Scalable Spatial Statistical Approximations as Emerging MPI Workloads in Climate and Weather Applications

Sameh Abdulah[1]
In collaboration with Qilong Pan[2], Hatem Ltaief [1], Marc G. Genton[2], David E. Keyes [1], and Ying Sun[2]
sameh.abdulah@kaust.edu.sa

[1]Applied Mathematics and Computational Sciences (AMCS) Program
[2]Statistics Program
King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.

August 20, 2025

Typical MPI workloads

Compute-bound

Heavy computation and minimal communication

Memory-bound

High memory bandwidth and limited CPU/GPU usage

Communication-bound

Frequent data exchange between processes

I/O-bound

Heavy disk read/write operations and I/O latency issues

GPUs Computing — Raw Compute Power

GPUs Lead in Raw Compute Power

- **NVIDIA H100 (2024):**
  - Peak FP64: **30 TFLOP/s** (More with Tensor Core)
  - Peak FP32: **60 TFLOP/s** (More with Tensor Core)
  - Memory Bandwidth: **3.35 TB/s** (HBM3)
  - NVLink 4.0: **900 GB/s** per GPU (bi-directional)
- **CPU Comparison — Intel Sapphire Rapids:**
  - FP64: ˜2 TFLOP/s
  - Memory Bandwidth: ˜300 GB/s

Key Insight

- GPUs provide **massively higher FLOP/s and memory bandwidth**.
- To leverage this, applications must **maximize arithmetic intensity**.

GPUs Computing — Why Compute-Bound Wins

Motivation to Favor Compute-Bound Workloads

Factor	Compute-Bound	Communication-Bound
GPU Utilization	✓ High (massive threads)	✗ Latency/sync overhead
Scalability	✓ Excellent	✗ Degrades at scale
Energy Efficiency	✓ High flops per watt	✗ Idle/wait energy waste
Optimization	✓ Easy to tile/vectorize	✗ Complex MPI+GPU overlap

Strategic Takeaway

- **Modern GPUs shine in compute-bound regimes**.
- Shift your algorithms toward:
  - **More local computation, less communication**
  - Batched/fused kernels and fewer synchronizations
- Result: Better **performance**, **scalability**, and **energy efficiency**.

MPI Workloads in Climate & Weather PDE Models

Model	Dominant MPI Workload	Hot MPI Ops
WRF	**Communication-bound**	`MPI_Isend/Irecv` + `MPI_Waitall`, `MPI_Allreduce`
FV3	**Communication-bound**	`Isend/Irecv`, `Iallreduce`
ICON	**Communication-bound**	`MPI_Neighbor_alltoallv`, `Isend/Irecv`
MPAS-A	**Communication-bound**	`Isend/Irecv`, `Allreduce`
IFS (spectral)	**Communication-bound**	`MPI_Alltoall(v)`, `Allreduce`
CESM / E3SM	**Communication-bound**	`Isend/Irecv`, `Allreduce`, `Allgather`
Ocean comps	**Communication-bound**	`Isend/Irecv`, `Allreduce`
I/O & Coupling	**I/O-bound**; **Comm-bound**	MPI-IO via HDF5/netCDF-4, `MPI_File_*`, `Allreduce`

Rethinking About Climate Modeling/Forecasting

From Physics-Based to Data-Driven Modeling

- Traditional models rely on solving PDEs (e.g., Navier-Stokes) with fine-grained grids and time steps
- These models are:
  - **Memory-bound** and **communication-heavy** on modern hardware
  - Difficult to scale efficiently on GPUs and MPI-based HPC systems

AI & Spatial Statistics: Natural Fit for GPU Era

- Graph Neural Networks (GNNs), Transformers, or Gaussian Process:
  - Are inherently **compute-bound** — ideal for GPUs
  - Offer **low communication overhead**

- To fully utilize GPU compute and MPI scalability, the community should:
  - Shift more resources toward **statistical emulation** and **ML-based forecasting**
- Future models may **bypass PDE bottlenecks** entirely, achieving faster, scalable, and energy-efficient forecasting.

Introduction to Gaussian Processes (GPs)

Gaussian Processes (GPs): are probabilistic models used primarily for regression and classification tasks in statistics and machine learning. It provides a principled way of modeling uncertainty over functions.

GPs for Spatial Statistics: Spatial data shows location-based dependence, with nearby points being more similar. GPs model this correlation to interpolate missing values, quantify prediction uncertainty, and capture smooth spatial trends.

Example

Assume there are n locations, $\mathbf{s}_1, \ldots, \mathbf{s}_n \in \mathbb{R}^d$ and their observations $\mathbf{y} = (y_1, \ldots, y_n)^\top$ where $y_i \in \mathbb{R}$. We model the data $\mathbf{y}$ as $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ where covaraince matrix is determined by $C_{\boldsymbol{\theta}}(\mathbf{s}_i, \mathbf{s}_j)$.

The inference about $\boldsymbol{\theta}$ is often based on the Gaussian log-likelihood function:

$$\ell(\boldsymbol{\theta}; \mathbf{y}) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| - \frac{1}{2}\mathbf{y}^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{y}. \tag{1}$$

Matérn Covariance Function

- The Matérn covariance function is a widely used kernel in Gaussian Processes for spatial statistics. It provides a flexible way to model the smoothness of spatial processes.

Parameterization

The popular parameterization of Matérn covariance function:

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

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

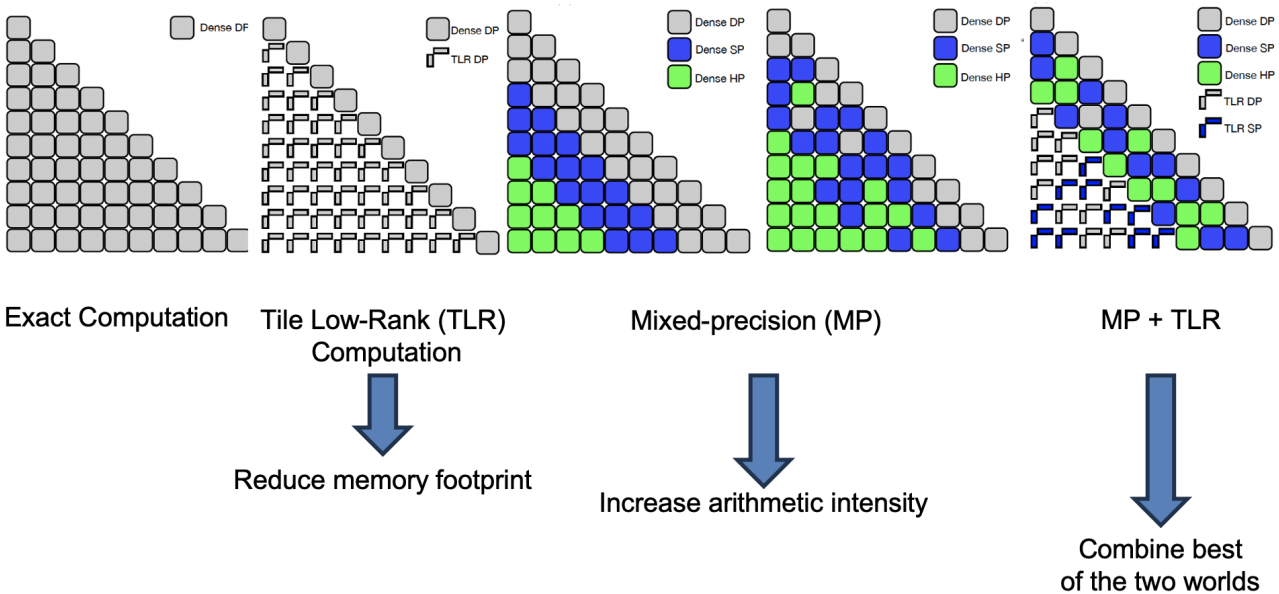
- The four parameters determining the covariance structure are: the partial sill σ^2, range $\beta > 0$, smoothness $\nu > 0$, and nugget τ^2

Maximum Likelihood Estimator (MLE)

Likelihood Function

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \{\log |\Sigma(\theta)| + \boldsymbol{Z}^\top \Sigma(\theta)^{-1} \boldsymbol{Z}\}$$

- Computing $\Sigma(\theta)^{-1}$ is computationally demanding, with $\mathcal{O}(n^3)$ complexity in time and $\mathcal{O}(n^2)$ complexity in memory for n spatial locations.
- **Full-Matrix Computation Methods:**



ExaGeoStat
https://github.com/ecrc/exageostat
https://github.com/ecrc/exageostatcpp

Classic Vecchia Approximation 1/2

- The Vecchia approximation restructures the GP's joint distribution into sequential univariate conditionals, each dependent on a limited subset of earlier observations.
- The likelihood can be written as a product of conditional densities:

$$L(\boldsymbol{\theta}; y) = p_{\boldsymbol{\theta}}\left(y_1, \ldots, y_n\right) \tag{2}$$

$$= p_{\boldsymbol{\theta}}\left(y_1^{\tau}\right) \prod_{i=2}^{n} p_{\boldsymbol{\theta}}\left(y_i^{\tau} \mid y_1^{\tau}, \ldots, y_{i-1}^{\tau}\right), \tag{3}$$

where τ is any permutation. Vecchia approximation replaces the complete conditioning vectors $\left(y_1^{\tau}, \ldots, y_{i-1}^{\tau}\right)$, with a subvector.

$$p_{\boldsymbol{\theta},\tau,J}\left(y_1, \ldots, y_n\right) = p_{\boldsymbol{\theta}}\left(y_1^{\tau}\right) \prod_{i=2}^{n} p_{\boldsymbol{\theta}}\left(y_i^{\tau} \mid y_{j_{i1}}^{\tau}, \ldots, y_{j_{im_i}}^{\tau}\right) \tag{4}$$

$$= p_{\boldsymbol{\theta}}\left(y_1^{\tau}\right) \prod_{i=2}^{n} p_{\boldsymbol{\theta}}\left(y_i^{\tau} \mid y_{J_i}^{\tau}\right). \tag{5}$$

$J = \{J_1, \ldots, J_n\}$, where each J_i is defined as $\{j_{i1}, \ldots, j_{im_i}\}$ and is referred to as the conditioning set containing m_i nearest neighbors

Classic Vecchia Approximation 2/2

- Independent Computing,

$$p_{\boldsymbol{\theta}}\left(y_1^{\tau}\right)\prod_{i=2}^{n}p_{\boldsymbol{\theta}}\left(y_i^{\tau}\mid y_{J_i}^{\tau}\right),$$

  where n conditional distributions is independent of each other;

- Abundant and Light-weight computation task,

$$\#\boldsymbol{J}_i \ll n,$$

  the conditioning sizes $\#\boldsymbol{J}_i$, e.g., 30, and the number of tasks, e.g., 1M

- We developed a highly optimized GPU-based implementation of the Vecchia approximation algorithm presented in:

Reference

Pan, Q., Abdulah, S., Genton, M. G., Keyes, D. E., Ltaief, H., and Sun, Y. (2024, May). GPU-accelerated Vecchia approximations of Gaussian processes for geospatial data using batched matrix computations. In ISC High Performance 2024 Research Paper Proceedings (39th International Conference) (pp. 1-12).

