatial Statistics for Large Datasets

- Through the competition, we have better understood when each approximation method became inadequate
- The full datasets with one million spatial locations are publicly available at:
  https://doi.org/10.25781/KAUST-8VP2V
  which act as benchmarking data for future research
- The exact MLEs and lowest RMSEs achieved by researchers worldwide are released so that other methods can have an easy comparison
- 2022 competition included univariate nonstationary spatial data, space-time data, and bivariate spatial data; 20 teams of which 15 submitted results; analysis coming out soon!

# Part II

## High-Performance Computing (HPC)

Exascale Computing for Spatial Statistics

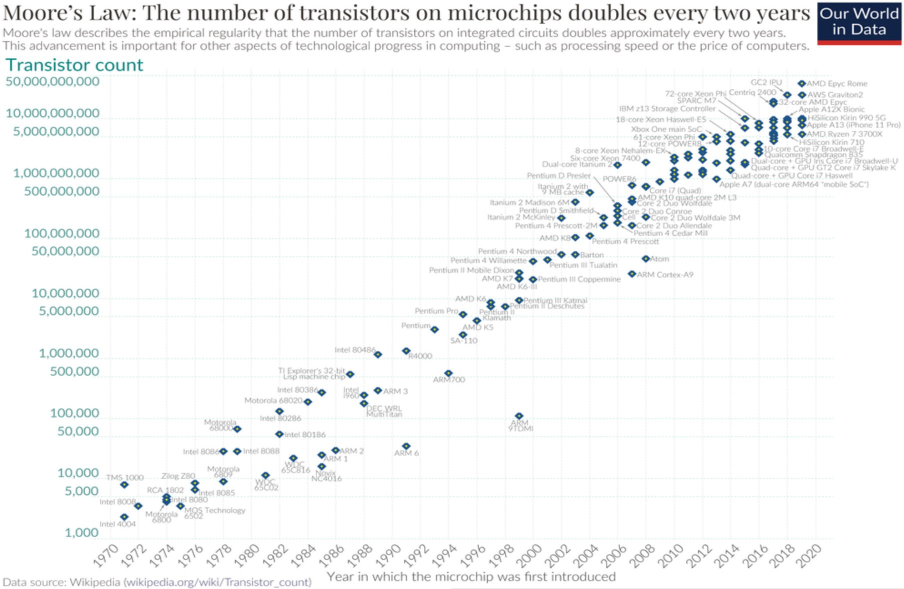
# High Performance Computing (HPC)

- Why HPC? Because of the flood of data in 2020:
  - The average internet user generates 1.5 GB of traffic per day (compared to 650MB in 2015)
  - Smart hospital generates over 3,000 GB per day
  - Self driving car generates over 4,000 GB per day
  - A connected plane generates over 40,000 GB per day
  - A connected factory generates over 1,000,000 GB per day
- Typical HPC workloads: Astrophysics, Bioinformatics, AI, Finance, Weather and Climate, and Cyber Security

# Parallel Computing

- A type of computation where many calculations or the execution of processes are carried out simultaneously
- Large problems can often be divided into smaller ones, which can then be solved at the same time
- Parallelism has long been employed in high-performance computing, but has gained broader interest due to the physical constraints preventing frequency scaling (i.e., end of Moore's law)
- As power consumption by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors

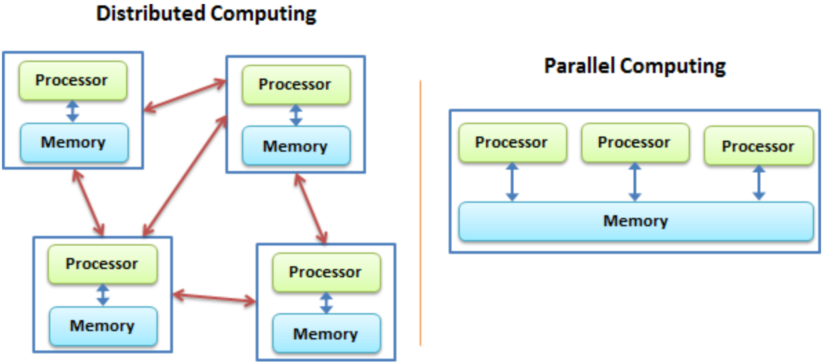
# Moore’s Law


Moore’s Law: The number of transistors on microchips doubles every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
GC2 IPU
AMD Epyc Rome
72-core Xeon Phi
Centriq 2400
AWS Graviton2
SPARC M7
32-core AMD Epyc
IBM z13 Storage Controller
Apple A12X Bionic
18-core Xeon Haswell-E5
HiSilicon Kirin 990 5G
Apple A13 (iPhone 11 Pro)
Xbox One main SoC
AMD Ryzen 7 3700X
61-core Xeon Phi
HiSilicon Kirin 710
12-core POWER8
10-core Core i7 Broadwell-E
8-core Xeon Nehalem-EX
Qualcomm Snapdragon 835
Six-core Xeon 7400
Dual-core + GPU Iris Core i7 Broadwell-U
Dual-core Itanium 2
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Apple A7 (dual-core ARM64 "mobile SoC")
Itanium 2 with 9 MB cache
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Atom
Pentium III Tualatin
Pentium II Mobile Dixon
AMD K7
Pentium III Coppermine
ARM Cortex-A9
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
WDC 65C816
ARM 1
ARM 6
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8008
Intel 8080
Motorola 6800
MOS Technology 6502
Intel 4004
1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018 2020
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
Year in which the microchip was first introduced

# Distributed Computing

- Multiple system processors can communicate with each other using messages that are sent over the network
- With a sufficiently fast network we can in principle extend this approach to millions of CPU-cores and beyond
- Benefits: Scalability, Reliability, and Performance
- Challenges: Complex architectural, construction, and debugging processes

