GPU-Accelerated Dense Covariance Matrix Generation for Spatial Statistics Applications

Zipei Geng[1], Sameh Abdulah[2], Hatem Ltaief[2], Ying Sun[12], Marc G. Genton[12], David E. Keyes[2]

[1]Statistics Program, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

[2]Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

{zipei.geng,sameh.abdulah,hatem.ltaief,ying.sun,marc.genton,david.keyes}@kaust.edu.sa

ABSTRACT

Large-scale parallel computing is crucial in Gaussian regressions to reduce the time complexity of spatial statistics applications. The log-likelihood function is utilized to evaluate the Gaussian model for a set of measurements in N geographical locations. Several studies have shown an utilization of modern hardware to scale the log-likelihood function for handling large numbers of locations. *ExaGeoStat* is an example of software that allows parallel statistical parameter estimation from the log-likelihood function. However, generating a covariance matrix is mandatory and challenging when estimating the log-likelihood function. In *ExaGeoStat*, the generation process was performed on CPU hardware due to missing math functions in CUDA libraries, e.g., the modified Bessel function of the second kind. This study aims to optimize the generation process using GPU with two proposed generation schemes: pure GPU and hybrid. Our implementations demonstrate up to 6X speedup with pure GPU and up to 1.5X speedup with the hybrid scheme.

KEYWORDS

Parallel Computing, GPU Acceleration, Covariance Matrix Generation, Spatial Statistics

ACM Reference Format:

Zipei Geng[1], Sameh Abdulah[2], Hatem Ltaief[2], Ying Sun[12], Marc G. Genton[12], David E. Keyes[2]. 2023. GPU-Accelerated Dense Covariance Matrix Generation for Spatial Statistics Applications. In *Proceedings of (SuperComputing Research Poster '23).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Spatial statistics applications aim to study patterns, distributions, and relationships of variables across different spatial locations in a given spatial region. At present, the abundance of massive spatial data volumes makes High-Performance Computing indispensable for effectively handling the processing of high-resolution maps. Numerous attempts to scale existing modeling and prediction methods in spatial statistics have been proposed in the literature [1, 2, 4]. For Example, *ExaGeoStat* [1] is a high-performance computational framework tailored to address large-scale spatial data challenges in climate and environmental studies. *ExaGeoStat* assumes that spatial measurements reflect a Gaussian spatial random field. The corresponding Gaussian log-likelihood is the most commonly used objective. In other literature related to machine learning, this concept is commonly referred to as Gaussian regression [6]. This function operates with various covariance matrices generated from predefined covariance functions. Given the nature of spatial problems, they typically involve many measurements taken at regular or irregular locations across a geographical area. This necessitates that the synthetic dataset generation process can handle dense exascale matrices. The statistical framework is given by first denoting spatial measurements $(z_1, ..., z_N) := (Z(\boldsymbol{s}_1), ..., Z(\boldsymbol{s}_N))$, where $Z(\boldsymbol{s})$ denotes the realization of the Gaussian random field at location $\boldsymbol{s}$. We further assume that the mean function of this random field is $m(\boldsymbol{s})$. Often, $m(\boldsymbol{s})$ will be constant, which serves as the prerequisite of a stationary random field. The kernel matrix of the random field is $\Sigma(\boldsymbol{\theta}) := \big(C(\boldsymbol{s}_i - \boldsymbol{s}_j; \boldsymbol{\theta})\big)_{ij}, i, j \in 1, ..., N$, where N is the number of spatial locations, $\boldsymbol{s}_i - \boldsymbol{s}_j$ is the spatial lag vector of location $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, and $C(\cdot\,; \boldsymbol{\theta})$ is a stationary parametric covariance/kernel function w.r.t. spatial lag vectors. Lastly, the frequent use of Gaussian random fields in geospatial studies can be attributed to their unique characteristic. Given a mean function and a kernel matrix, a Gaussian random field can be uniquely defined [6]. This property allows for precise and consistent spatial data modeling, which is crucial in geospatial studies.