Motivations

The BV approximation allows GP modeling efficiently, but it cannot scale to even large problems, e.g., **billion-level** and **high-dimensional** problems GP emulators are widely used in various research areas:

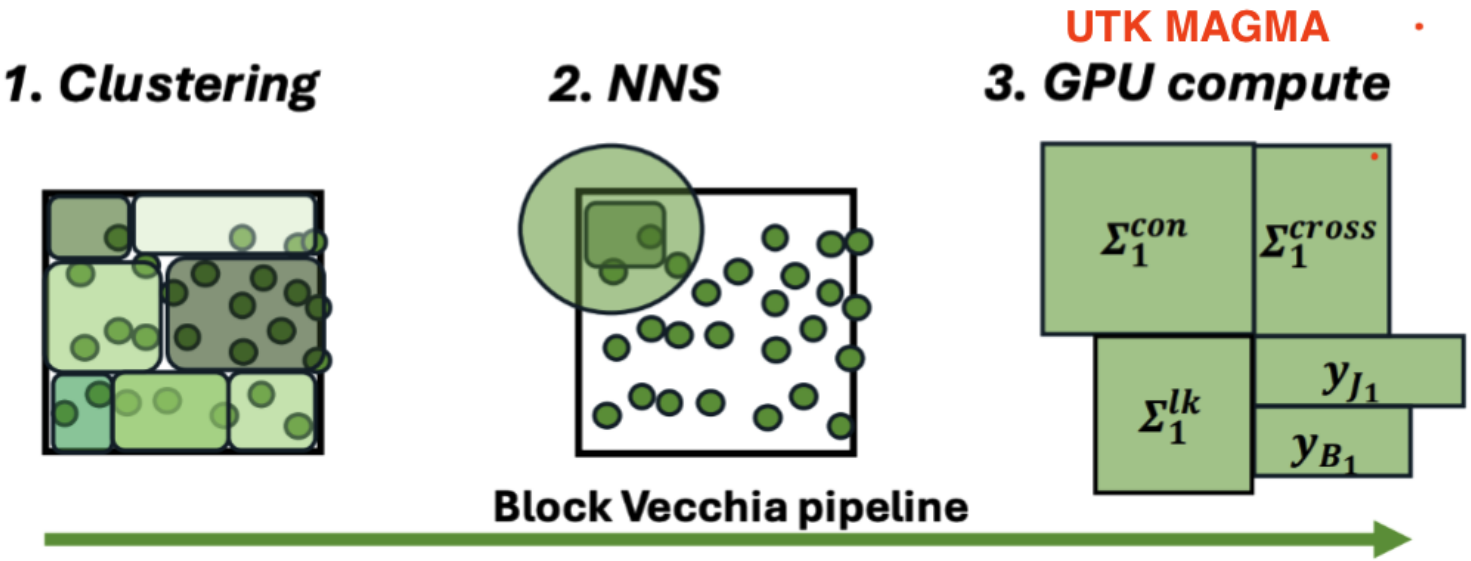
1. **Engineering Design Optimization**: High-fidelity emulation of complex systems (e.g., aerospace components or automotive designs) require fast and scalable surrogate models
2. **Environmental Modeling**: Climate emulation or environmental monitoring often involve large spatiotemporal datasets
3. **Bayesian Optimization in Drug Discovery**: In fields like drug discovery, where high-dimensional and costly simulations are run, fast surrogate modeling and efficient exploration of the chemical space is needed

Large Problem Size & High Dimension

The two challenges are addressed by the following approaches:

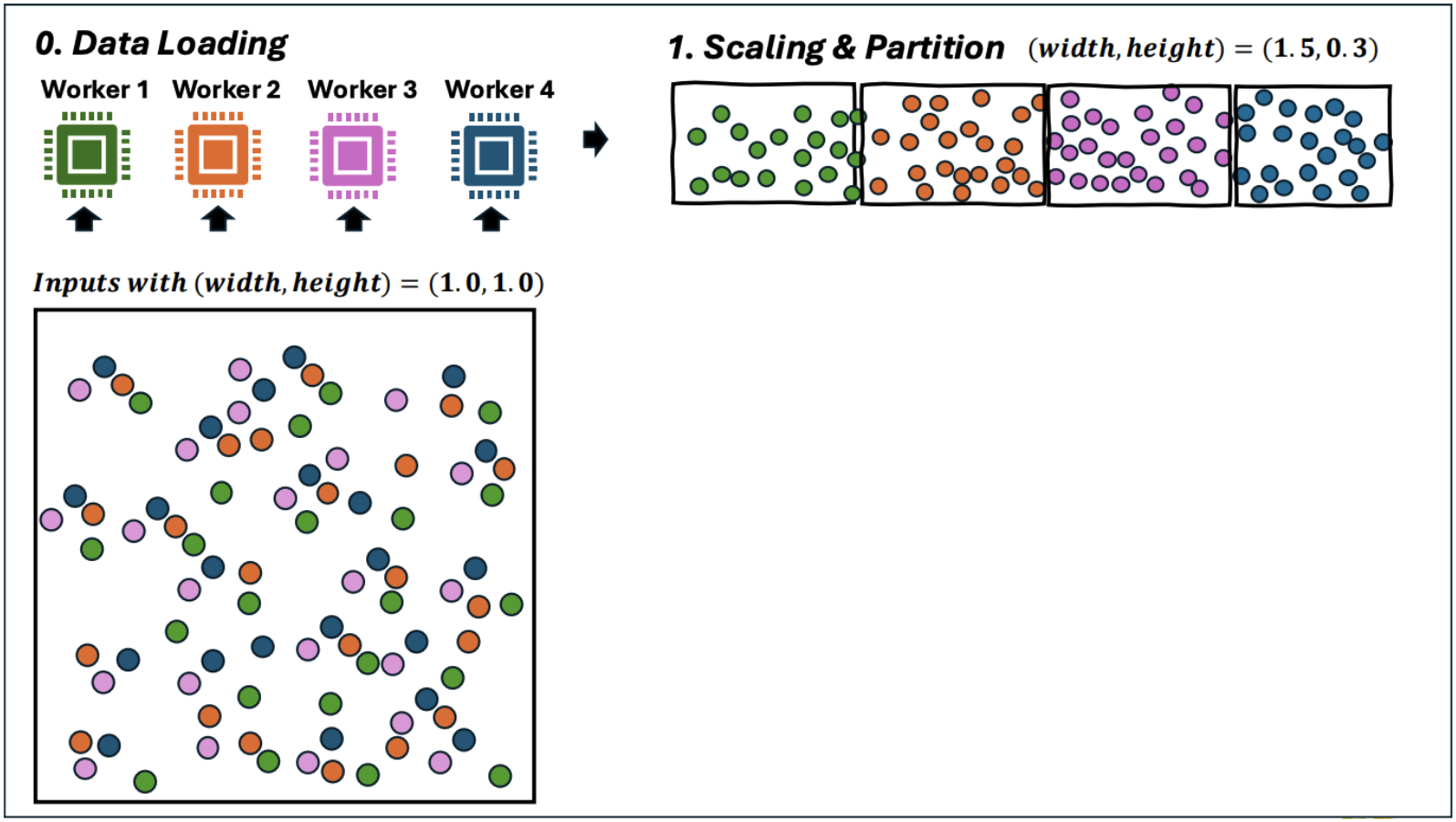
1. **High-dimensional input space in GPs** $\Rightarrow$ Modified covariance functions.
2. **Large problem size** $\Rightarrow$ Distribute the BV algorithm.

Block Vecchia (BV) Pipeline



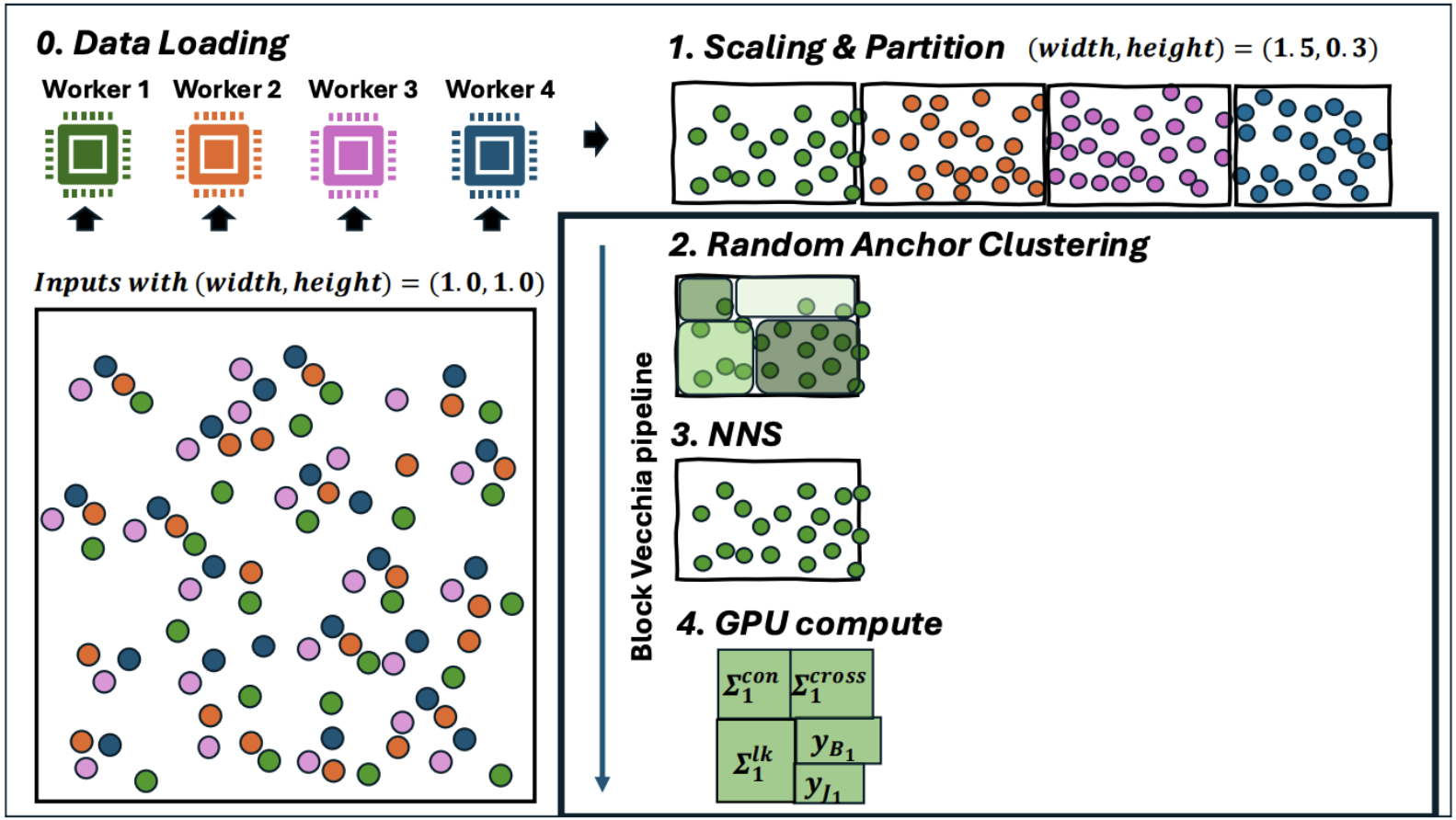
Block Vecchia (BV) computation pipeline,
1) Clustering
2) Nearest Neighbors Searching (NNS)
3) batched GPU compute