# Parallel computing VS Distributed computing



# What is HPC?

- High-Performance Computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly
- It is an aggregation of computing powers to solve problems which are either **too large** for standard computers or take **too long**
- Depending on the HPC system, the compute nodes, even individually, might be much more powerful than a typical personal computer
- They often have multiple processors (each with many cores), and may have accelerators (such as Graphics Processing Units (GPUs))
- HPC does not mean only parallel processing, it mainly means high computing capabilities through a number of computing units

# What is HPC?

- HPC term applies to systems that function above a TFLOPS or $O(10^{12})$ floating-point operations per second (Flops/s)

| Name | Unit | Value |
|---|---|---|
| kiloFLOPS | kFLOPS | $10^3$ |
| megaFLOPS | MFLOPS | $10^6$ |
| gigaFLOPS | GFLOPS | $10^9$ |
| teraFLOPS | TFLOPS | $10^{12}$ |
| petaFLOPS | PFLOPS | $10^{15}$ |
| exaFLOPS | EFLOPS | $10^{18}$ |
| zettaFLOPS | ZFLOPS | $10^{21}$ |
| yottaFLOPS | YFLOPS | $10^{24}$ |

# The Japanese Fugaku: the Most Powerful Supercomputer in the World

- Fugaku Supercomputer (Japan, 1st ranked; now 2nd since June 2022) has around 537.2 PFlops/s theoretical peak performance for 7,630,848 cores (achieves 442 PFlops/s) – cost : $1 billion!
- You can think of Fugaku as putting 20 million smartphones in a single room, or equivalently 300,000 standard servers in a single room

https://my.matterport.com/show/?m=2TLoTcWigBf (Virtual tour)

## Other Supercomputers in the list https://www.top500.org (June 2022)

- 1st Frontier (US, ORNL) has around 1685 PFlops/s theoretical peak performance for 8,730,112 cores (achieves 1102 PFlops/s)
- 2nd Fugaku
- 4th Summit (US, ORNL) has around 200 PFlops/s theoretical peak performance for 2,414,592 cores (achieves 148 PFlops/s) – cost: $200 million!
- 11th JUWELS Booster Module (Germany – 70 PFlops/s)
- 12th HPC5 (Italy – 51 PFlops/s)
- 21st Marconi-100 (Italy – 29 PFlops/s)
- 23rd Piz Daint (Switzerland – 27 PFlops/s)
- 33rd PANGEA III (France – 25 PFlops/s)
- 60th Dragao (Brazil, Petroleo Brasileiro – 14 PFlops/s)
- 82nd MareNostrum (Spain – 10 PFlops/s)
- In Saudia Arabia:
    - 18th Dammam-7 (Aramco – 55 PFlops/s)
    - 97th Shaheen-2 (KAUST – 7 PFlops/s)

# Why HPC?

- Scientific simulation and modeling drive the need for greater computing power
- Single-core processors are not enough for the simulations needs
- Making processors with faster clock speeds is difficult due to the power/heat limitations
- It is also expensive to put huge memory on single processor
- Solution: parallel computing – divide up the the work among numerous linked system
- The use of HPC in modeling complex physical phenomena such as weather, fluid dynamics, molecular interactions, astronomy calculations and engineering design is well known to researchers in those fields

# Computational Statistics for Weather Prediction Applications

- Applications from climate and weather science often deal with a very large number of measurements regularly or irregularly located in a certain geographical region
- In geospatial statistics, these data are usually modeled as a realization from a Gaussian spatial random field
- This translates into evaluating the log-likelihood function, involving a large dense covariance matrix
- Computing this covariance matrix in large-scale is prohibitive and requires innovative ideas

# Geospatial Statistics Motivation

- With the explosion of spatial data coming from different sources such as sensors and monitoring devices, large scale computations became an important goal for associated applications
- Hardware availability and underlying supported algorithms in linear algebra motivated us to build robust parallel software to deal with large scale geospatial modeling and prediction

# Geospatial Statistics Motivation

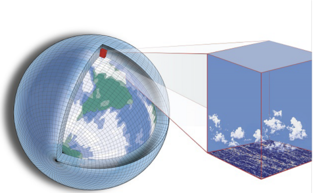
- "Increasing amounts of data are being produced (e.g., by remote sensing instruments and numerical models), while techniques to handle millions of observations have historically lagged behind. [...] computational implementations that work with irregularly-space observations are still rare." -Dorit Hammerling,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

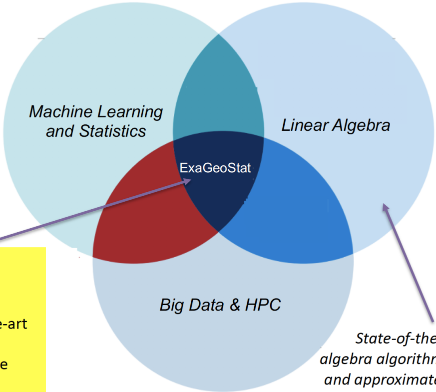
  - 1M X 1M DP matrix requires 8TB, $N^3 \approx 10^{18}$ Flops
  - Traditional approaches: Global low rank - Covariance tapering (zero outer diagonals)
  - Better approaches: Hierarchical low rank - Reduced precision outer diagonals

# Geostatistical Modeling in Exascale Era

- Parallel computing in Gaussian process calculation becomes a necessity to avoid computational and memory restrictions associated with Geostatistics applications
- The evaluation of the Gaussian log-likelihood function requires $O(n^2)$ storage and $O(n^3)$ operations
- Assume $n = 1M$, the total required space is $8\,TB$, the total number of flops is one Exaflops $(10^{18})$ flops
- Applications for climate and environmental predictions are among the principal simulation workloads running on today's supercomputer facilities
- However, hardware alone is not enough to scale applications, usually the parallel algorithms are limiting the hardware usage

# ExaGeoStat Framework






Exascale Geostatistics (ExaGeoStat) is A multidisciplinary software which exploits **machine learning**, **statistical modeling and forecasting**, the state-of-the-art **linear algebra algorithms,** **and supercomputing simulations** to handle large-scale Geostatistics data.