Moreover, *ExaGeoStat* uses a 'black-box' (zero-gradient/derivative-free) optimization method known as BOBYQA [5]. The BOBYQA algorithm fits a quadratic model to the function to minimize (or maximize) the log-likelihood function and obtains a set of statistical parameters that represents the underlying spatial region. To precisely evaluate the function value of log-likelihood, the MLE objective function can be expressed with the following formula:

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

As dimensions increase, the computational demand typically grows significantly. The state-of-the-art linear algebra system MAGMA [3] can deal with matrix multiplication, determinant evaluation, and matrix inversion on GPU hardware accelerators. However, we also need fast covariance matrix generation on GPUs. Prior works exclusively relied on the CPU for matrix generation. Instead, the current work focuses on leveraging the power of GPUs to generate the covariance matrix for spatial statistics applications.

2 PROPOSED GPU ACCELERATED SCHEMES

Namely, we proposed two schemes to implement :

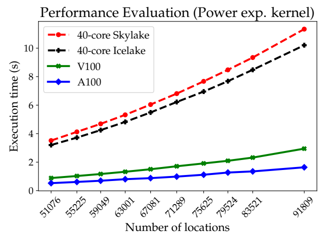
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SuperComputing Research Poster '23, November 12-17, 2023, Denver, CO

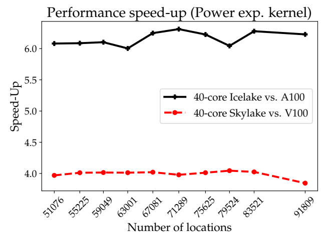
© 2023 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

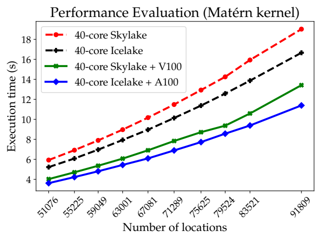
https://doi.org/10.1145/nnnnnnn.nnnnnnn



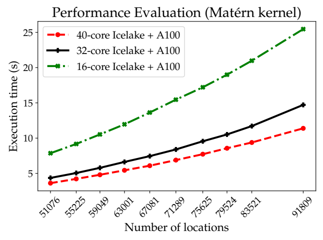
(a) **Power exp. kernel.**



(b) **Power exp. kernel (speed-up).**



(c) **Matérn kernel.**



(d) **Matérn kernel (scalability).**

Figure 1: Performance evaluation on two different machines equipped with GPUs.

(1) **Pure GPU:** users can adopt this scheme when there are no special functions that cannot be evaluated on the GPU, such as the modified Bessel function of the second kind, which lacks a GPU-based implementation. We have demonstrated the benefits of GPU acceleration using the representative power exponential kernel for covariance matrix generation. Namely,

$$C_1(\|\boldsymbol{h}\|_2) = \sigma^2 \exp\left(\frac{-\|\boldsymbol{h}\|_2^{\beta_2}}{\beta_1}\right),$$

where σ^2 is the variance, β_1 and β_2 are corresponding parameters which uniquely decide the kernel function. In this scheme, matrix $\mathbf{M}_0$ is initialized with zeros and split into tiles $\boldsymbol{T}_0 := (T_{01}, ..., T_{0n_t})$, where n_t denotes the number of tiles the matrix $\mathbf{M}_0$ has, locations are then generated, and parameters are set on the CPU. Then, these variables are copied to the GPU, and the corresponding kernel function $f(\cdot) := C_1(\|\boldsymbol{h}\|_2)$ is evaluated on tiles $\boldsymbol{T}_0$ to generate tiles $\boldsymbol{T}_1$ with entries $f(\cdot)$. The tiles $\boldsymbol{T}_1$ form the covariance matrix $\mathbf{M}_1$.