Distributed Scaled Block Vecchia (SBV) Framework 1/6



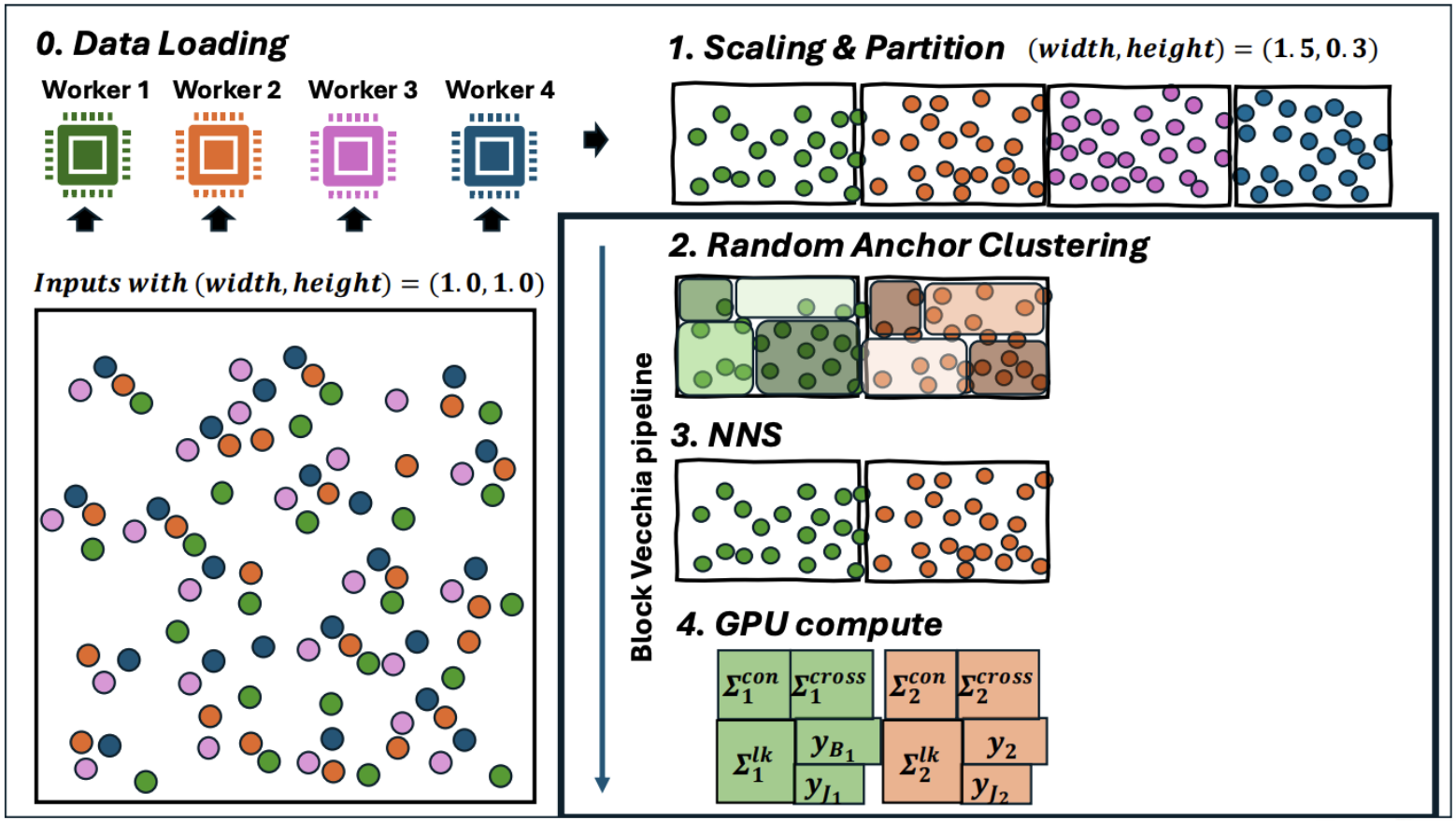
Distributed Scaled Block Vecchia (SBV) Pipeline

Distributed Scaled Block Vecchia (SBV) Framework 2/6



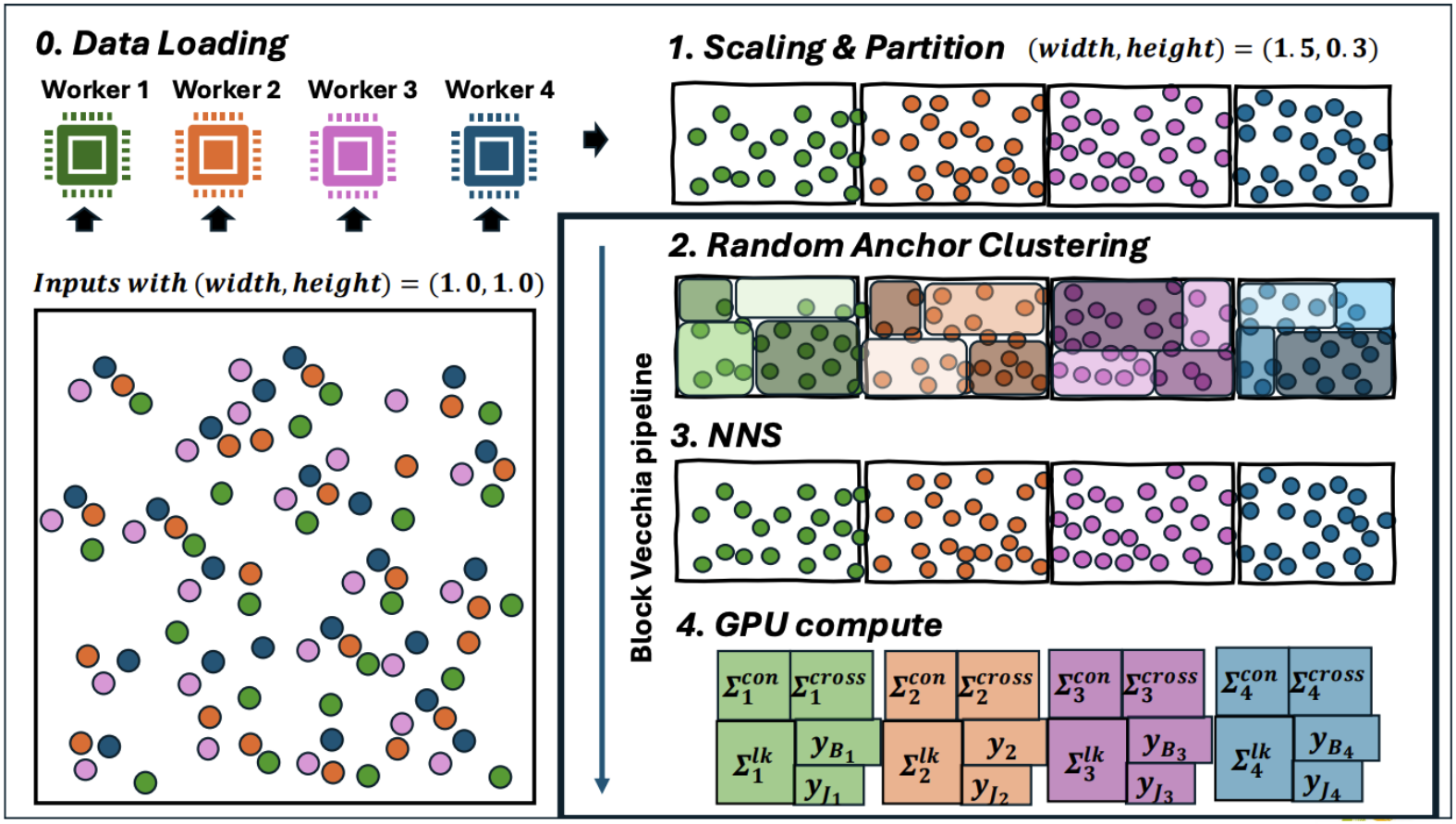
Distributed Scaled Block Vecchia (SBV) Pipeline

Distributed Scaled Block Vecchia (SBV) Framework 3/6



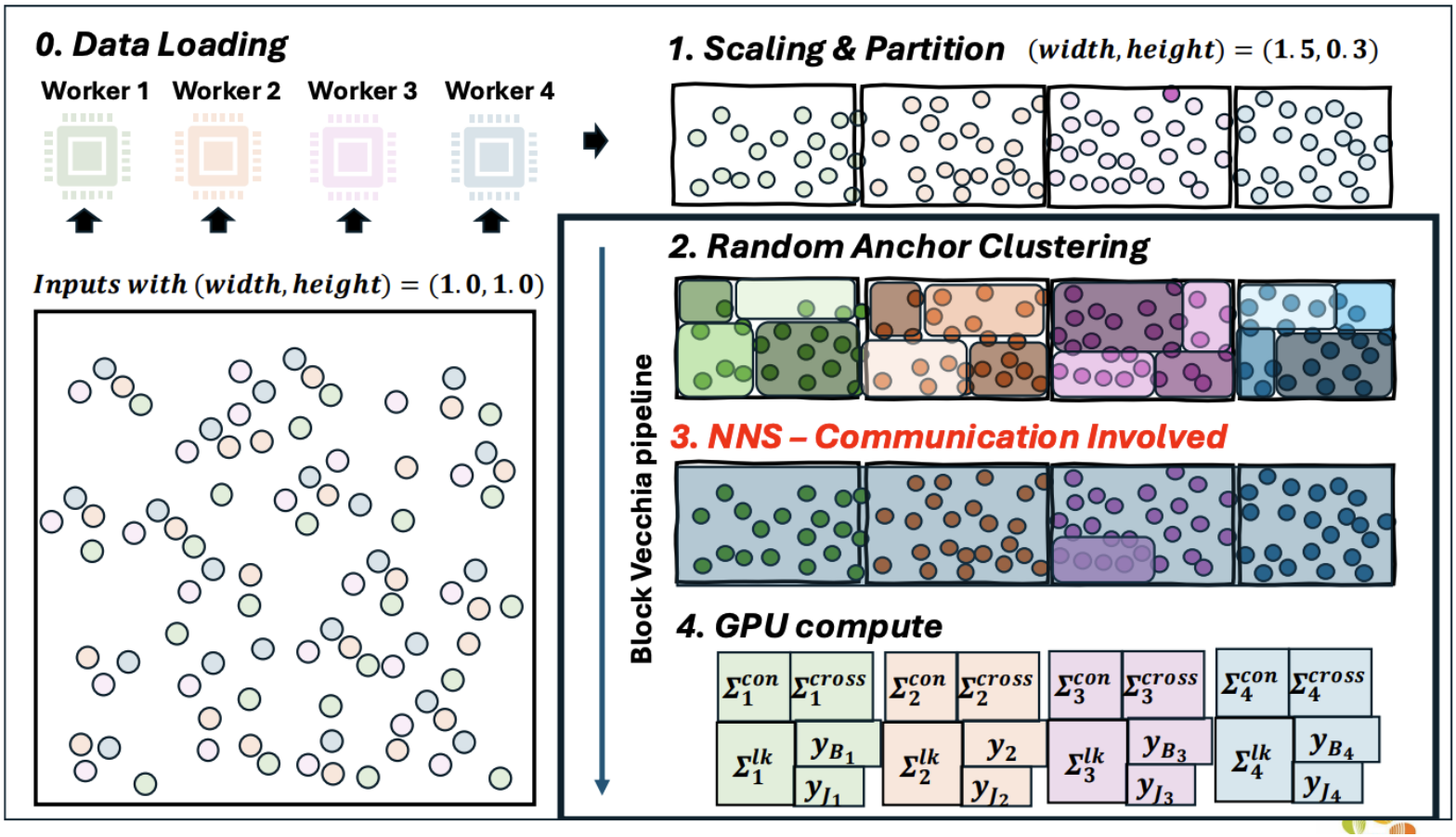
Distributed Scaled Block Vecchia (SBV) Pipeline