*State-of-the-Art linear algebra algorithm for both exact and approximated computation*

# ExaGeoStat in a Nutshell

- Exploits synergistically machine learning, statistical modeling and forecasting, HPC, and the state-of-the-art linear algebra techniques to process large-scale Geostatistics data
- Scales the statistical approaches to analyze large geostatistics data
- Optimizes the performance on different hardware architectures
- Includes an internal synthetic geostatistics data generator for conducting numerical simulations with different kernels to better understand the statistical models
- Supports currently univariate, multivariate, and space-time Gaussian stationary geostatistics, non-Gaussian (Tukey $g$-and-$h$) random fields (soon also covering non-stationary kernels)

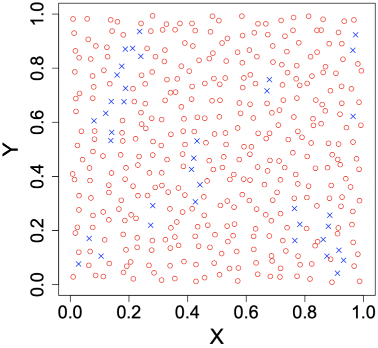
# ExaGeoStat Components

- Synthetic Dataset Generator
  - Generate large-scale Geospatial datasets which can be separately used as benchmark datasets for other software packages https://ecrc.github.io/exageostat/md_docs_examples.html
- Maximum Likelihood Estimator (MLE)
  - Evaluate the Gaussian maximum likelihood function on large-scale geospatial datasets
  - Support full machine precision accuracy (full-matrix) and Tile Low-Rank (TLR) approximation
- ExaGeoStat Predictor
  - Predict unknown measurements at new geospatial locations by leveraging the MLE estimated parameters

# Synthetic Dataset Generator

- Builds the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta}_t)$ using a specific kernel and truth parameter vector $\boldsymbol{\theta}_t$
- Computes Cholesky factorization of $\boldsymbol{\Sigma}(\boldsymbol{\theta}_t)$: $\boldsymbol{\Sigma}(\boldsymbol{\theta}_t) = \mathbf{V} \cdot \mathbf{V}^\top$
- Generates $\mathbf{Z}$ vector: $\mathbf{Z} = \mathbf{V} \cdot \mathbf{e}, \quad \mathbf{e} \sim N(0,1)$ i.i.d.

Figure: An example of 400 points irregularly distributed in space, with 362 points (o) for maximum likelihood estimation and 38 points (x) for prediction validation.



# 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 loop with different $\boldsymbol{\theta}$ to maximize the likelihood function estimation until convergence
  - Generate the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ using a specific kernel and the parameter vector $\boldsymbol{\theta}$ ($\boldsymbol{\theta}$ comes from the optimization function)
  - Log determinant and linear solver requires a Cholesky factorization of the given covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$
- Cholesky factorization requires $O(n^3)$ floating-point operations and $O(n^2)$ memory storage
- NLOPT optimization library has been used to maximize the likelihood function until convergence in both cases

# ExaGeoStat Predictor

- Assuming $\boldsymbol{\Sigma}_{11} \in \mathbb{R}^{m \times m}$, $\boldsymbol{\Sigma}_{12} \in \mathbb{R}^{m \times n}$, $\boldsymbol{\Sigma}_{21} \in \mathbb{R}^{n \times m}$, and $\boldsymbol{\Sigma}_{22} \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{bmatrix} \sim N_{m+n}\left( \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right)$$

- The associated conditional distribution can be represented as

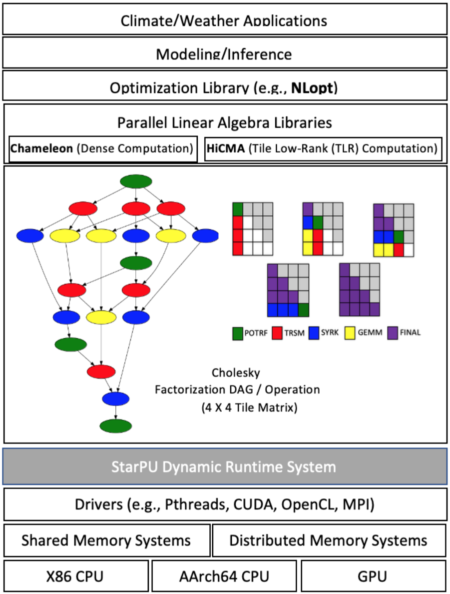
$$\mathbf{Z}_1 | \mathbf{Z}_2 \sim N_m(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{Z}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21})$$

- Assuming that the known measurements vector $\mathbf{Z}_2$ has a zero-mean function (i.e., $\boldsymbol{\mu}_1 = \mathbf{0}$ and $\boldsymbol{\mu}_2 = \mathbf{0}$), the unknown measurements vector $\mathbf{Z}_1$ can be predicted using

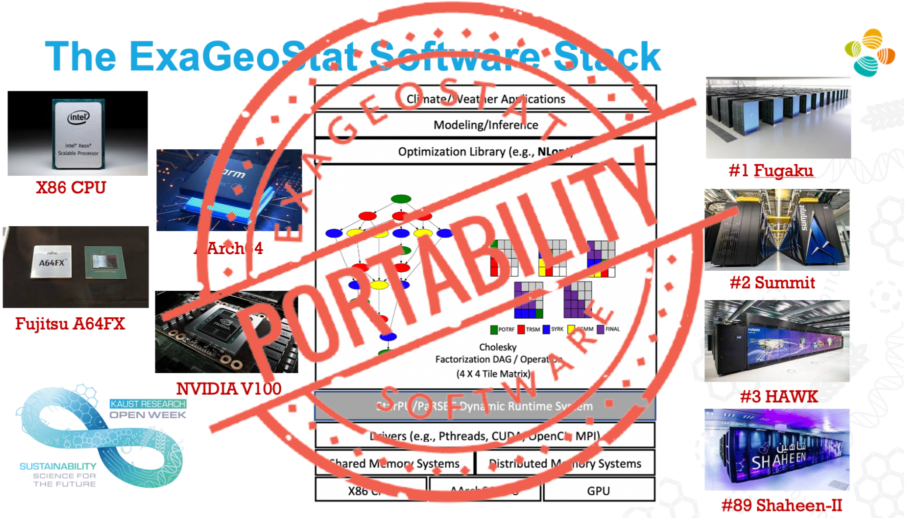
$$\mathbf{Z}_1 = \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\mathbf{Z}_2$$