(2) **Hybrid:** users can employ this scheme in situations where certain functions cannot be handled by the GPU. In such cases, we first evaluate these functions on the CPU and store the results in memory. Subsequently, the pre-computed matrices will be utilized to generate the kernel matrices. We can show the advantage of GPU acceleration using the representative Matérn kernel to generate the covariance matrix. Namely,

$$C_2(\|\boldsymbol{h}\|_2) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\|\boldsymbol{h}\|_2}{\beta}\right)^{\nu} \mathcal{K}_\nu\left(\frac{\|\boldsymbol{h}\|_2}{\beta}\right),$$

where the modified Bessel function of the second kind is given by $\mathcal{K}_\nu$. In this scheme, matrix $\mathbf{M}_0$ is initialized with zeros and split into tiles $\boldsymbol{T}_0$ (same split pattern as Pure GPU scheme), locations are then generated, and parameters are set on the CPU. Then, these variables are used to evaluate the corresponding special math function $f_1(\cdot) := \mathcal{K}_\nu\left(\frac{\|\boldsymbol{h}\|_2}{\beta}\right)$ and these values are used as entries in penultimate/intermediate tiles $\boldsymbol{T}_1$. Utilizing $\boldsymbol{T}_1$, the program uses the GPU with the corresponding function

$$f_2(\cdot) := \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\|\boldsymbol{h}\|_2}{\beta}\right)^{\nu}$$

and generate the tiles $\boldsymbol{T}_2$ by algebraic multiplication $f_1(\cdot) \times f_2(\cdot)$. Finally, $\boldsymbol{T}_2$ forms the kernel matrix $\mathbf{M}_1$. These schemes promise a notable increase in efficiency and speed in handling such vast computational tasks, as shown in the numerical experiment we conducted.

3 RESULTS

We evaluated our proposed implementations by conducting a comparison with the CPU-based matrix generation on two distinct systems. The first machine consists of a dual-socket 20-core Intel Skylake Xeon Platinum 8260 CPU running at 2.40 GHz and an NVIDIA Tesla V100 GPU SXM2 with 32GB memory. The second machine comprises a dual-socket 28-core Intel Icelake Xeon Gold 6330 running at 2.00 GHz and an NVIDIA A100 Tensor Core GPU with 40GB memory. To conduct the performance comparison, we repeated each experiment five times and calculated the average time consumption from the results.

Figures 1a and 1b illustrate the performance on various hardware configurations when employing the power exponential kernel to generate the covariance matrix. Herein, the pure GPU approach demonstrates a significant speed-up, generating the power exponential kernel matrix approximately 4X to 6X faster than the CPU. As the number of locations increases, the advantages of utilizing GPU become even more evident. Moreover, the performance of the NVIDIA A100 GPU surpasses that of the NVIDIA V100 GPU thanks to the hardware technology scaling observed across GPU generations.

Figures 1c and 1d show the performance with the Matérn kernel. The hybrid GPU/CPU outperforms the CPU version with Matérn. Yet, some CPU computations lessen hybrid advantages over pure GPU. We also witnessed the scalability of our hybrid implementation across various core counts, extending up to 40 cores on a shared-memory system architecture.

REFERENCES

[1] Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G. Genton, and David E. Keyes. 2018. ExaGeoStat: A High Performance Unified Software for Geostatistics on Manycore Systems. *IEEE Transactions on Parallel and Distributed Systems* 29, 12 (Dec. 2018), 2771–2784. https://doi.org/10.1109/TPDS.2018.2850749 arXiv:1708.02835 [cs].
[2] Sameh Abdulah, Hatem Ltaief, Ying Sun, Marc G Genton, and David E Keyes. 2018. Parallel approximation of the maximum likelihood estimation for the prediction of large-scale geostatistics simulations. In *2018 IEEE international conference on cluster computing (CLUSTER)*. IEEE, 98–108.
[3] Wieb Bosma, John Cannon, and Catherine Playoust. 1997. The Magma algebra system. I. The user language. *J. Symbolic Comput.* 24, 3-4 (1997), 235–265. https://doi.org/10.1006/jsco.1996.0125 Computational algebra and number theory (London, 1993).
[4] Robert B Gramacy. 2016. laGP: large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software* 72 (2016), 1–46.
[5] M J D Powell. 2007. The BOBYQA algorithm for bound constrained optimization without derivatives. (2007).
[6] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*. MIT Press, Cambridge, Mass. OCLC: ocm61285753.