Distributed Scaled Block Vecchia (SBV) Framework 4/6



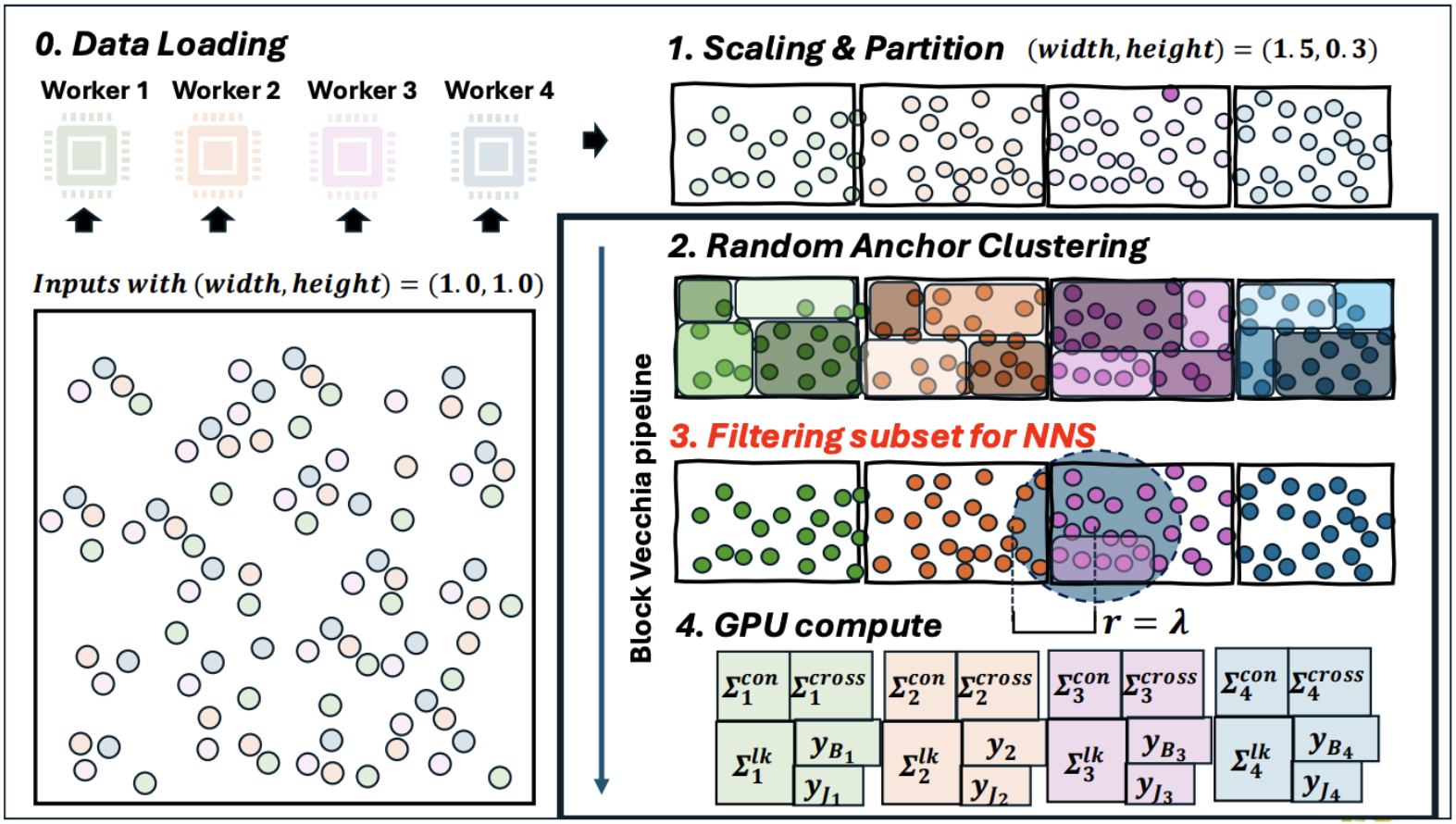
Distributed Scaled Block Vecchia (SBV) Pipeline

Distributed Scaled Block Vecchia (SBV) Framework 5/6



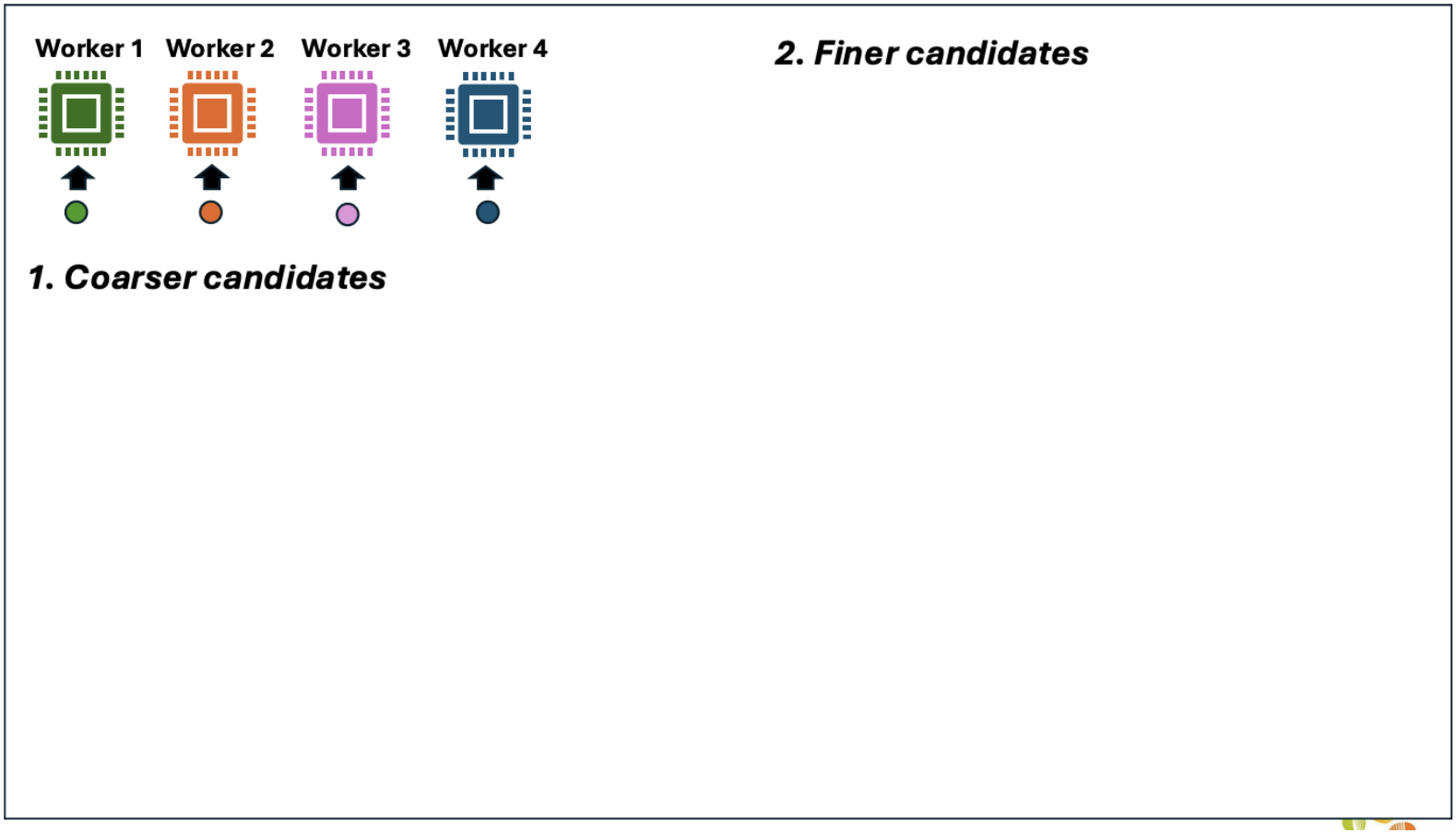
Distributed Scaled Block Vecchia (SBV) Pipeline