- Solution of system of linear equation ($\boldsymbol{\Sigma}_{22}^{-1}\mathbf{Z}_2$) requires also the Cholesky factorization of $\boldsymbol{\Sigma}_{22}$

# ExaGeoStat Software Layers



# Portability



## Dynamic Runtime Systems

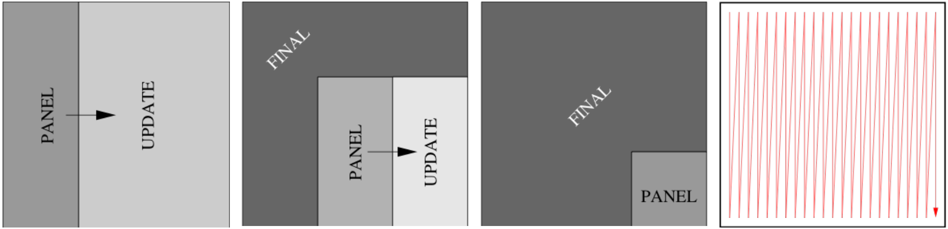
- Parallel coding on different hardware architectures requires different skills and coding tools:
  - Shared-memory systems (ex., OpenMP)
  - GPUs (ex., OpenCL, CUDA)
  - Distributed systems (ex., Message Passing Interface (MPI))
- Eventhough specific programming tools can utilize the usage of the underlying hardware architecture, it requires a lot of work to add portability capabilities to the parallel programs
- Dynamic Runtime Systems is the approach to the development of parallel applications
- Programmers only specify the potential parallelism in their programs but not how parallel execution is implemented on a specific system

# Dynamic Runtime Systems, Cont.

- Operate directly on the sequential code and schedule the various tasks across the underlying hardware resources (task-based parallelism)
- Ensure that the data dependencies are not violated
- Enhance the software productivity by abstracting the hardware complexity from the end users
- QUARK
  - UTK, US
  - QUeuing And Runtime for Kernels
  - Multi-core environment
- PaRSEC
  - UTK, US
  - Parallel Runtime Scheduling and Execution Controller
  - Shared Memory, GPUs, Distributed Systems
- StarPU
  - INRIA Bordeaux, France
  - A unified Runtime System for Heterogeneous Multicore Architectures
  - Shared Memory, GPUs, Distributed Systems

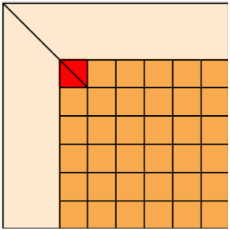
# State-of-the-art Linear Algebra Libraries

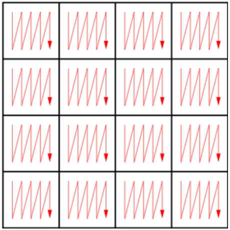
- Block-based algorithms
- LAPACK/MKL
  - The matrix computation is decomposed in two successive panels
  - Parallel performance is only exploited during the update of the training submatrix
  - Synchronization points in-between computational phases impede parallel performance
  - Block-columns algorithm
  - MKL is a vendor optimized library for LAPACK block algorithms



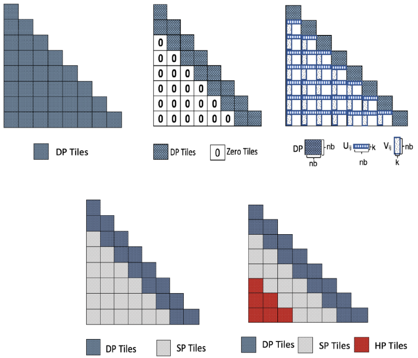
# State-of-the-art Linear Algebra Libraries

- Tile Algorithms
  - PLASMA, Chameleon, and FLAME
  - The dense matrix is broken into tiles
  - Weaken the synchronization points by bringing the parallelism in multithreaded BLAS



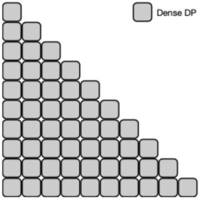


# ExaGeoStat Covariance Matrix Representation

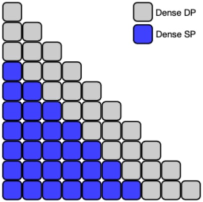



1- Exact Computation
2- Diagonal Super Tile approximation
3- Tile Low-Rank (TLR) approximation
4- Double/Single Precision approximation
5- Double/Single/Half Precision approximation

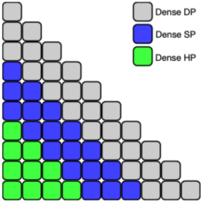
# Responsibly Reckless Matrix Algorithms for HPC



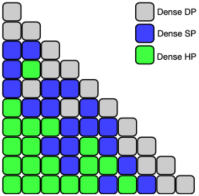

(a) Dense w/ FP64.




(b) Dense w/ band FP64/FP32.

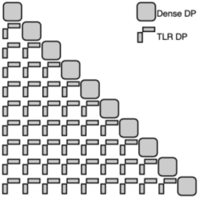



(c) Dense w/ band FP64/FP32/FP16.

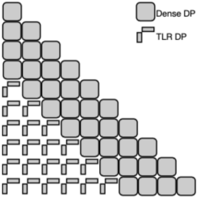



(d) Dense w/ adaptive tile-centric FP64/FP32/FP16.

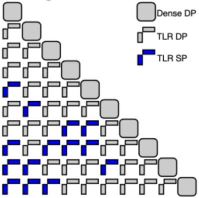
Fig. 2: Mixed-precision dense computations.



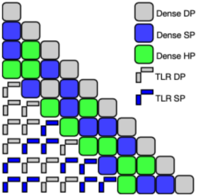

(a) Diag dense + TLR w/ FP64.




(b) Band dense + TLR w/ FP64.




(c) Diag dense + TLR w/ FP64/FP32.

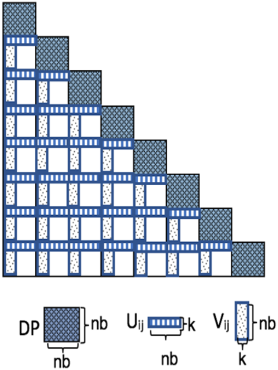



(d) Band dense + TLR w/ FP64/FP32/FP16.

Fig. 3: Combining mixed precisions and TLR approximations.

# Tile-Low Rank Approximation

- Tile Low-Rank (TLR) Algorithms
  - HiCMA Library (KAUST, 2017)
  - Use *SVD*, approximate each off-diagonal tile, keep the most significant $k$ (matrix rank) singular values and their left and right singular vectors, $U$ and $V$
  - Depend on Selected accuracy (application specific)
  - Two variations can be provided:
    - Fixed Rank
    - Fixed Accuracy
  - In the case of fixed accuracy, $k$ varies from one tile to another. Therefore, load imbalance issues appear
  - Solution: rely on dynamic runtime systems

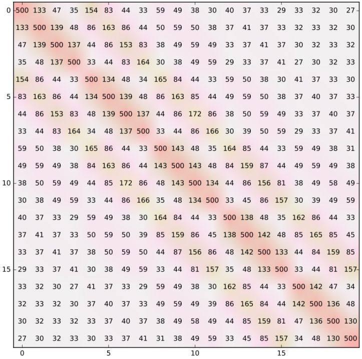


# Tile-Low Rank Approximation

- Climate/ weather modeling applications requires $10^{-9}$ accuracy threshold
- Using Exponential covariance function

$$C(h;\boldsymbol{\theta}) = \theta_1 \exp(-\frac{h}{\theta_2})$$

- Example, rank distribution on $2k \times 2k$ matrix where $nb = 500$, 2D problem



# ExaGeoStat Development Timeline

**2017 (Debut Year)**

ExaGeoStat V0.1.0/ ExaGeoStaR V0.1.0 were released with large scale spatial data generator, exact modeling and prediction capabilities

**2018 (TLR Year)**

ExaGeoStat V1.0.0/ ExaGeoStat V1.0.0 were released with TLR support through KAUST HiCMA library

**2019 (MP Year)**

Two-level mixed-precision ExaGeoStat with high accuracy achievements

**2020 (Covid19 Year)**

ExaGeoStat V1.1.0/ ExaGeoStat V1.0.1 were released. UTK PaRSEC runtime system integration. Three-level mixed-precision ExaGeoStat

**2021 (New Heights)**

2021 KAUST Spatial Statistics Competition. KAUST-NVIDIA Workshop- Geospatial data science competition. Space-Time modeling support. Parallel optimization with PSwarm algorithm. Non-Gaussian Modeling (Exact/TLR/HODLR)

**2022 (New Release)**

2022 KAUST Spatial Statistics Competition. ExaGeoStat V1.0.2 will be released. BM function to assess different modeling performance

# ExaGeoStat Under the Microscope

- ExaGeoStat is an open-source software which is available at https://github.com/ecrc/exageostat
- ExaGeoStatR is available at https://github.com/ecrc/exageostatR
- ExaGeoStat 0.1.0 (Nov. 9th 2017)
    - Support exact computation using Chameleon dense Linear algebra library and StarPU runtime system
    - Support real and synthetic geospatial datasets
- ExaGeoStat 1.0.0 (Nov. 6th 2018)
    - Tile-Low Rank approximation (TLR) using HiCMA TLR approximation
    - Super Diagonal Tile (SDT) approximation
    - Performance results of TLR-based computations on shared and distributed-memory systems attain up to $13X$ and $5X$ speedups
    - Support Out-Of-Core (OOC) execution
- ExaGeoStat 1.1.0 (June 6th 2020)
    - Mixed-precision approximation
    - Multivariate modeling support

# List of Publications


- Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2018). ExaGeoStat: A High Performance Unified Software for Geostatistics on Manycore Systems. IEEE Transactions on Parallel and Distributed Systems, 29(12), 2771-2784.
- Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2018, September). Parallel Approximation of the Maximum Likelihood Estimation for the Prediction of Large-scale Geostatistics Simulations. In 2018 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 98-108).
- Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2019, December). Geostatistical Modeling and Prediction using Mixed Precision Tile Cholesky Factorization. In 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 152-162). IEEE.
- Abdulah, S., Li, Y., Cao, J., Ltaief, H., Keyes, D. E., Genton, M. G., & Sun, Y. (2019). ExaGeoStatR: A Package for Large-scale Geostatistics in R. (arXiv:1908.06936).
- Hong, Y., Abdulah, S., Genton, M. G., & Sun, Y. (2021). Efficiency Assessment of Approximated Spatial Predictions for Large Datasets. Spatial Statistics, 43:100517.
- Huang, H., Abdulah, S., Sun, Y., Ltaief, H., Keyes, D. E., and Genton, M. G. (2021). Competition on spatial statistics for large datasets (with discussion). Journal of Agricultural, Biological, and Environmental Statistics, 26, 580-595.
- Salvaña, M. L. O., Abdulah, S., Huang, H., Ltaief, H., Sun, Y., Genton, M. G., & Keyes, D. E. (2021). High Performance Multivariate Geospatial Statistics on Manycore Systems. IEEE Transactions on Parallel and Distributed Systems, 32, 2719-2733.