Distributed Scaled Block Vecchia (SBV) Framework 6/6



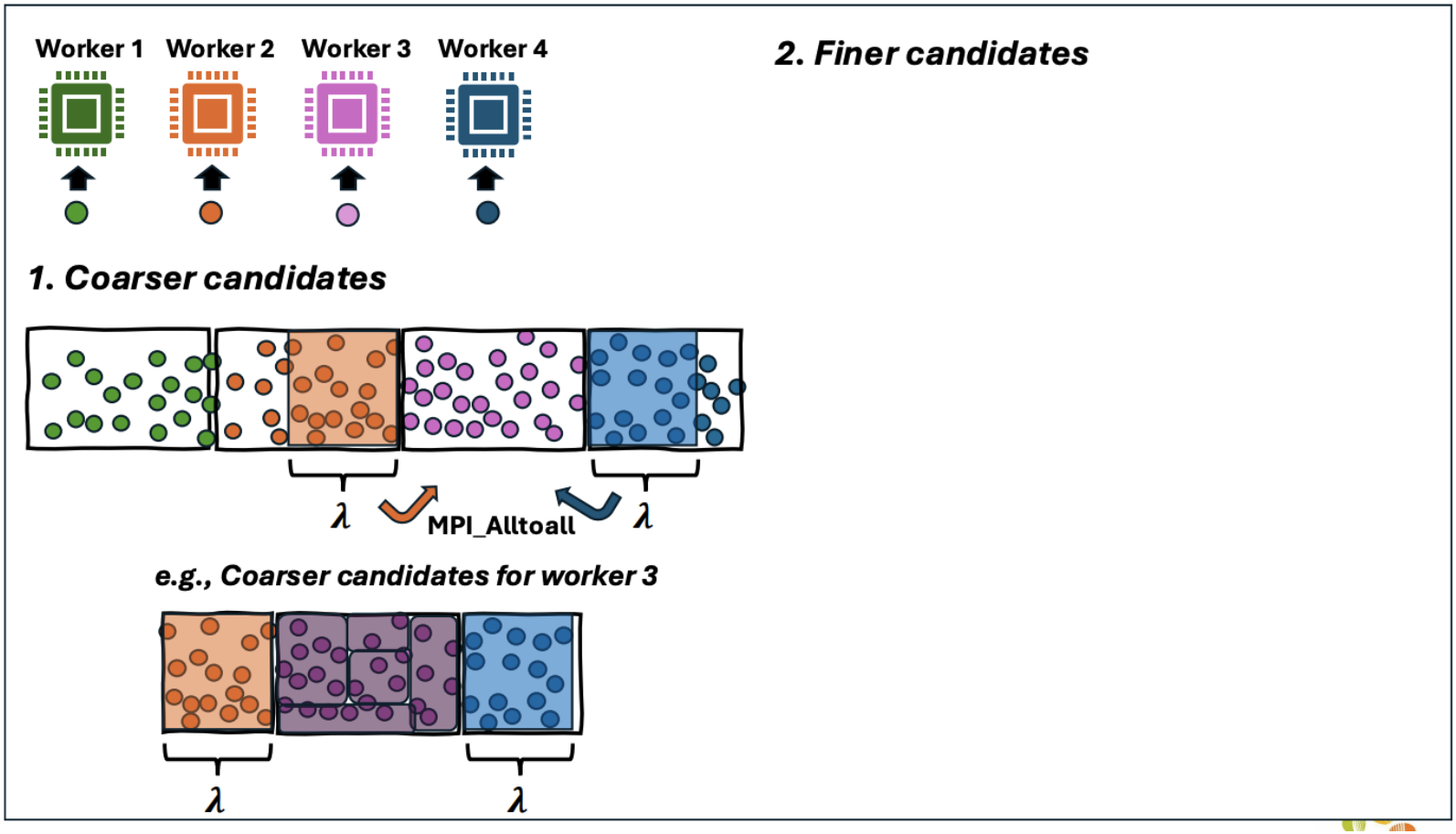
Distributed Scaled Block Vecchia (SBV) Pipeline

Filtering subset for NNS 1/5



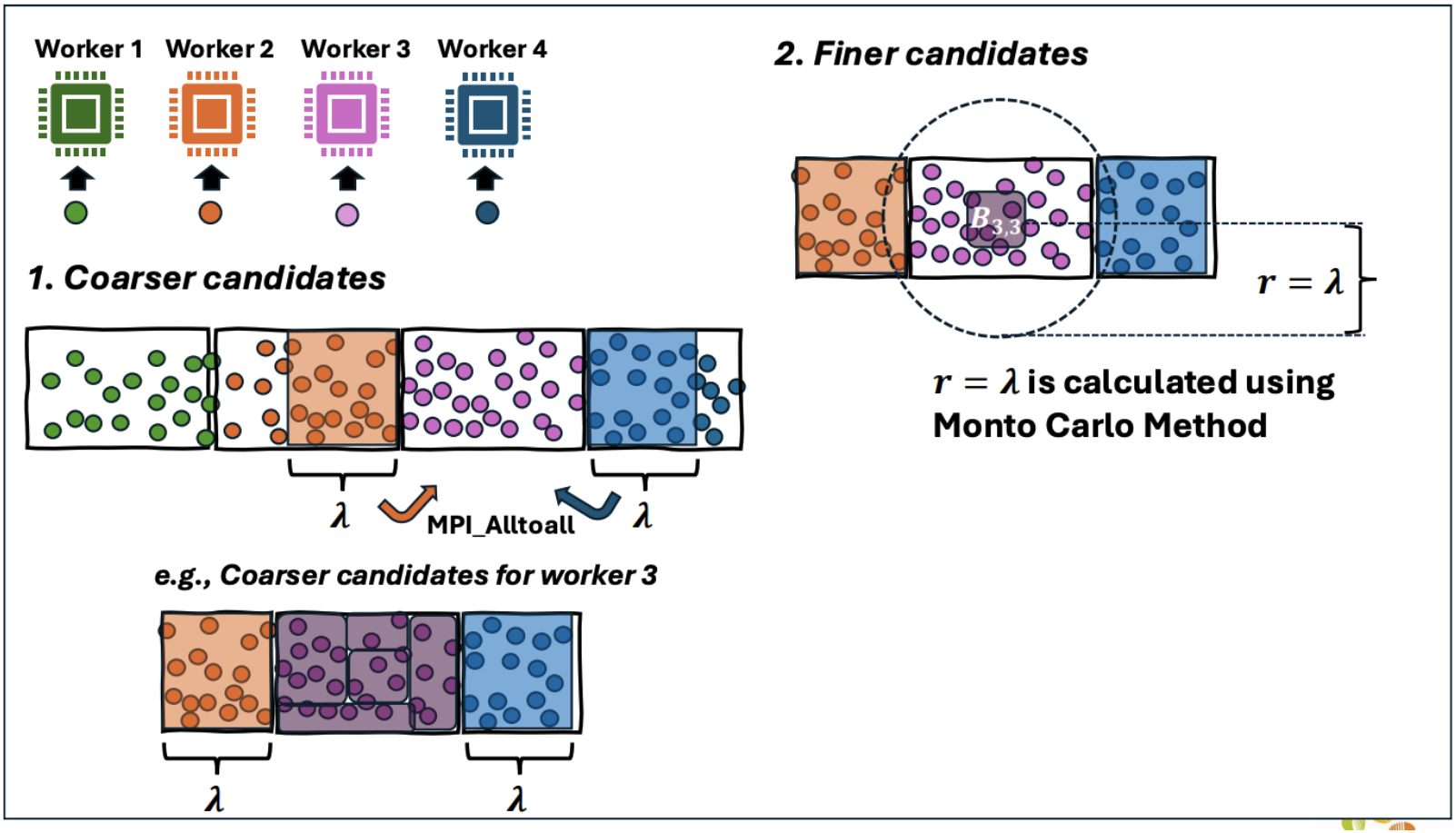
Distributed Scaled Block Vecchia (SBV) Pipeline

Filtering subset for NNS 2/5



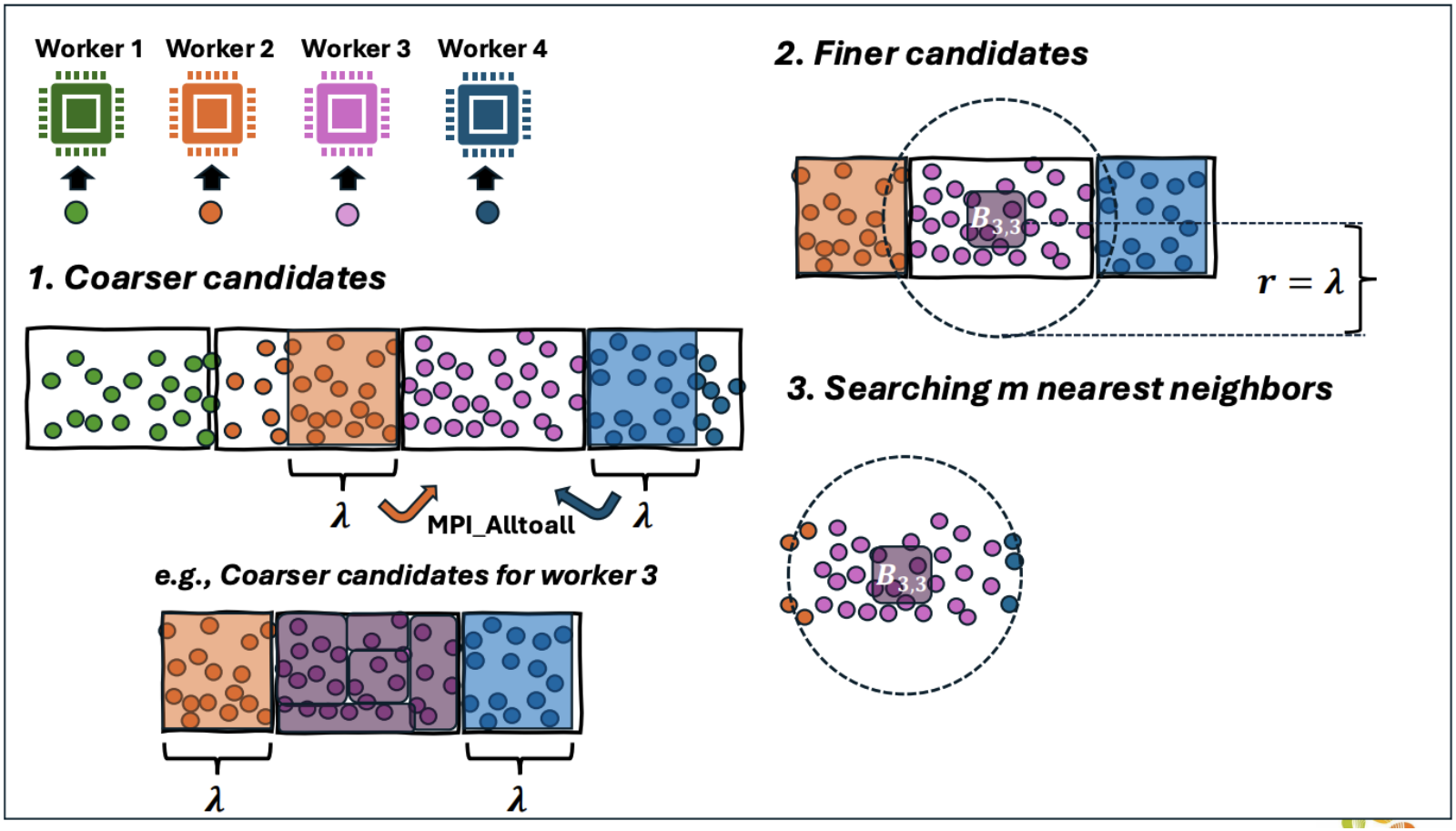
Distributed Scaled Block Vecchia (SBV) Pipeline