# List of Publications

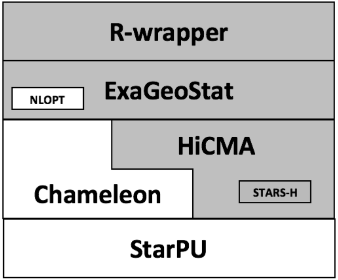

- Mondal, S., Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., and Keyes, D. E. (2022). Parallel approximations of the Tukey g-and-h likelihoods and predictions for non-Gaussian geostatistics. International Parallel and Distributed Processing Symposium, to appear.
- Salvaña, M. L., Abdulah, S., Ltaief, H., Sun, Y., Genton, M. G., and Keyes, D. E. (2022). Parallel space-time likelihood optimization for air pollution prediction on large-scale systems. In: Platform for Advanced Scientific Computing Conference (PASC '22), Basel, Switzerland, Article No. 17, 1-11.
- Abdulah, S., Cao, Q., Pei, Y., Bosilca, G., Dongarra, J., Genton, M. G., Keyes, D. E., Ltaief, H., & Sun, Y. (2022), Accelerating Geostatistical Modeling and Prediction with Mixed-precision Computations: A High-productivity Approach with PaRSEC. IEEE Transactions on Parallel and Distributed Systems, 33, 964-976.
- Cao, Q., Abdulah, S., Alomairy, R., Pei, Y., Nag, P., Bosilca, G., Dongarra, J., Genton, M. G., Keyes, D. E., Ltaief, H., and Sun, Y. (2022). Reshaping geostatistical modeling and prediction for extreme-scale environmental applications. Super Computing 2022 (finalist for Gordon Bell prize), to appear.

# Part III

## ExaGeoStatR Package

# ExaGeoStatR

- ExaGeoStatR is a package for large-scale Geostatistics in R that supports parallel computation of the Gaussian maximum likelihood function on shared memory, GPU, and distributed memory systems



# Existing Gaussian Likelihood Calculations in R packages

| Package | **geoR** | **fields** | **ExaGeoStatR** |
|---|---|---|---|
| Function name | `likfit` | `MLESpatialProcess` | `exact_mle` |
| Mean | estimated | estimated | fixed as zero |
| Variance | estimated | estimated | estimated |
| Spatial Range | estimated | estimated | estimated |
| Smoothness | estimated | fixed | estimated |
| Default optimization method | `Nelder-Mead` | `BFGS`[1] | `BOBYQA`[2] |

[1]. BFGS: Broyden-Fletcher-Goldfarb-Shanno. [2]. BOBYQA: bound optimization by quadratic approximation

# Existing Gaussian Likelihood Calculations in R packages (geoR-Example)

- $library(geoR)$
- $sigma\_sq = 1$
- $beta = 0.1 \qquad \#choose one from c(0.3, 0.1, 0.03)$
- $nu = 0.5 \qquad \#choose one from c(0.5, 1, 2)$
- $sims = grf(n = 1600, grid = "reg", cov.pars = c(sigma_sq, beta), kappa = nu, RF = FALSE)$
- $Rdata = list(x = sims\$coords[, 1], y = sims\$coords[, 2], z = sims\$data)$

# Existing Gaussian Likelihood Calculations in R packages (fields-Example)

- $library(fields)$
- $grid = list(x = (1:40)/20, y = (1:40)/20)$
- $xy = expand.grid(x = (1:40)/20, y = (1:40)/20)$
- $obj = matern.image.cov(grid = grid, theta = 0.1, smoothness = 0.5, setup = TRUE)$
- $sigma\_sq = 1$
- $sims.fields = sqrt(sigma\_sq) * sim.rf(obj)$
- $data.fields.reg = list(x = xy[, 1], y = xy[, 2], +z = c(sims.fields))$

# ExaGeoStatR Installation

- ExaGeoStatR prerequisites
    - Intel MKL (set MKLROOT): Link
    - GIT (through Homebrew or Link)
    - CMake (through Homebrew or Link)
    - pkg-config (through Homebrew or Link)
    - gfortran (through Homebrew or Link)
    - wget (through Homebrew or Link)
    - NLopt (self-installation)
    - GSL (self-installation)
    - hwloc (self-installation)
    - StarPU (self installation)
    - Chameleon (Self-installation)
    - Stars-H (Self installation)
    - HiCMA (Self-installation)

# ExaGeoStatR Installation

- Installing ExaGeoStatR from GitHub

```
library("devtools")
install_git(url="https://github.com/ecrc/exageostatR" )
```

- To enable MPI support for distributed memory systems

```
library("devtools")
install_git(url="https://github.com/ecrc/exageostatR", configure.args=C)'--enable-mpi'))
```

- To enable CUDA support for GPU systems

```
library("devtools")
install_git(url="https://github.com/ecrc/exageostatR", configure.args=C)'--enable-cuda'))
```

# ExaGeoStatR Main Functions

| Function Name | Description |
|---|---|
| exageostat_init | Initiate ExaGeoStat instance. |
| simulate_data_exact | Generate $\mathbf{Z}$ measurements vector. |
| simulate_obs_exact | Generate $\mathbf{Z}$ measurements vector on $n$ given 2D locations. |
| exact_mle | Exact parameter vector evaluation. |
| dst_mle | DST approximation parameter vector evaluation. |
| tlr_mle | TLR approximation parameter vector evaluation. |
| exageostat_finalize | Finalize active ExaGeoStat instance. |

# Code Examples (Example 1)

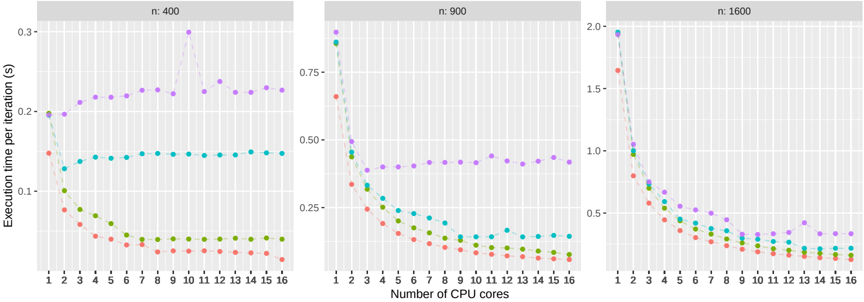
- Generating **Z** vector using random irregular (x, y) locations with exact MLE computation

```
library("exageostatr")                                          #Load ExaGeoStatR lib.
seed          = 0                                               #Initial seed to generate XY locs.
sigma_sq      = 1                                               #Initial variance.
beta          = 0.1                                             #Initial smoothness.
nu            = 0.5                                             #Initial range.
dmetric       = 0                                               #0 --> Euclidean distance,
                                                                #1--> great circle distance.
n             = 1600                                            #n*n locations grid.
#theta_out[1:3]                  = -1.99
exageostat_init(hardware = list (ncores=2, ngpus=0,
ts=320, pgrid=1, qgrid=1))#Initiate exageostat instance
#Generate Z observation vector
data          = simulate_data_exact(sigma_sq, beta, nu,
dmetric, n, seed) #Generate Z observation vector
#Estimate MLE parameters (Exact)
result        = exact_mle(data, dmetric, optimization = list(clb = c(0.001, 0.001, 0.001),
cub = c(5, 5,5 ), tol = 1e-4, max_iters = 20))

#print(result)
#Finalize exageostat instance
exageostat_finalize()
```

# ExaGeoStatR on Shared-memory System

- 16-core Intel Sandy Bridge Xeon E5-2650 Chip

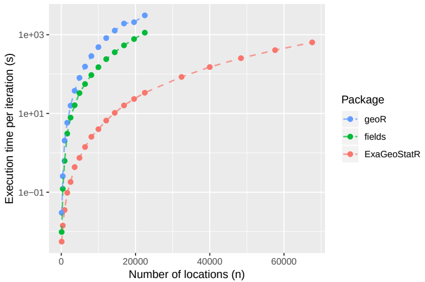


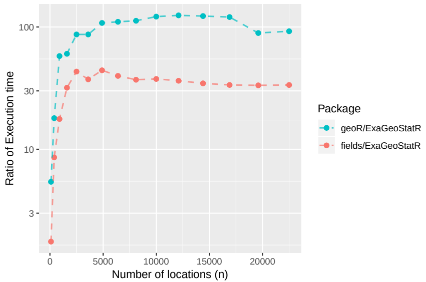
# geoR - fields - ExaGeoStatR Comparison (Time)

- Average on 100 samples

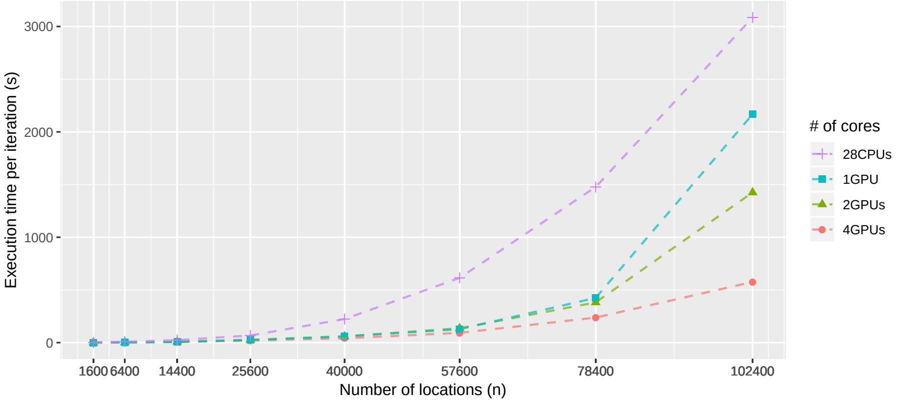
| The average execution time per iteration (seconds) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Package | **geoR** | | | **fields** | | | **ExaGeoStatR** | | |
| $\nu=$ \ $\beta=$ | 0.03 | 0.1 | 0.3 | 0.03 | 0.1 | 0.3 | 0.03 | 0.1 | 0.3 |
| 0.5 | 1.39 | 1.49 | 1.47 | 0.75 | 0.97 | 0.99 | 0.10 | 0.12 | 0.12 |
| 1 | 1.35 | 1.49 | 1.56 | 0.66 | 0.90 | 0.90 | 0.09 | 0.13 | 0.13 |
| 2 | 1.34 | 1.56 | 1.57 | 0.67 | 0.91 | 0.93 | 0.09 | 0.13 | 0.13 |
| The average number of iterations to reach the tolerance | | | | | | | | | |
| $\nu=$ \ $\beta=$ | 0.03 | 0.1 | 0.3 | 0.03 | 0.1 | 0.3 | 0.03 | 0.1 | 0.3 |
| 0.5 | 160 | 157 | 135 | 73 | 72 | 70 | 231 | 204 | 237 |
| 1 | 193 | 33 | 23 | 75 | 75 | 80 | 318 | 320 | 275 |
| 2 | 216 | 25 | 20 | 100 | 70 | 85 | 427 | 436 | 332 |