Filtering subset for NNS 3/5



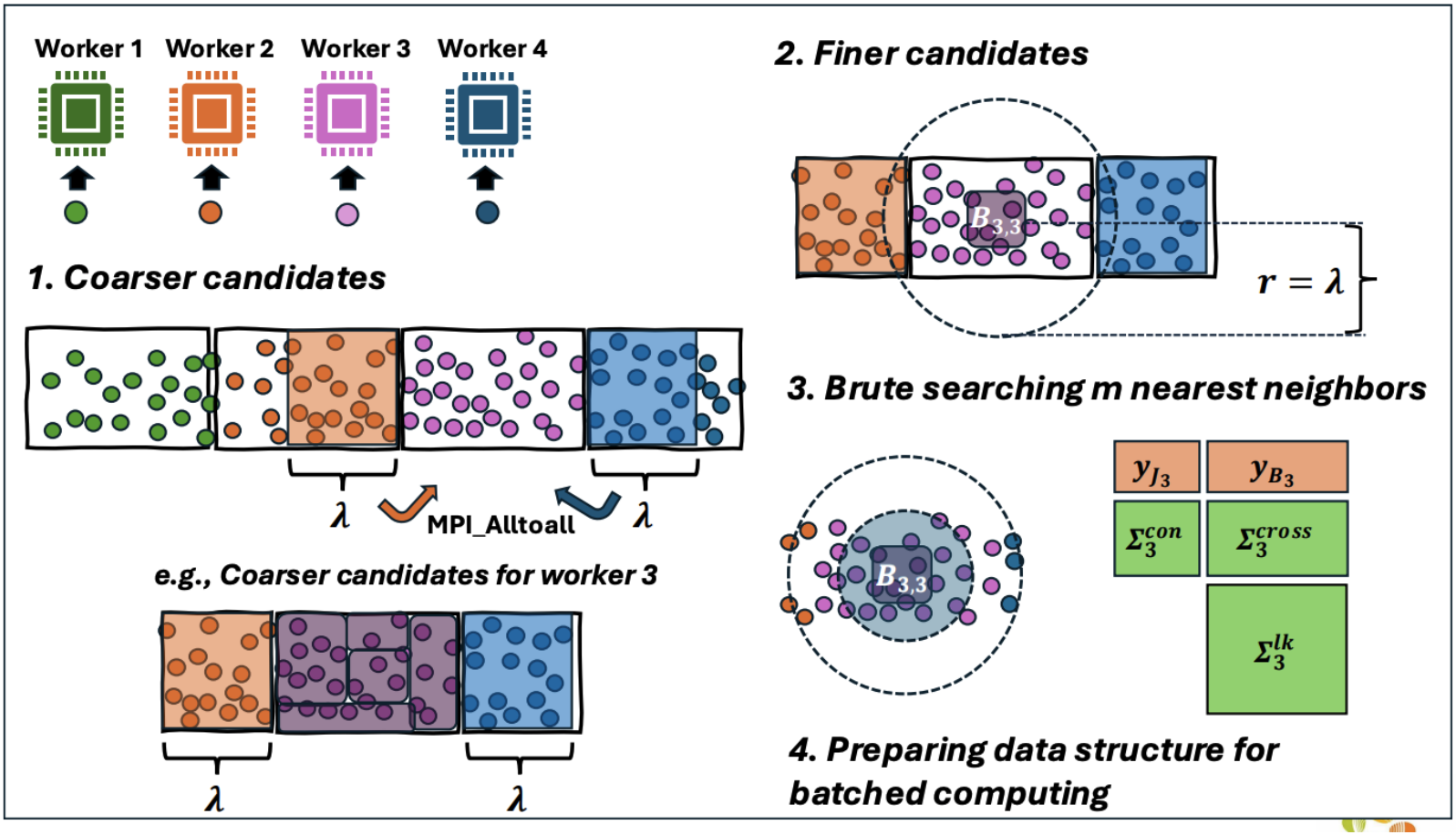
Distributed Scaled Block Vecchia (SBV) Pipeline

Filtering subset for NNS 4/5



Distributed Scaled Block Vecchia (SBV) Pipeline

Filtering subset for NNS 5/5

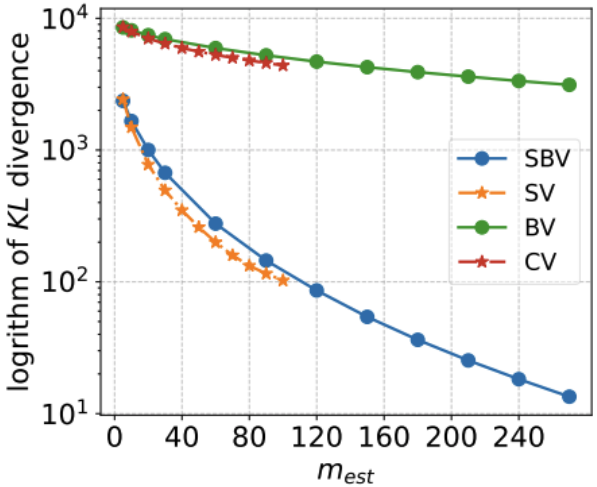


Distributed Scaled Block Vecchia (SBV) Pipeline

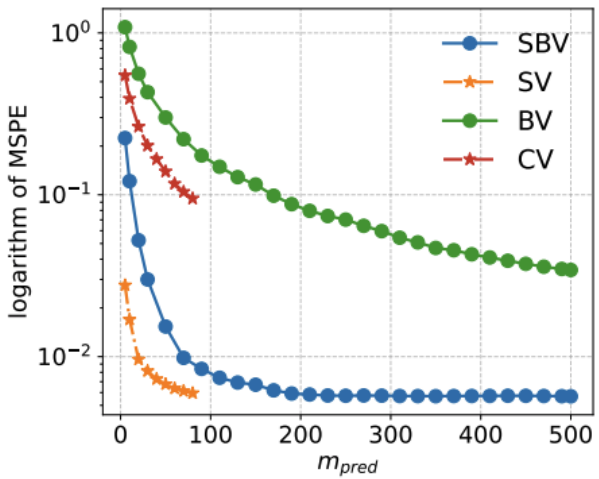
High-dimensional GP Simulations

Configuration. $x \in [0,1]^{10}$, $y \sim \mathcal{N}(0, \mathcal{K}(x_1, x_2))$ with $\mathcal{K}$ Matérn ($\nu = 3.5$). (All following keep the same GP setting.) $\beta_1 = \beta_2 = 0.05$ and $\beta_3 = \cdots = \beta_{10} = 5$.

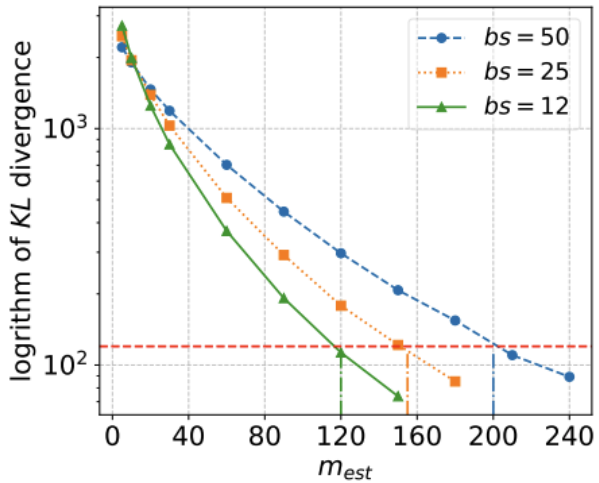
- **CV**: Classic Vecchia
- **BV**: Block Vecchia, $bs_{\text{est}} = bs_{\text{pred}} = 10$ in (a),(b)
- **SV**: Scaled Vecchia
- **SBV**: Scaled Block Vecchia, with $bs_{\text{est}} = bs_{\text{pred}} = 10$ in (a),(b)



(a) KL divergence for fitting.



(b) Prediction (MSPE).



(c) Effect of block size (bs) on SBV accuracy.

Figure 1: KL divergence for fitting.

Satellite Drag (Benchmark Dataset) (1/3)

The satellite drag dataset is a widely used benchmark for evaluating GP-based models in high-dimensional settings [1] and [2]. It is generated from the low Earth orbit simulator and comprises 6 sub-datasets/species, each comprising **2 million simulation runs**.

The 6 species are O, O_2, N, N_2, He, and H with **8-dimensional input** $x \in [0,1]^8$, including: relative velocity; surface temperature; atmospheric temperature; yaw angle; pitch angle; two accommodation coefficients. Here, we still adopt the same GP setting as mentioned previously.

[1] Sun, F., Gramacy, R. B., Haaland, B., Lawrence, E., & Walker, A. (2019). Emulating satellite drag from large simulation experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 7(2), 720–759.

[2] Katzfuss, M., Guinness, J., & Lawrence, E. (2022). Scaled Vecchia approximation for fast computer-model emulation. *SIAM/ASA Journal on Uncertainty Quantification*, 10(2), 537–554.

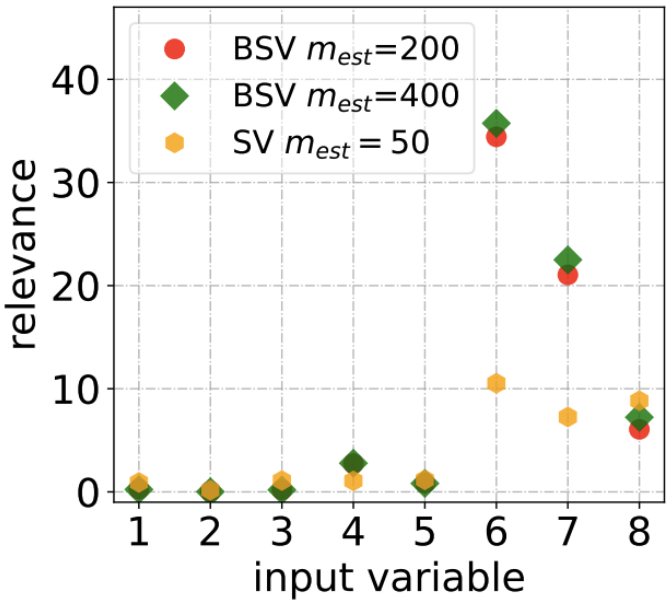
Satellite Drag (Benchmark Dataset) (2/3)

The approximation methods considered are listed as follows (SV is the current best approximation to our knowledge),

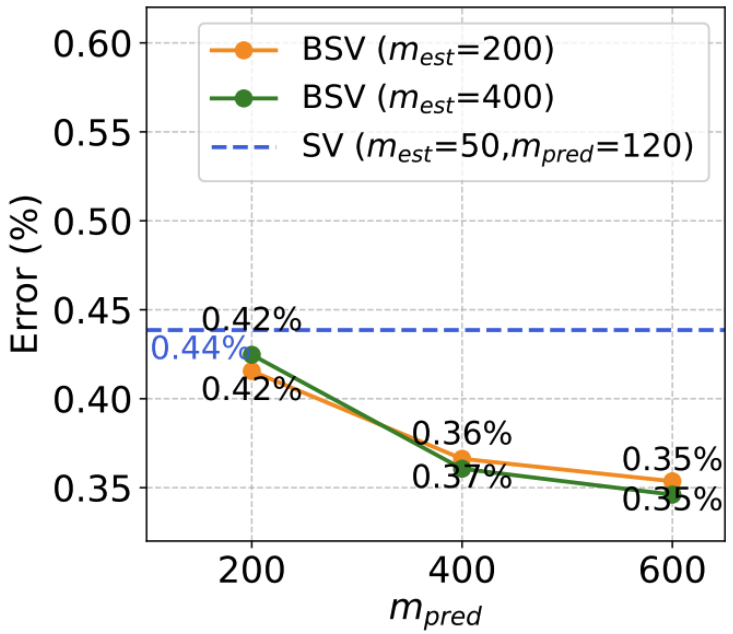
Table 1: Vecchia-based GP configurations on the satellite drag dataset, showing block sizes ($b_{s_{\text{est}}}, b_{s_{\text{pred}}}$) and neighbor counts ($m_{\text{est}}, m_{\text{pred}}$) for estimation and prediction.

Model	SV	SBV_1	SBV_2	SBV_3	SBV_4	SBV_5	SBV_6
$b_{s_{\text{est}}}$	1	100	100	100	100	100	100
$b_{s_{\text{pred}}}$	1	5	5	5	5	5	5
m_{est}	50	**200**	**200**	**200**	**400**	**400**	**400**
m_{pred}	140	**200**	**400**	**600**	**200**	**400**	**600**

Satellite Drag (Benchmark Dataset) (3/3)



(d) Parameter Estimation of N_2



(e) RMSPE of N_2

Remark

1) Inputs (6, 7, 8) — **pitch angle** and two **accommodation coefficients** — are relevant to N_2. 2) Larger $m_{\text{est}}, m_{\text{pred}}$ improve prediction accuracy.

Scalability & Power Consumption 1/6

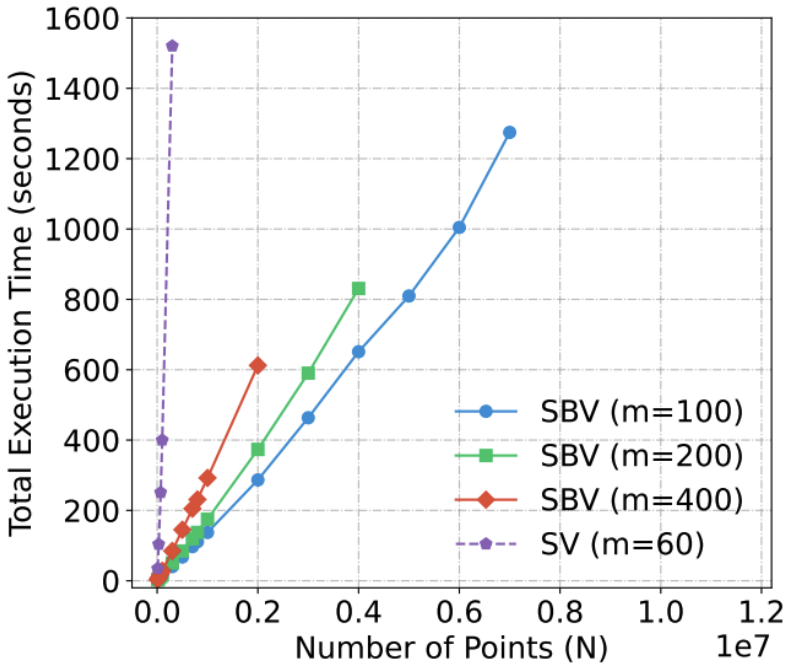
We conduct experiments on two modern NVIDIA GPU architectures, **A100 and GH200**, hosted at the **Jülich Supercomputing Centre (JSC)** on the **JURECA-DC GPU and JEDI systems**.

1. **JEDI.** 48 nodes, each equipped with four NVIDIA GH200 Grace Hopper Superchips. Each Superchip integrates:
   - 72-core Grace CPU (3.1 GHz, 120 GB)
   - Hopper GPU (96 GB HBM3)
   - NVLink-C2C interconnect (900 GB/s)
   - $\sim$680 W power consumption per Superchip
2. **JURECA-DC GPU.** 192 nodes, each with:
   - Two 64-core AMD EPYC 7742 CPUs (2.25 GHz, 512 GB, 225 W/socket)
   - Four NVIDIA A100 GPUs (40 GB HBM2, 400 W/GPU)

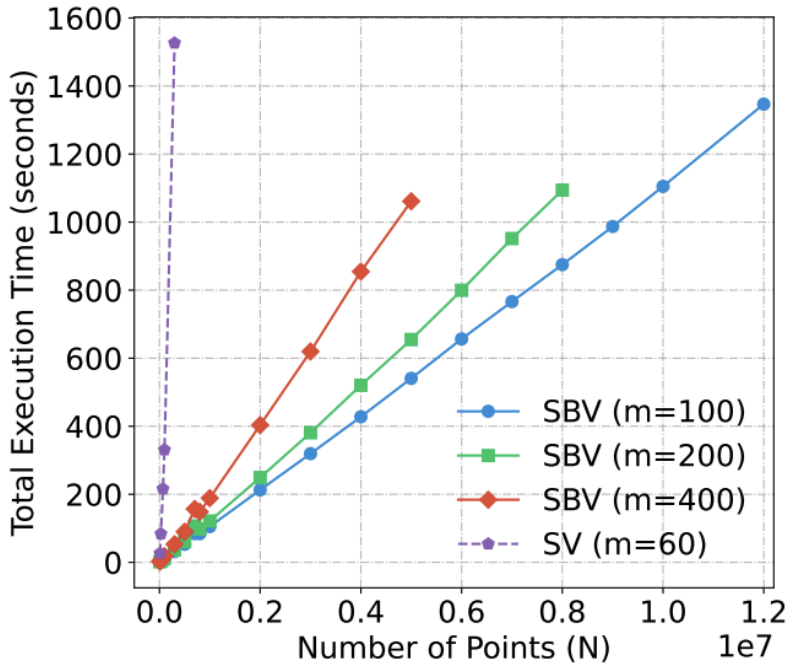
Both systems use InfiniBand interconnects:

- JEDI: four NDR200 (200 Gbit/s) links per node
- JURECA-DC GPU: two HDR (200 Gbit/s) links per node

Scalability & Power Consumption (2/6)



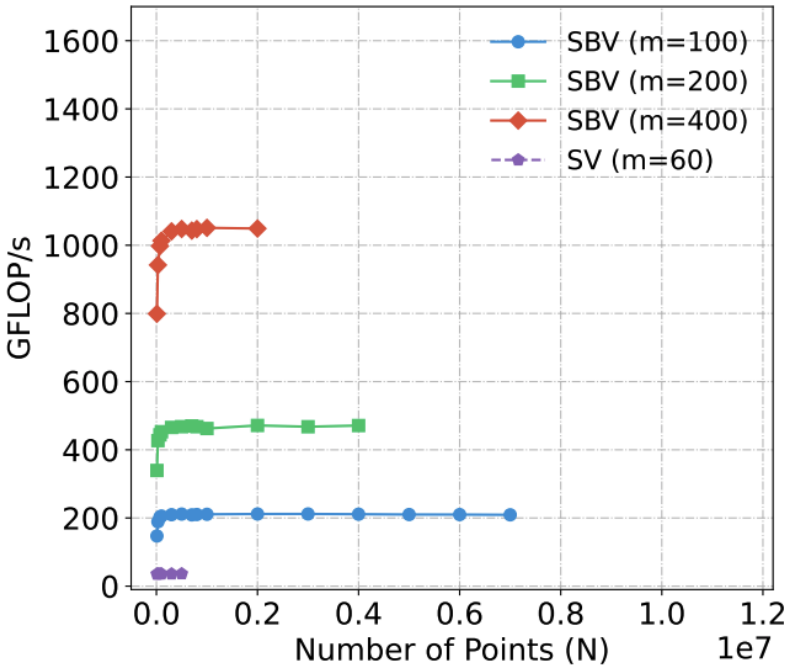
(a) A100 GPU (CPU+GPU).



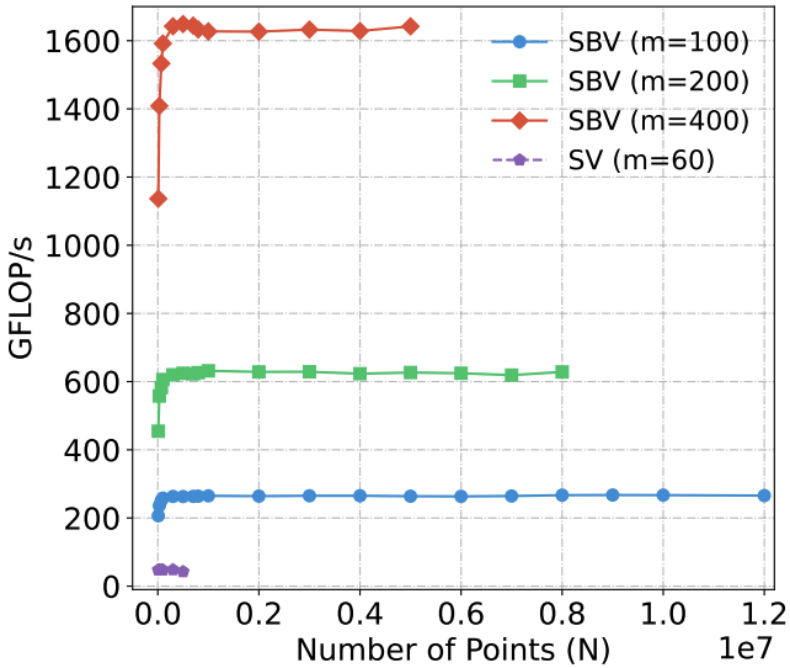
(b) GH200

Performance comparison of SBV and SV methods with 500 MLE iterations on a single AMD EPYC with NVIDIA A100 (40 GB) and single **NVIDIA GH200 superchip (96 GB)**.