# geoR - fields - ExaGeoStatR Comparison (Speedup)

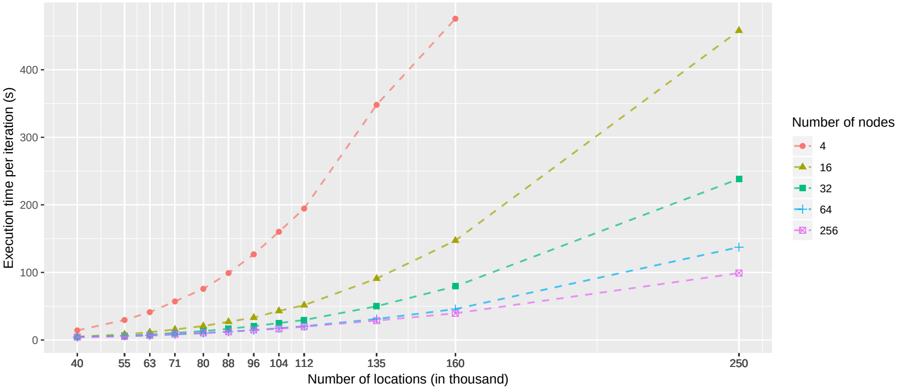


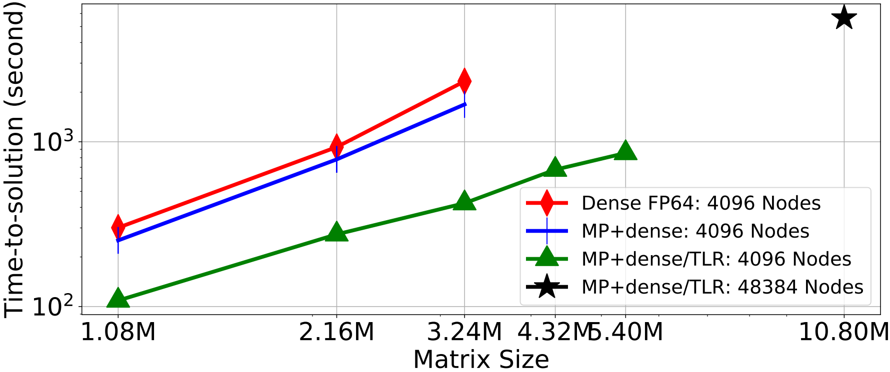
# ExaGeoStatR Performance on Heterogeneous System (CPU/GPU)



# ExaGeoStatR Performance on Distributed-Memory System (Shaheen-II)



# ExaGeoStat Performance on Distributed-Memory System (Fugaku)




Performance of Matérn 2D space-time of strong correlation on 4096 and 48384 Fugaku nodes

# Code Examples (Example 2)

- Generating **Z** vector using random irregular (x, y) locations with TLR MLE computation

```
library("exageostatr")                                   #Load ExaGeoStatR lib.
seed            = 0                                      #Initial seed to generate XY locs.
sigma_sq        = 1                                      #Initial variance.
beta            = 0.03                                   #Initial smoothness.
nu              = 0.5                                    #Initial range.
dmetric         = 0                                      #0 --> Euclidean distance,
                                                         #1--> great circle distance.
n               = 900                                    #n*n locations grid.
tlr_acc         = 7                                      #Approximation accuracy 10^-(acc)
tlr_maxrank     = 450                                    #Max Rank

#Initiate exageostat instance
exageostat_init(hardware = list (ncores=2, ngpus=0,
ts=320, lts=600,  pgrid=1, qgrid=1))#Initiate exageostat instance
#Generate Z observation vector
data            = simulate_data_exact(sigma_sq, beta, nu,
dmetric, n, seed) #Generate Z observation vector
#Estimate MLE parameters (TLR approximation)
result          = tlr_mle(data, tlr_acc, tlr_maxrank,  dmetric, optimization =
list(clb = c(0.001, 0.001, 0.001), cub = c(5, 5,5 ), tol = 1e-4, max_iters = 20))
#print(result)
#Finalize exageostat instance
exageostat_finalize()
```

# Code Examples (Example 3)

- Generating **Z** vector using random (x, y) irregular locations with DST MLE computation

```
library("exageostatr")                                          #Load ExaGeoStatR lib.
seed            = 0                                              #Initial seed to generate XY locs.
sigma_sq        = 1                                              #Initial variance.
beta            = 0.03                                           #Initial smoothness.
nu              = 0.5                                            #Initial range.
dmetric         = 0                                              #0 --> Euclidean distance,
                                                                 #1--> great circle distance.
n               = 900                                            #n*n locations grid.
dst_thick       = 3                                              #Number of used Diagonal Super Tile (DST
#Initiate exageostat instance
exageostat_init(hardware = list (ncores=4, ngpus=0,
ts=320, lts=0,  pgrid=1, qgrid=1))
#Generate Z observation vector
data            = simulate_data_exact(sigma_sq, beta, nu,
dmetric, n, seed) #Generate Z observation vecto
#Estimate MLE parameters (DST approximation)
result          = dst_mle(data, dst_thick, dmetric, optimization =
list(clb = c(0.001, 0.001, 0.001), cub = c(5, 5,5 ), tol = 1e-4, max_iters = 20))
#print(result)
#Finalize exageostat instance
exageostat_finalize()
```

# Code Examples (Example 4)

- Generating **Z** vector using given (x, y) locations with exact MLE computation

```
library("exageostatr")                                                #Load ExaGeoStatR lib.
sigma_sq        = 1                                                   #Initial variance.
beta            = 0.1                                                 #Initial smoothness.
nu              = 0.5                                                 #Initial range.
dmetric         = 0                                                   #0 --> Euclidean distance,
                                                                      #1--> great circle distance.
n               = 1600                                                #n*n locations grid.
x               = rnorm(n = 1600, mean = 39.74, sd = 25.09)     #x measurements of n locations.
y               = rnorm(n = 1600, mean = 80.45, sd = 100.19)    #y measurements of n locations.
#Initiate exageostat instance
exageostat_init(hardware = list (ncores=2, ngpus=0,
ts=320, lts=0,  pgrid=1, qgrid=1))#Initiate exageostat instance
#Generate Z observation vector based on given locations
data            = simulate_obs_exact( x, y, sigma_sq, beta, nu, dmetric)
#Estimate MLE parameters (Exact)
result          = exact_mle(data, dmetric, optimization =
list(clb = c(0.001, 0.001, 0.001), cub = c(5, 5,5 ), tol = 1e-4, max_iters = 20))
#print(result)
#Finalize exageostat instance
exageostat_finalize()
```

# Code Examples (Example 5)

- Batch R script on Distributed Environment Example

```
#!/bin/bash
#SBATCH --job-name=job_name
#SBATCH --output=output_file.txt
#SBATCH --partition=XXXX
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=31
#SBATCH --time 00:30:00

srun Rscript Rtest.r
```

# Thank You!

Questions?