Scalability & Power Consumption (3/6)



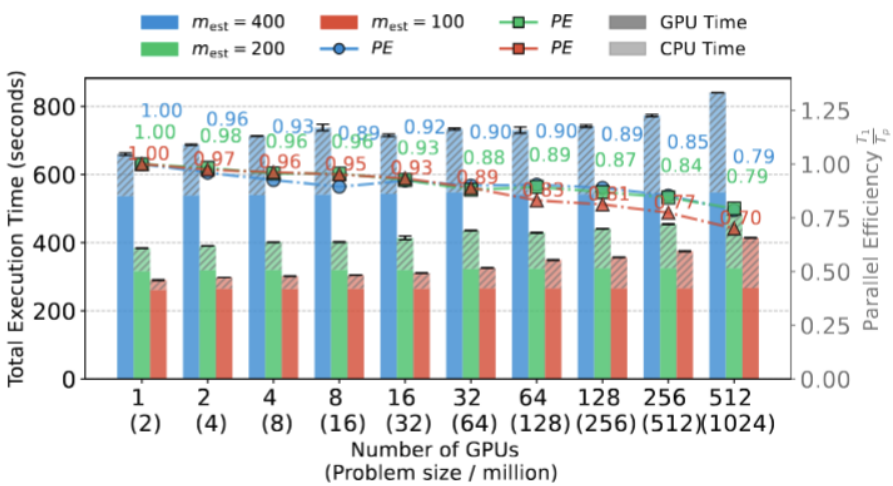
(a) A100 GPU (CPU+GPU).



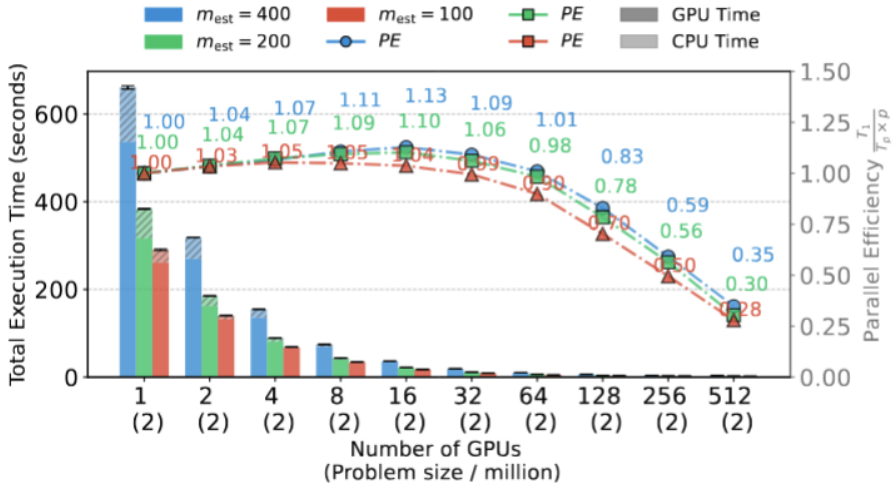
(b) GH200

Performance comparison of SBV and SV methods with 500 MLE iterations on single AMD EPYC with NVIDIA A100 (40 GB) and single NVIDIA GH200 superchip (96 GB).

Scalability & Power Consumption (4/6)



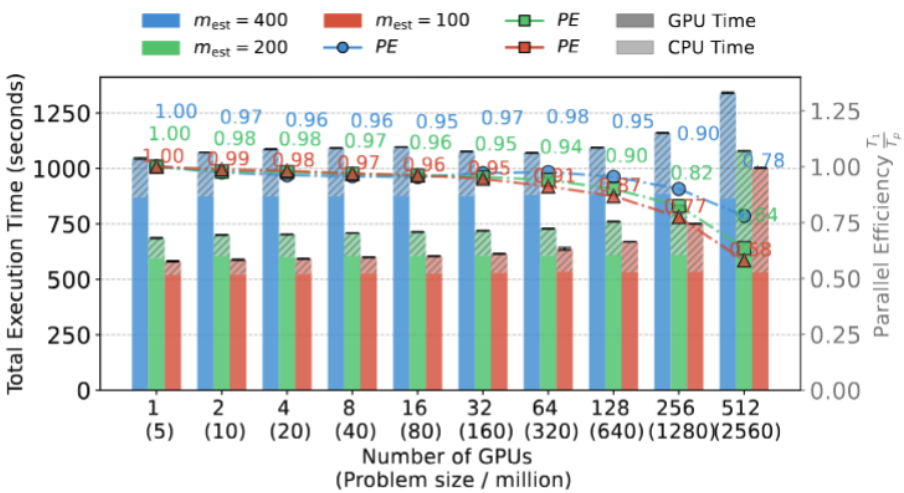
(a) Weak scalability and PE (A100) (2M–1024M points).



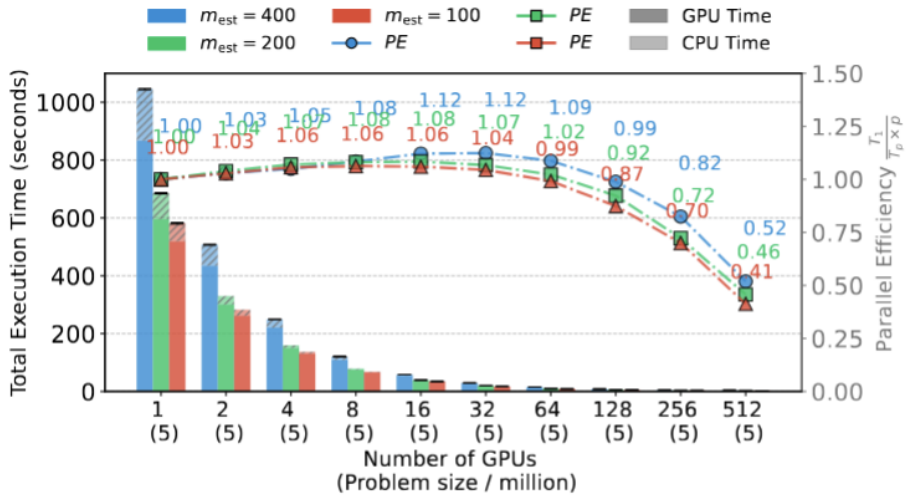
(b) Strong Scalability and PE (A100) (2M points).

Weak and strong scaling of SBV were evaluated on up to 512 GPUs, using 500 MLE iterations on AMD EPYC with NVIDIA A100 (40 GB, up to 128M points).

Scalability & Power Consumption (5/6)



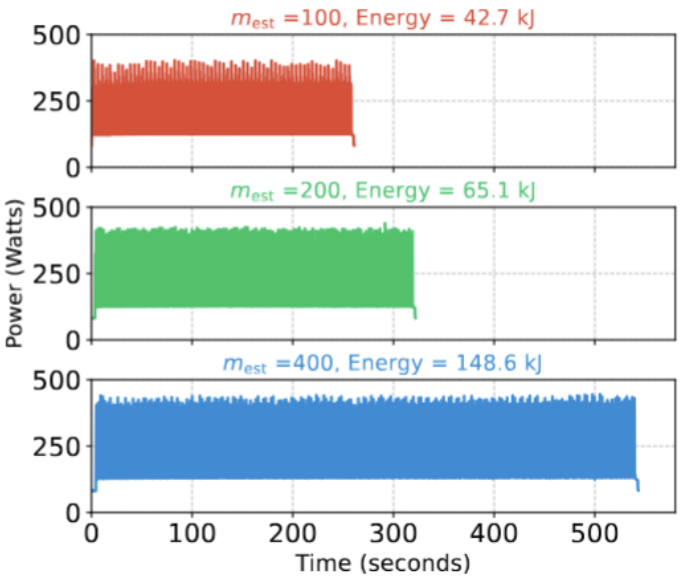
(c) Weak scalability and PE (GH200) (5M–2.56B points).



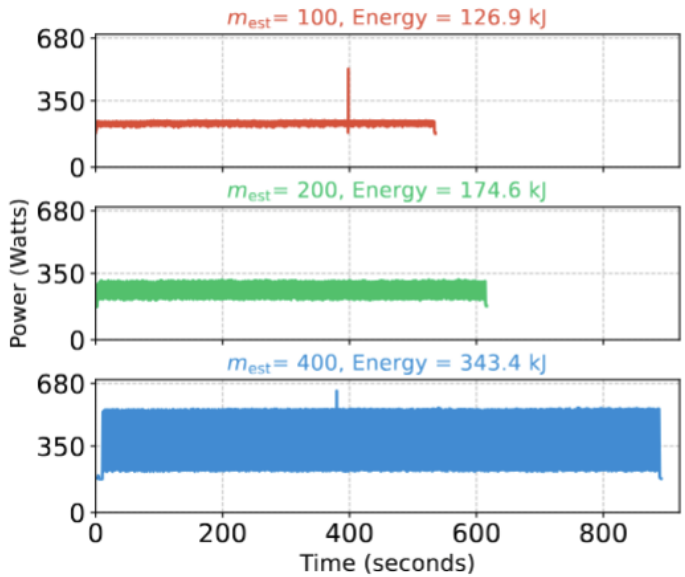
(d) Strong Scalability and PE (GH200) (5M points).

Weak and strong scaling of SBV were evaluated on up to 512 GPUs, using 500 MLE iterations on NVIDIA GH200 superchip (96 GB, up to 2.56B points).

Scalability & Power Consumption (6/6)



(a) Single NVIDIA A100 GPU (2M points).



(b) Single NVIDIA H100 GPU (5M points).

Power consumption/energy (kJ) on a single GPU for two NVIDIA GPUs over 500 iterations per m_{est}. A100 GPU with 400 W max power cap and Hopper GPU with (680 W - the power usage of the Grace CPU, RAM, and system I/O) max power cap.

Take Home Messages

- **Climate and weather models** are increasingly limited by communication and I/O bottlenecks in traditional HPC settings
- **Modern GPUs favor compute-bound workloads**, optimizing for this enables scalable, energy-efficient performance
- **Gaussian Process (GP) models** are powerful but computationally expensive. Approximations such as Vecchia are essential for large-scale use
- **Block Vecchia (BV) and Scaled BV (SBV)** unlock GP modeling for billions of data points using hierarchical batching and GPU acceleration
- **SBV demonstrates strong scalability and reduced power consumption** across multiple NVIDIA GPU architectures (A100, GH200)
- **Applications in environmental modeling, engineering design, and scientific emulation** benefit from fast, scalable, and accurate GP modeling

References

[Pan et al., 2024] Pan, Q., Abdulah, S., Genton, M. G., Keyes, D. E., Ltaief, H., & Sun, Y. (2024).

GPU-accelerated Vecchia approximations of Gaussian processes for geospatial data using batched matrix computations.

In *ISC High Performance 2024 Research Paper Proceedings (39th International Conference)* (pp. 1–12). Prometeus GmbH.

[Pan et al., 2025a] Pan, Q., Abdulah, S., Genton, M. G., & Sun, Y. (2025).

Block Vecchia Approximation for Scalable and Efficient Gaussian Process Computations.

Technometrics, 1–13. Taylor & Francis.

[Pan et al., 2025b] Pan, Q., Abdulah, S., Abduljabbar, M., Ltaief, H., Herten, A., Bode, M., Pratola, M., Fadikar, A., Genton, M. G., Keyes, D. E., et al. (2025).

Scaled Block Vecchia Approximation for High-Dimensional Gaussian Process Emulation on GPUs.

arXiv preprint arXiv:2504.12004.